Importing libraries and modules

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

Loading the dataset

```
df=pd.read_csv('/content/winequality-red.csv')
df.head(10)
```

```
   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
5            7.4              0.66         0.00             1.8
0.075
6            7.9              0.60         0.06             1.6
0.069
7            7.3              0.65         0.00             1.2
0.065
8            7.8              0.58         0.02             2.0
0.073
9            7.5              0.50         0.36             6.1
0.071

   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

5                 13.0                  40.0   0.9978  3.51       0.56
```

```
6                  15.0                       59.0   0.9964  3.30       0.46

7                  15.0                       21.0   0.9946  3.39       0.47

8                   9.0                       18.0   0.9968  3.36       0.57

9                  17.0                      102.0   0.9978  3.35       0.80


   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
5      9.4        5
6      9.4        5
7     10.0        7
8      9.5        7
9     10.5        5
```

```
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.shape
```

```
(1599, 12)
```

```
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
```

Data Preprocessing

Checking for Null values

```
df.isnull().sum()
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
```

```
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

Visualizations

Univariate Analysis - 1 (Pie Chart)

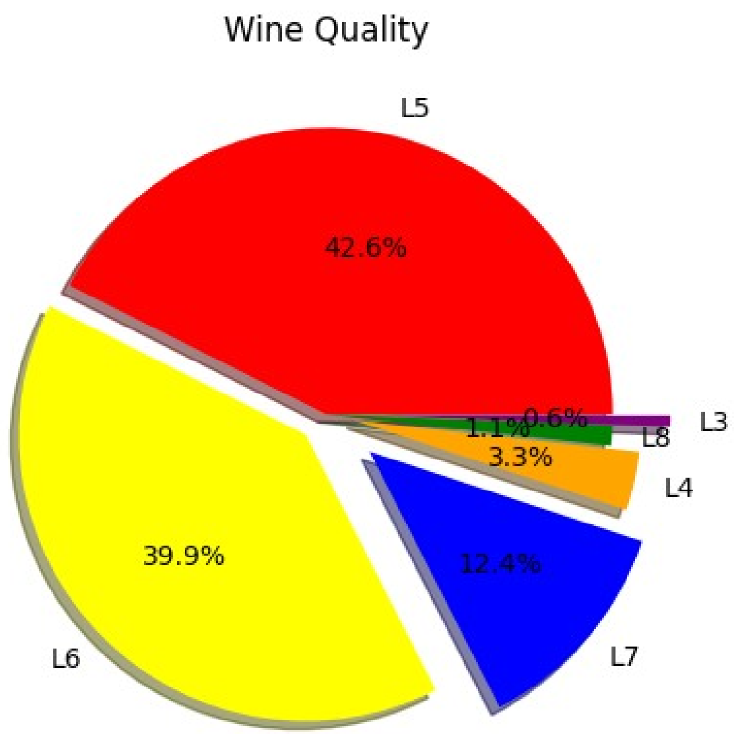
```
df['quality'].unique()
```

```
array([5, 6, 7, 4, 8, 3])
```

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

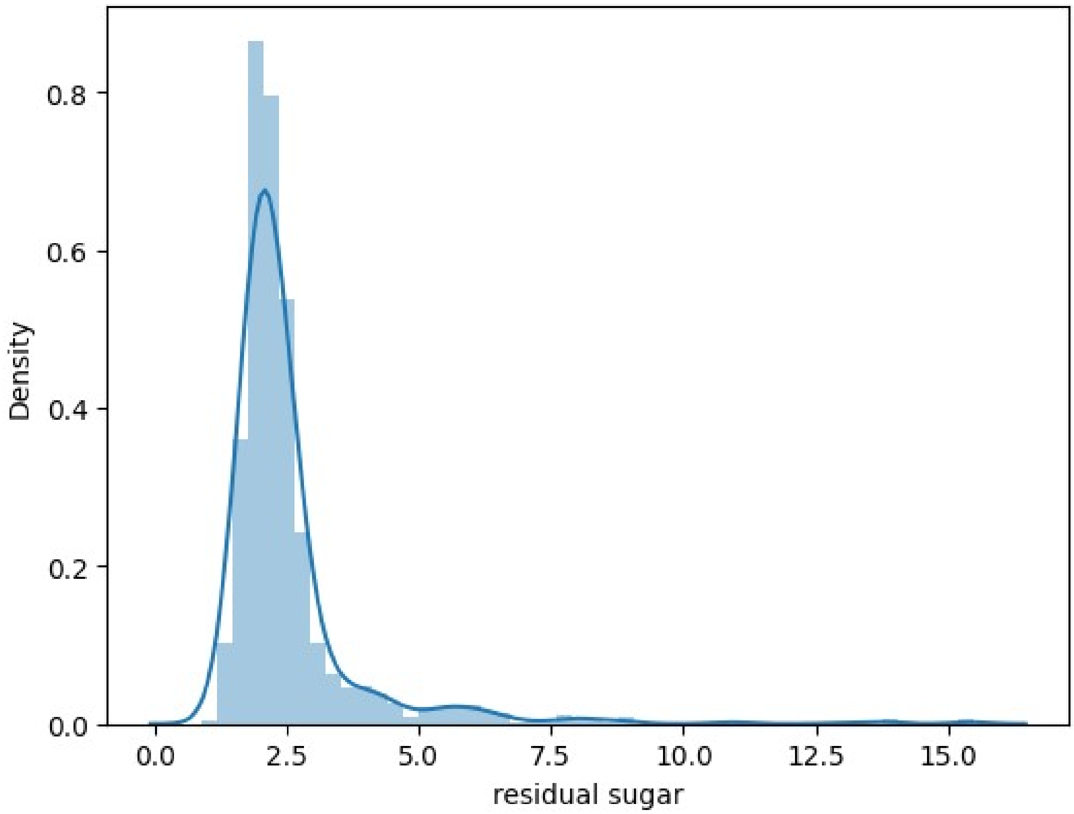
```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

```
plt.pie(df['quality'].value_counts(),
[0,0.1,0.2,0.1,0,0.2],labels=['L5','L6','L7','L4','L8','L3'],autopct='
%1.1f%
%',shadow=True,colors=['red','yellow','blue','orange','green','purple'
])
plt.title('Wine Quality')
plt.show()
```



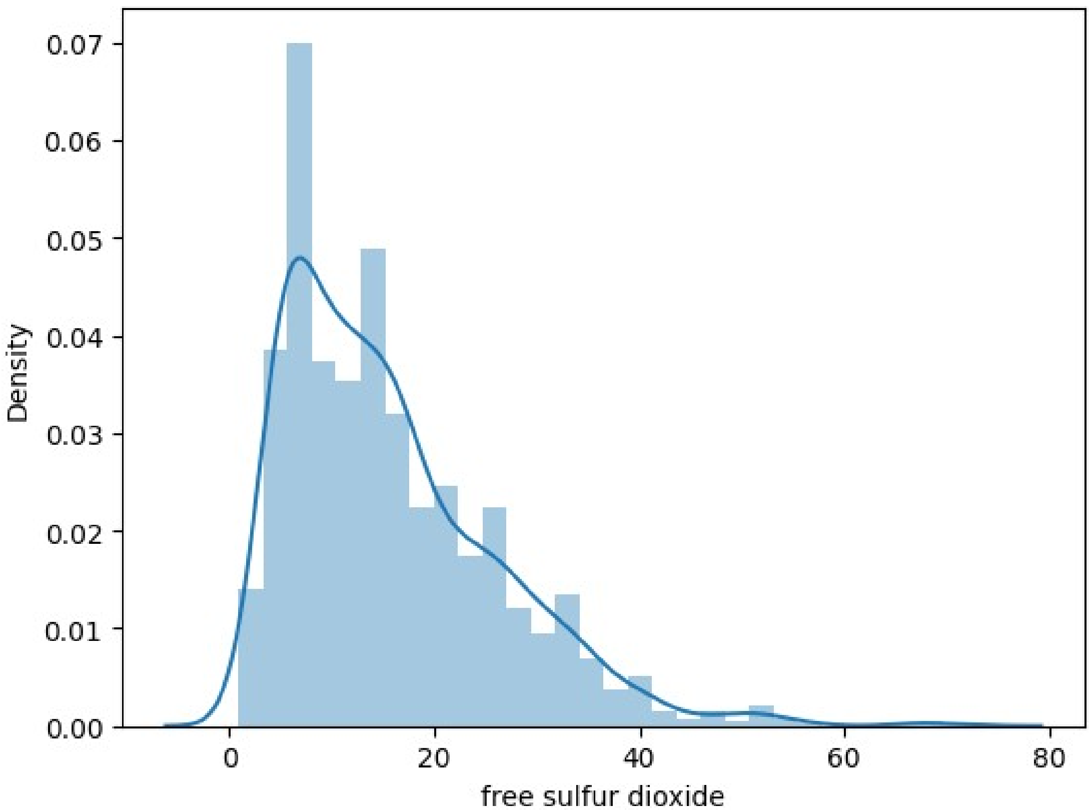
Univariate Analysis -2 (distplot)

```
sns.distplot(df['residual sugar'])
plt.show()

<ipython-input-791-3ebe262dcd43>: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['residual sugar'])
```



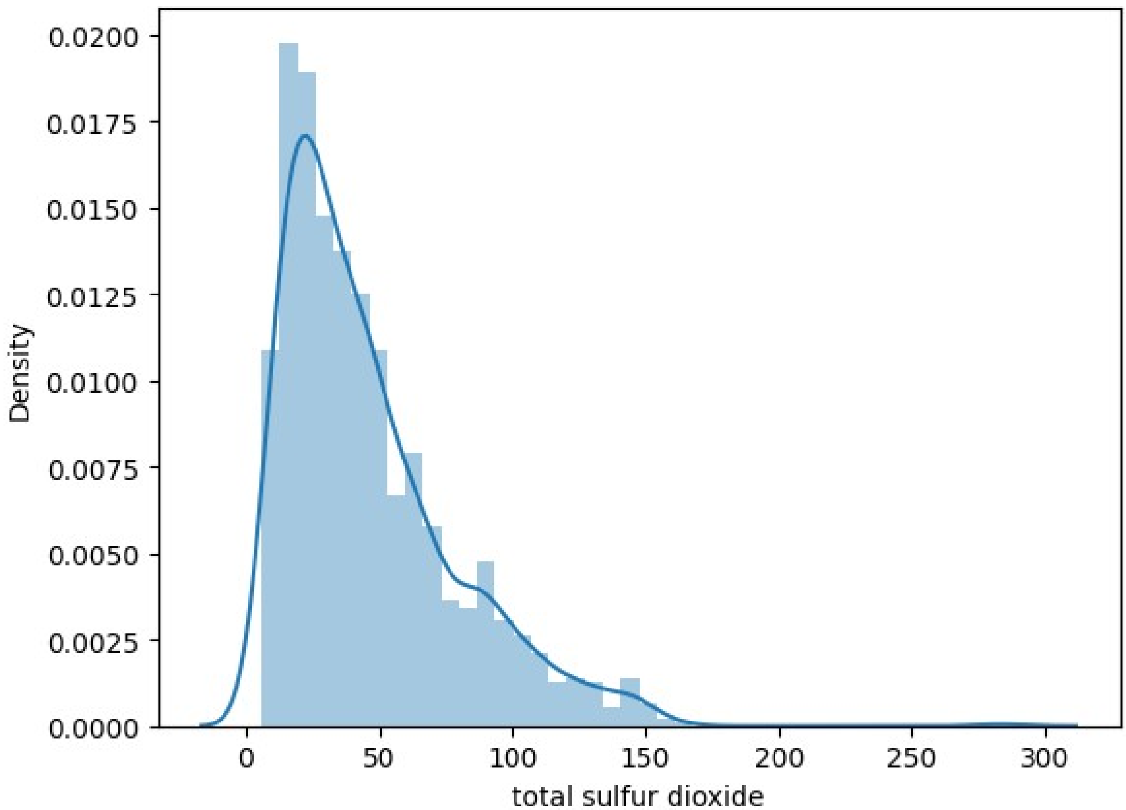

```
sns.distplot(df['free sulfur dioxide'])
plt.show()
```

```
<ipython-input-792-12e549d3d17b>: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['free sulfur dioxide'])
```




```
sns.distplot(df['total sulfur dioxide'])
plt.show()
```

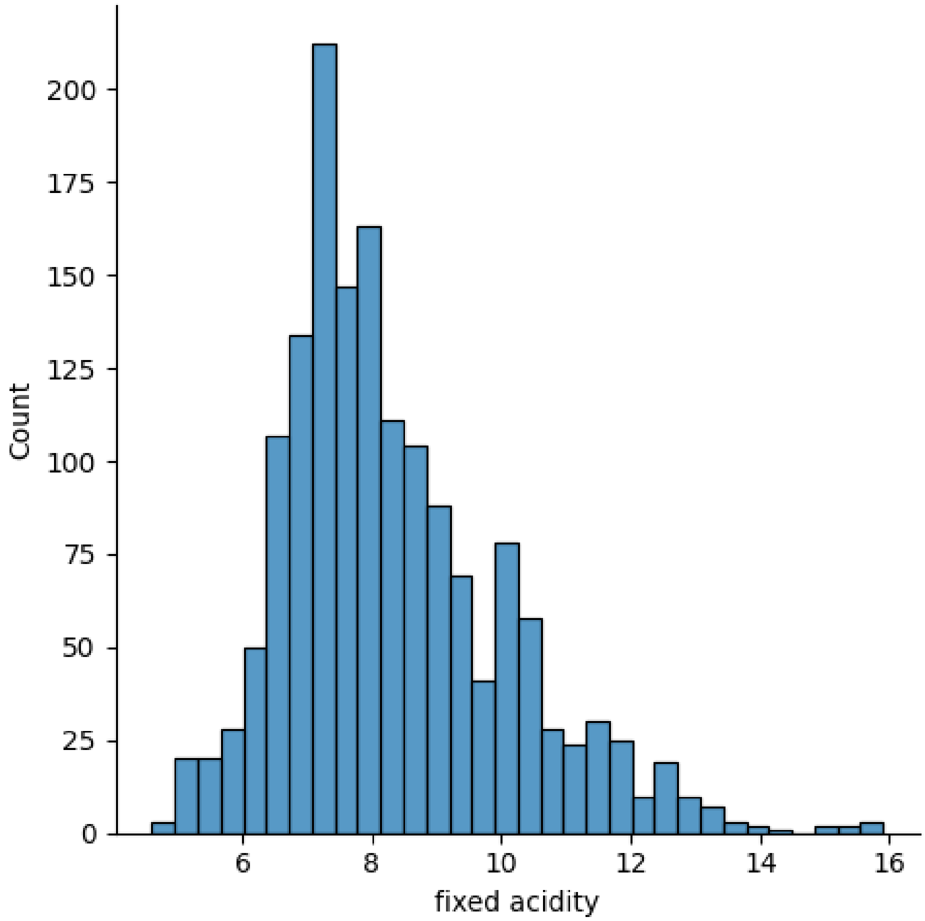
```
<ipython-input-793-f2f9a4b197ba>:1: UserWarning:

`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['total sulfur dioxide'])
```



Univariate Analysis -3 (displot)

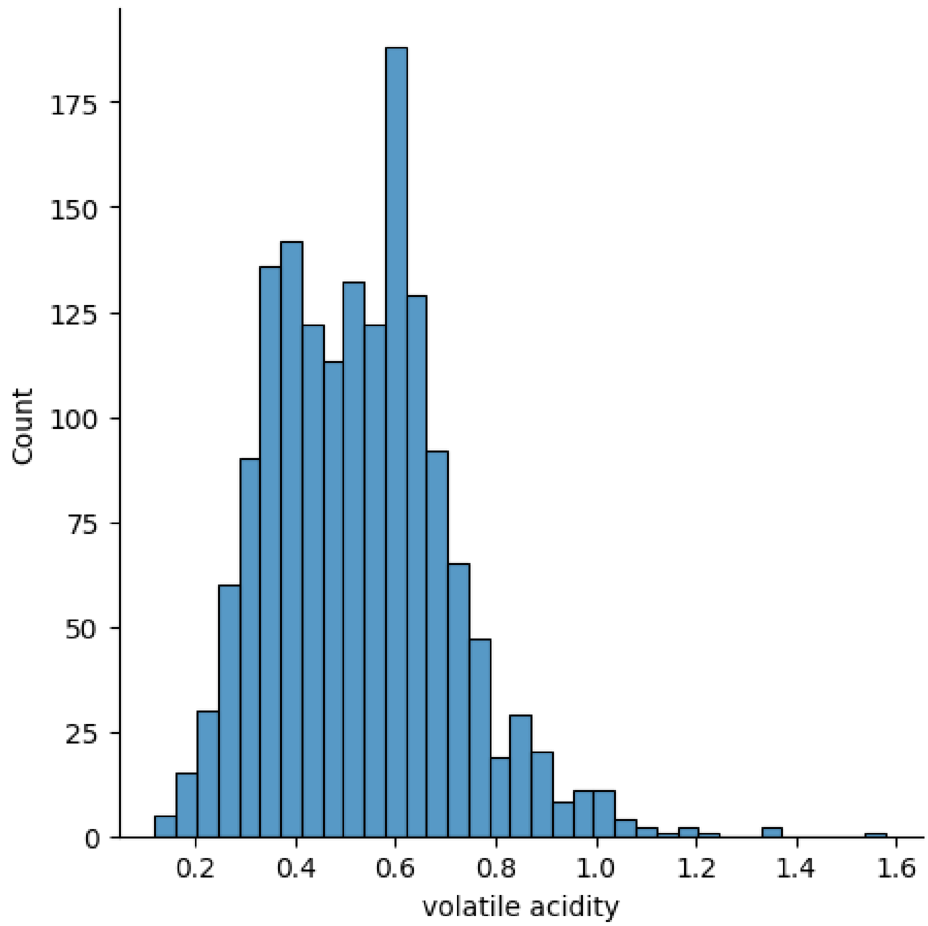
```
sns.displot(df['fixed acidity'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7c543fb7e7a0>
```




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

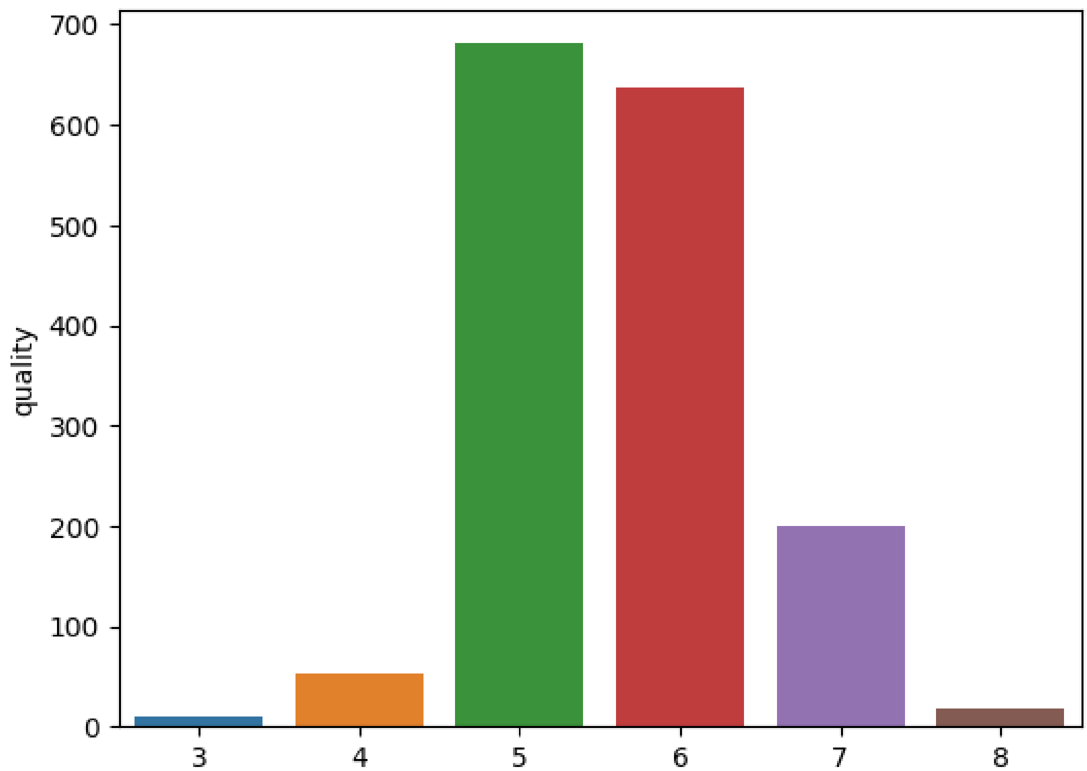
<seaborn.axisgrid.FacetGrid at 0x7c543fb7cdf0>



Univariate Analysis - 4 (Barplot)

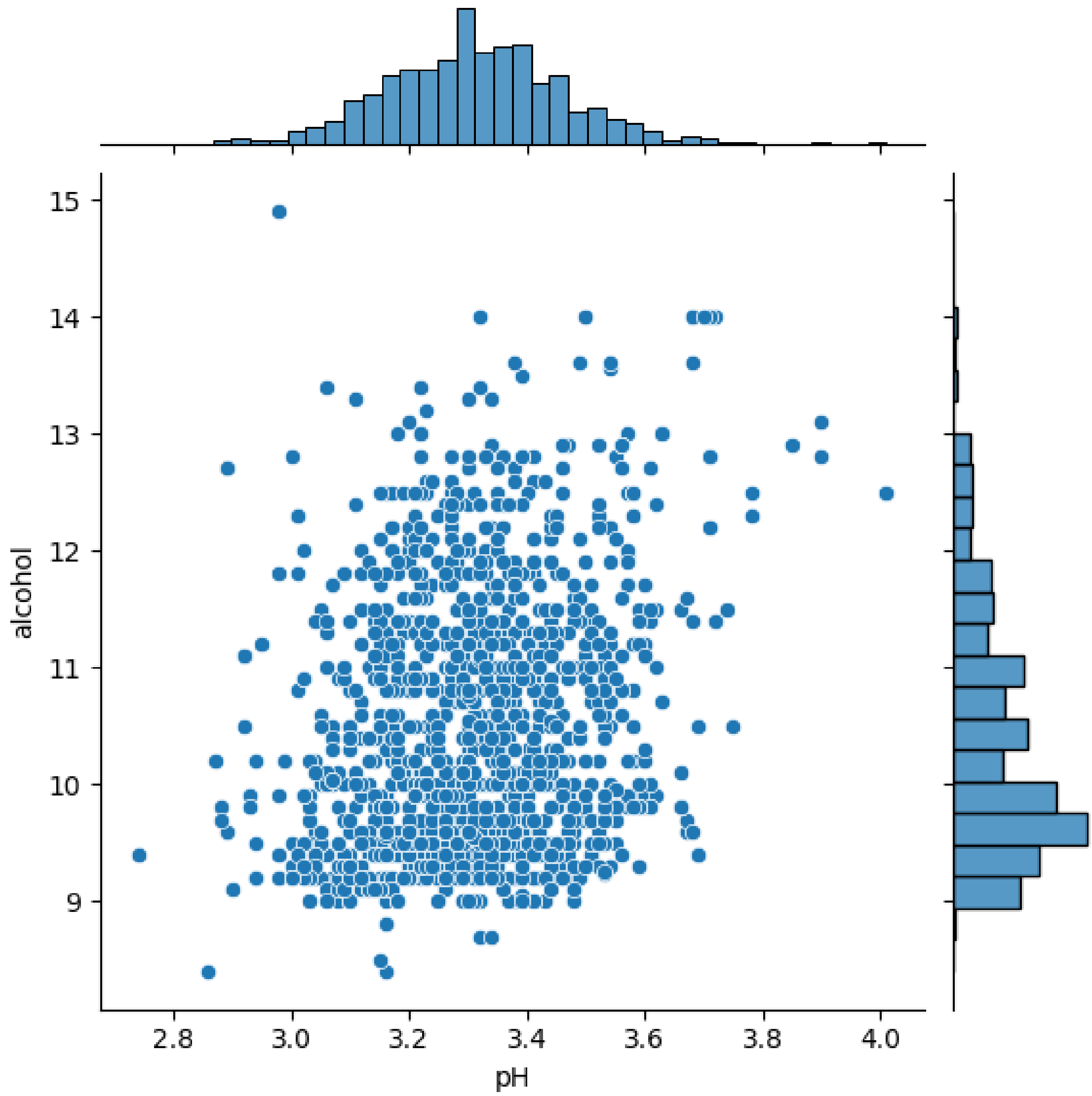
```
sns.barplot(x=df['quality'].value_counts().index,y=df['quality'].value
_counts())
```

```
<Axes: ylabel='quality'>
```

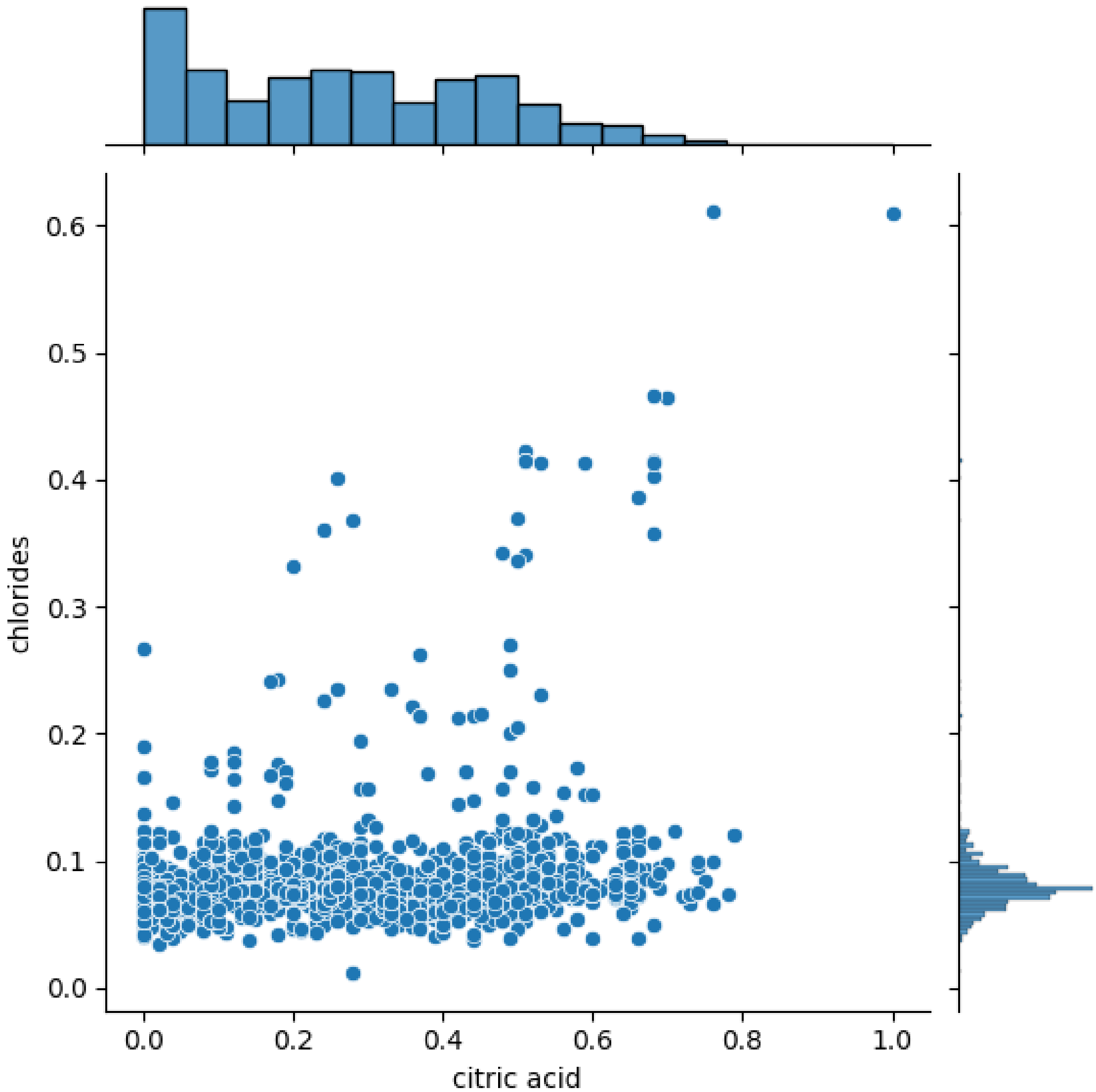


Bivariate Analysis - 1 (jointplot)

```
sns.jointplot(x = df['pH'], y = df['alcohol'], data = df)
plt.show()
```



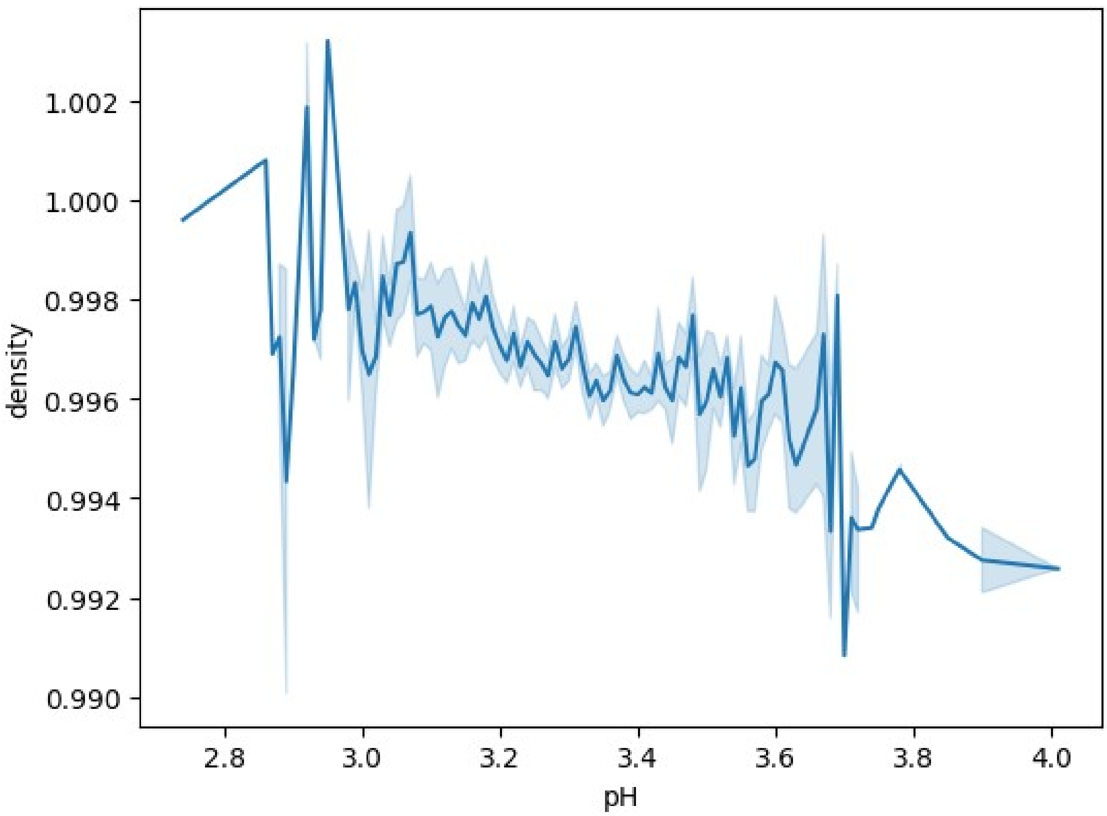

```
sns.jointplot(x = df['citric acid'], y = df['chlorides'], data = df)
plt.show()
```



Bivariate analysis - 2 (lineplot)

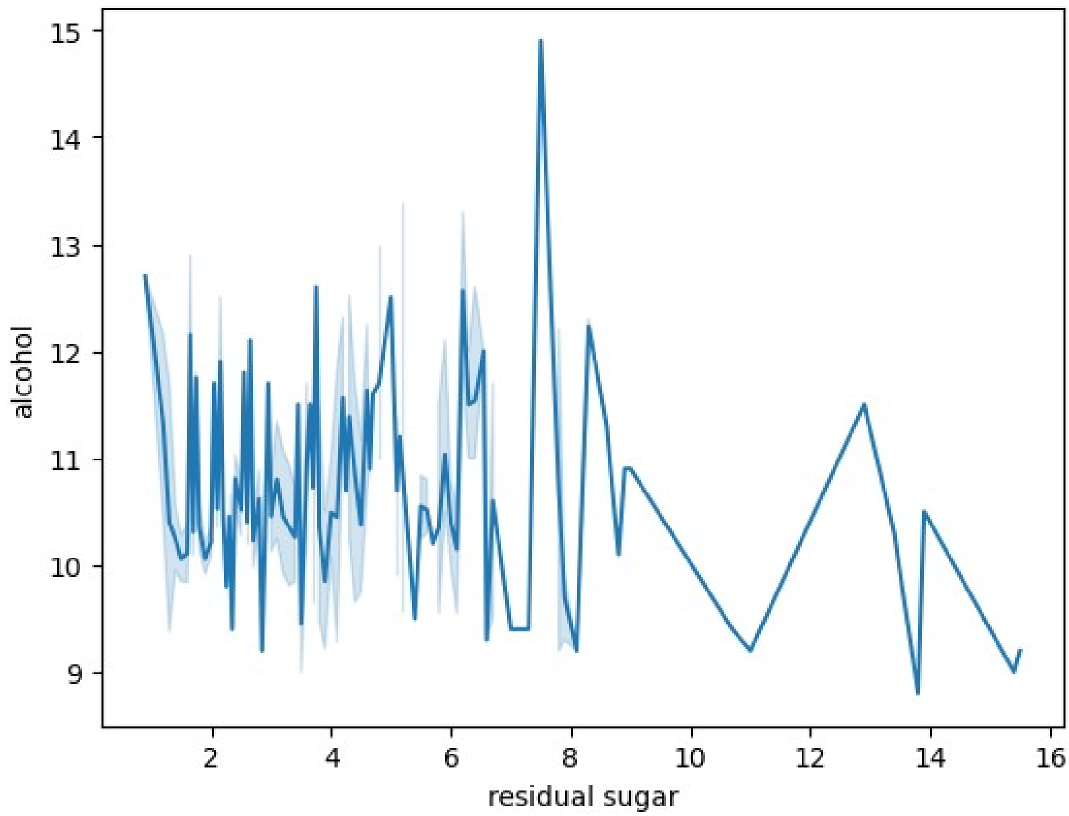
```
sns.lineplot(x=df['pH'],y=df['density'])
```

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




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

```
<Axes: xlabel='residual sugar', ylabel='alcohol'>
```

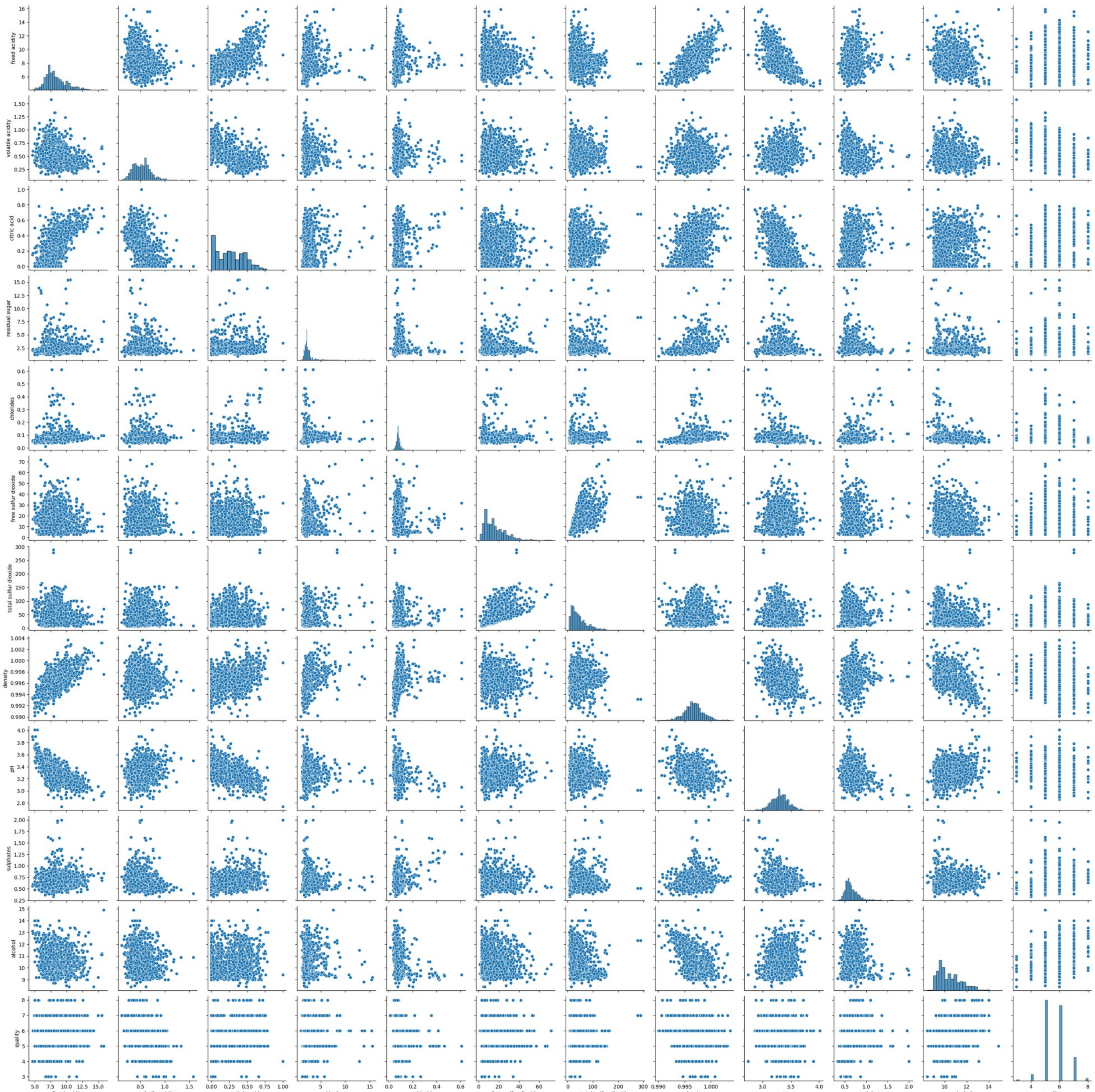


Multivariate Analysis - 1 (pairplot)

```
plt.figure(figsize=(20,20))
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7c543f4b0af0>
```

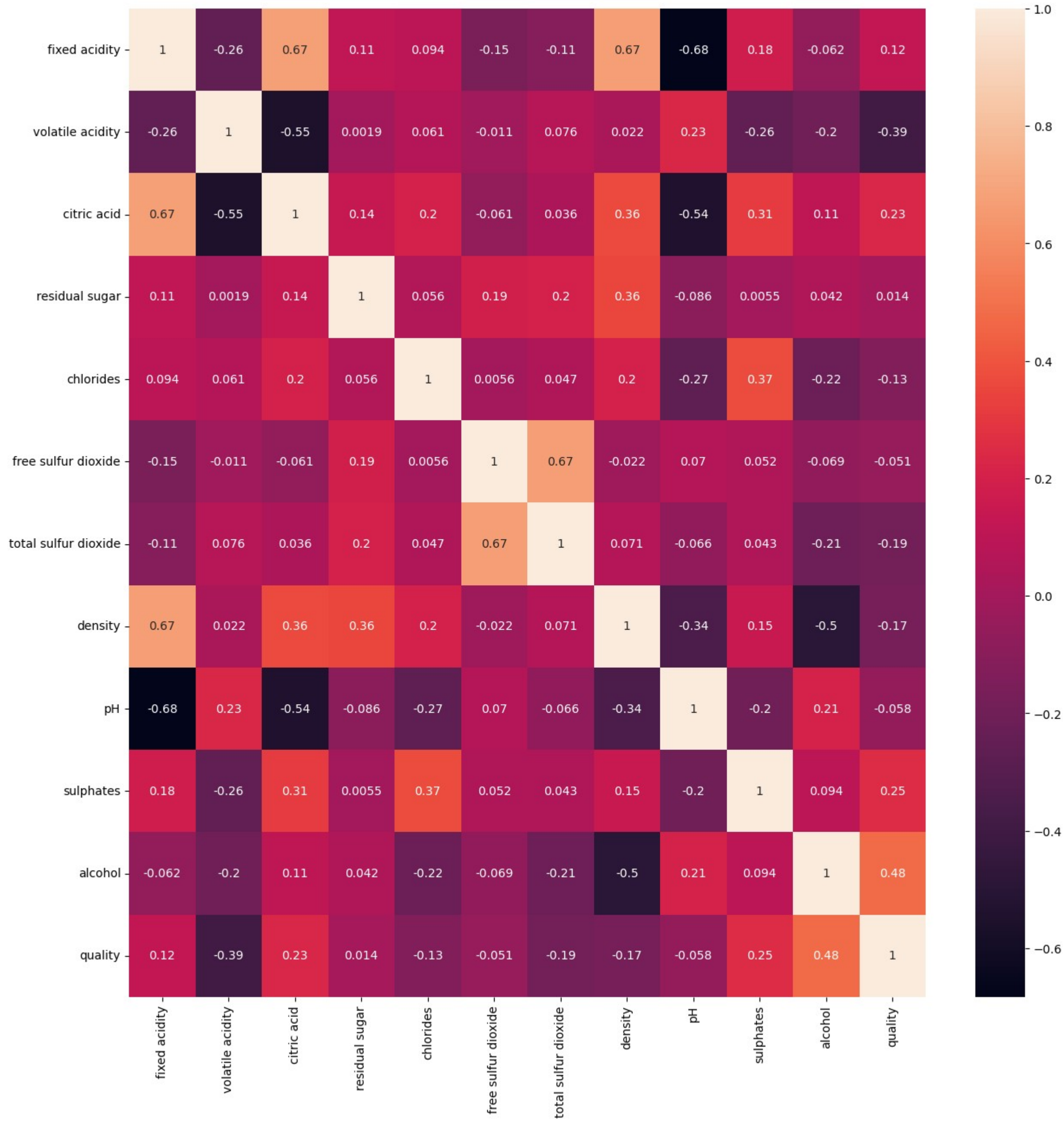
```
<Figure size 2000x2000 with 0 Axes>
```



Multivariate Analysis - 2 (Heatmap)

```
plt.figure(figsize=(15,15))
sns.heatmap(df.corr(),annot=True)
```

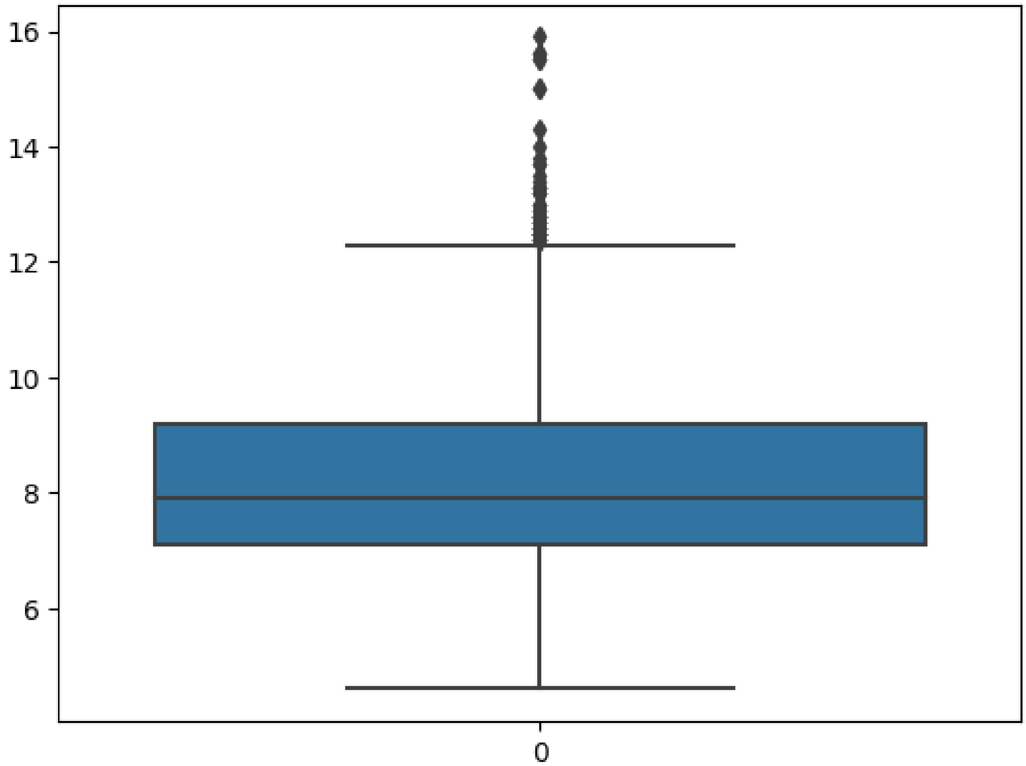
```
<Axes: >
```



Outlier Detection and Replacement

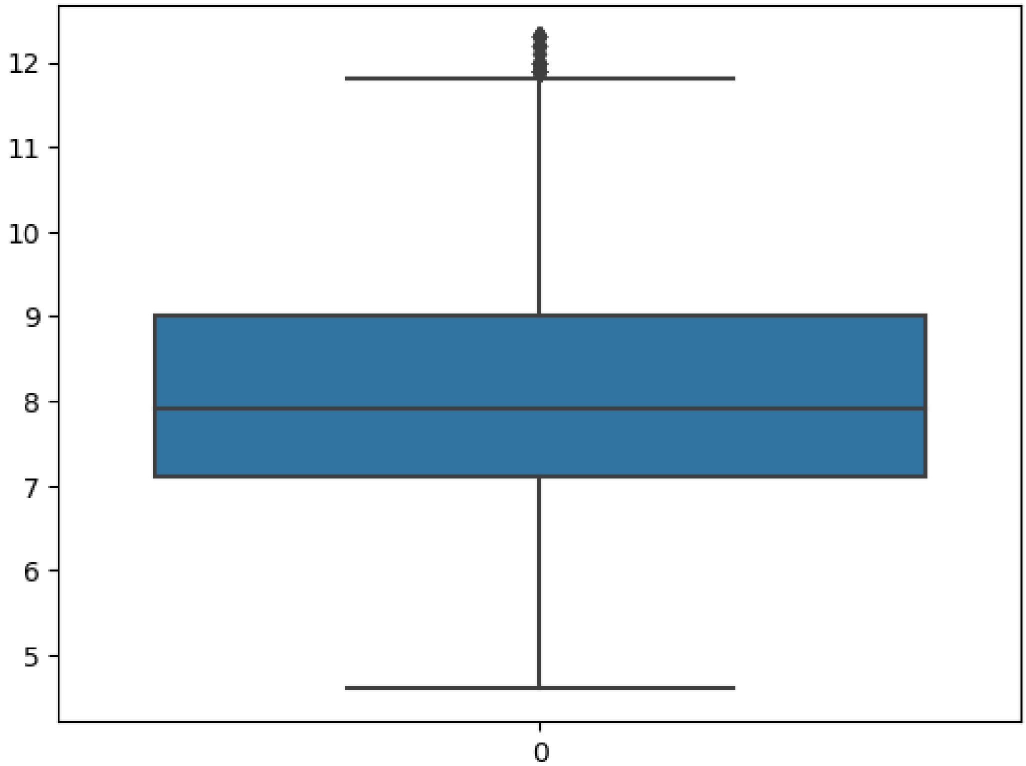
```
sns.boxplot(df['fixed acidity'])
```

```
<Axes: >
```



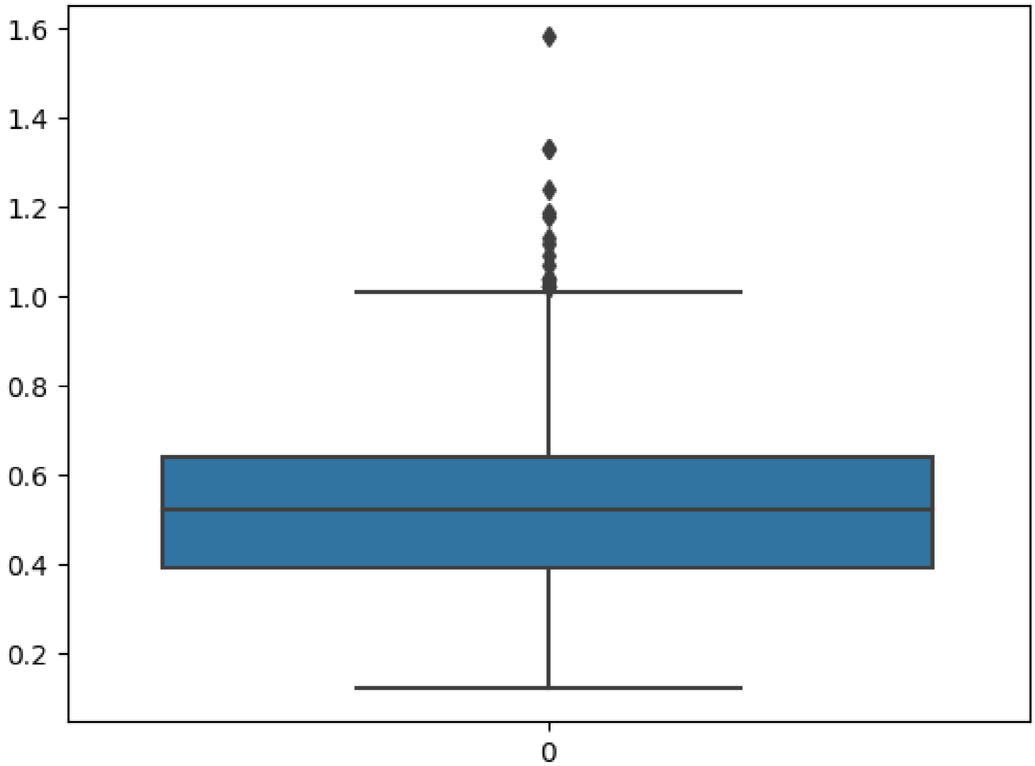

```
q1=df['fixed acidity'].quantile(0.25)
q3=df['fixed acidity'].quantile(0.75)
IQR = q3-q1
upper_limit = q3 + 1.5*IQR
lower_limit = q1 - 1.5*IQR
df['fixed acidity'] = np.where((df['fixed acidity']>upper_limit) |
(df['fixed acidity']<lower_limit),df['fixed
acidity'].median(),df['fixed acidity'])

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

```
<Axes: >
```

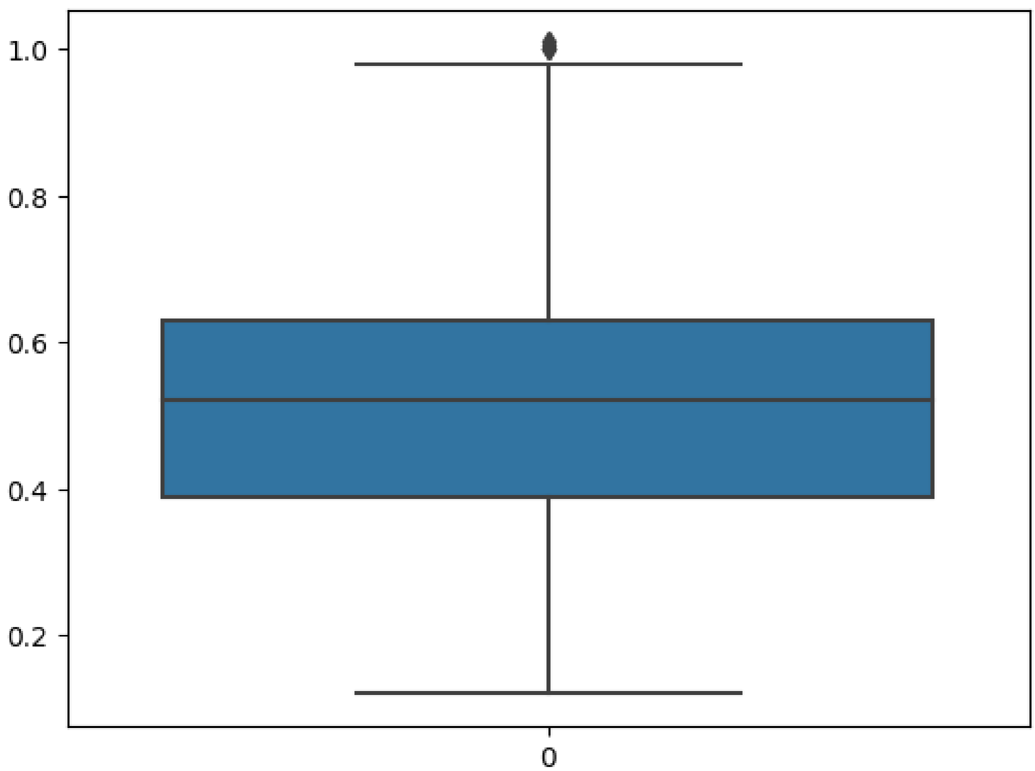



```python
sns.boxplot(df['volatile acidity'])
```

```
<Axes: >
```

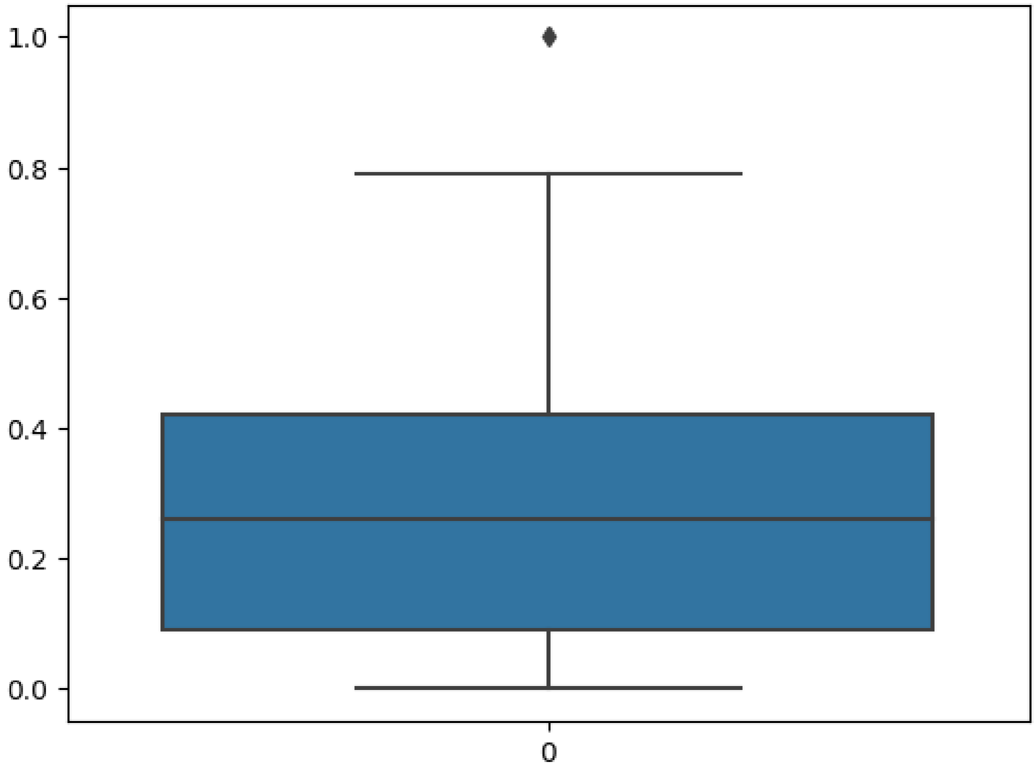



```
q1=df['volatile acidity'].quantile(0.25)
q3=df['volatile acidity'].quantile(0.75)
IQR = q3-q1
upper_limit = q3 + 1.5*IQR
lower_limit = q1 - 1.5*IQR
df['volatile acidity'] = np.where((df['volatile acidity']>upper_limit)
| (df['volatile acidity']<lower_limit),df['volatile
acidity'].median(),df['volatile acidity'])

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

<Axes: >
```



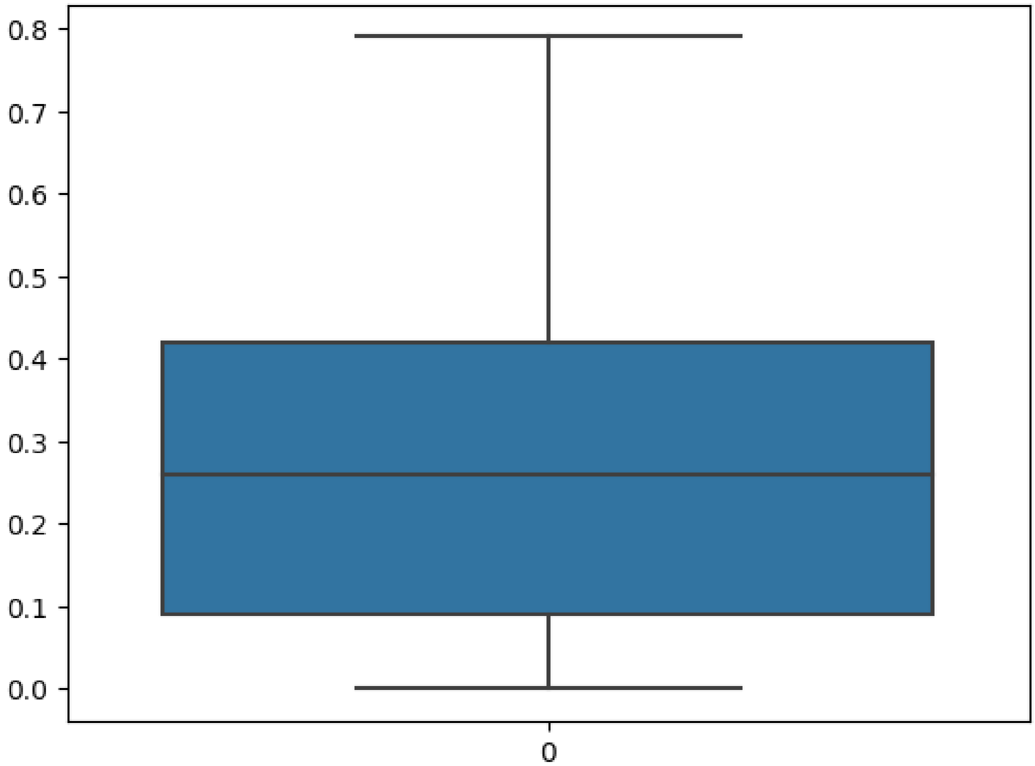

```
sns.boxplot(df['citric acid'])
```

```
<Axes: >
```



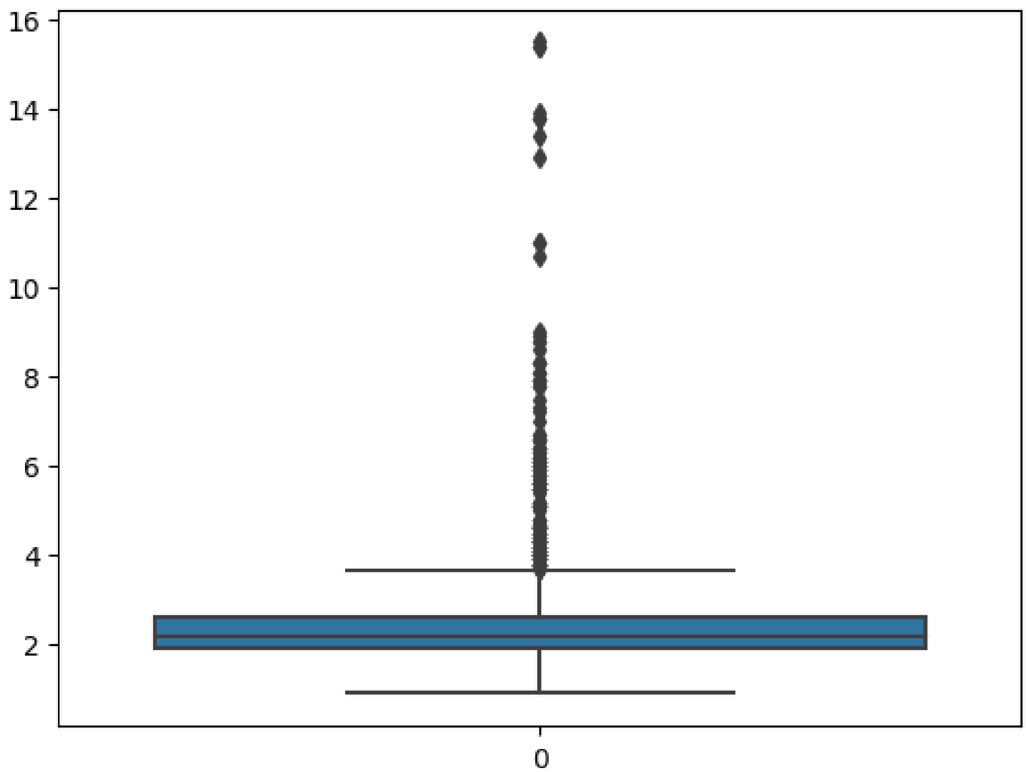

```
q1=df['citric acid'].quantile(0.25)
q3=df['citric acid'].quantile(0.75)
IQR = q3-q1
upper_limit = q3 + 1.5*IQR
lower_limit = q1 - 1.5*IQR
df['citric acid'] = np.where((df['citric acid']>upper_limit) |
(df['citric acid']<lower_limit),df['citric acid'].median(),df['citric
acid'])

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

```
<Axes: >
```

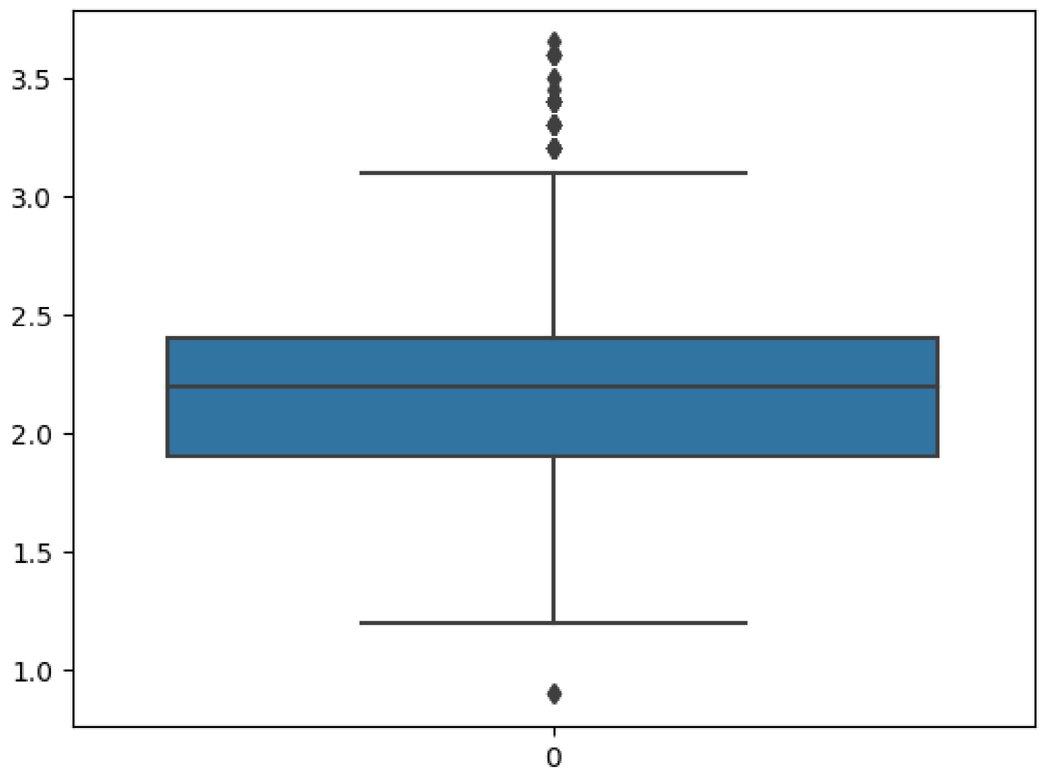



```
sns.boxplot(df['residual sugar'])
```

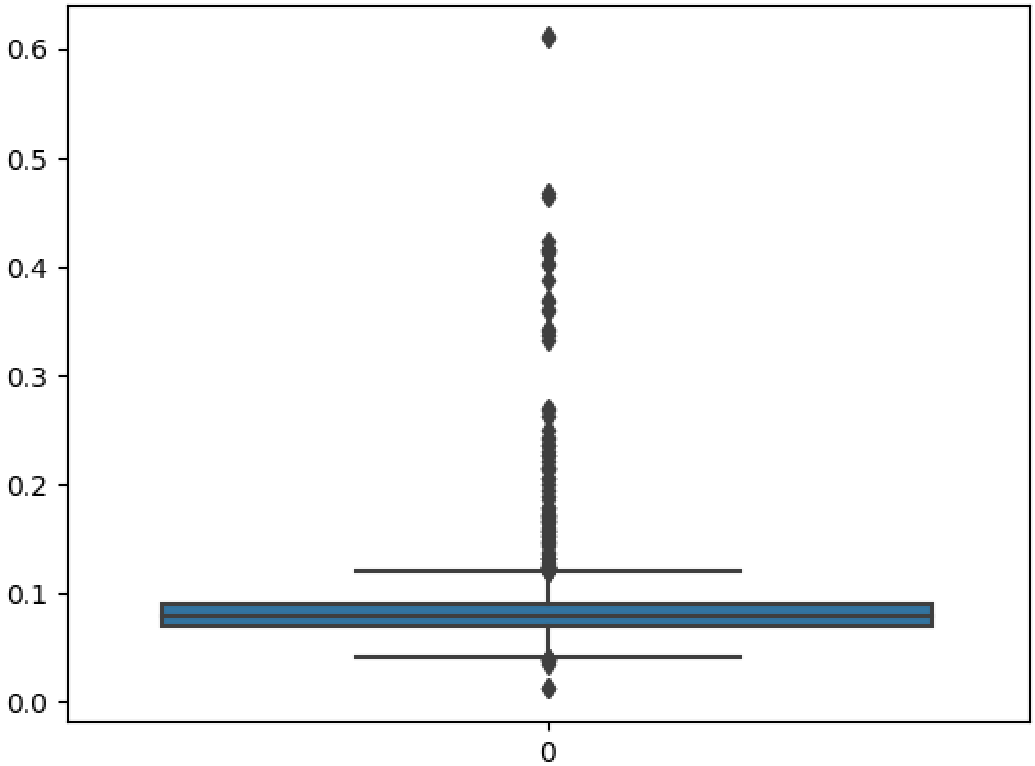
```
<Axes: >
```



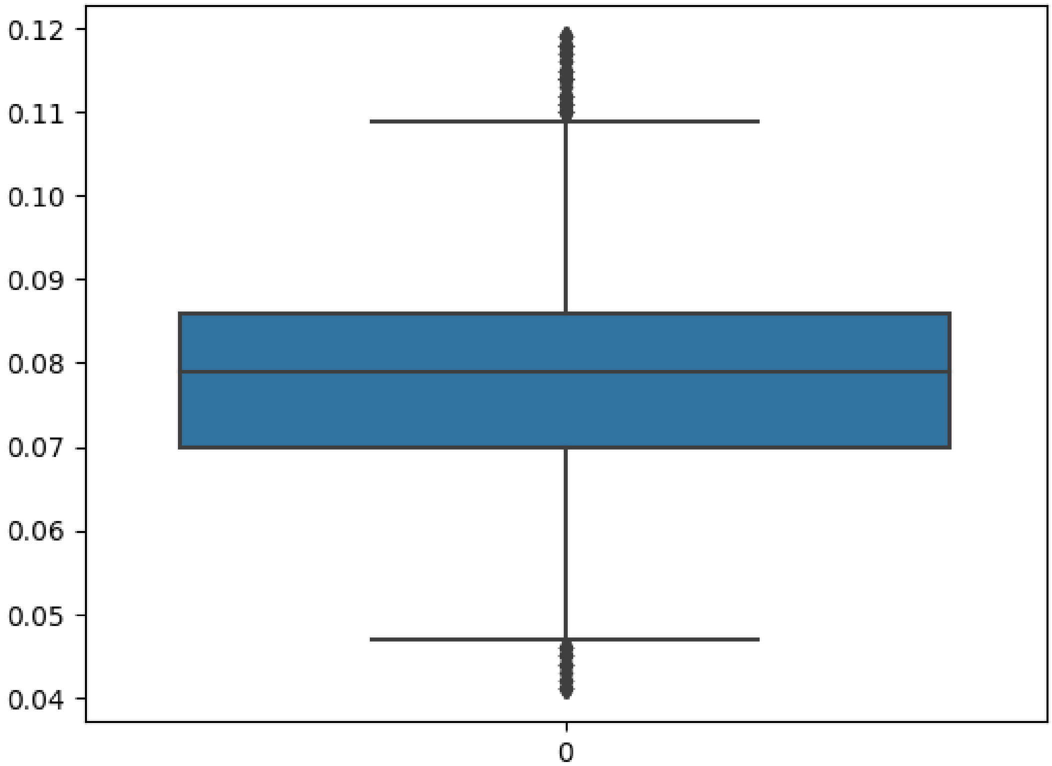

```
q1=df['residual sugar'].quantile(0.25)
q3=df['residual sugar'].quantile(0.75)
IQR = q3-q1
upper_limit = q3 + 1.5*IQR
lower_limit = q1 - 1.5*IQR
df['residual sugar'] = np.where((df['residual sugar']>upper_limit) |
(df['residual sugar']<lower_limit),df['residual
sugar'].median(),df['residual sugar'])

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

<Axes: >
```




```
sns.boxplot(df['chlorides'])
<Axes: >
```

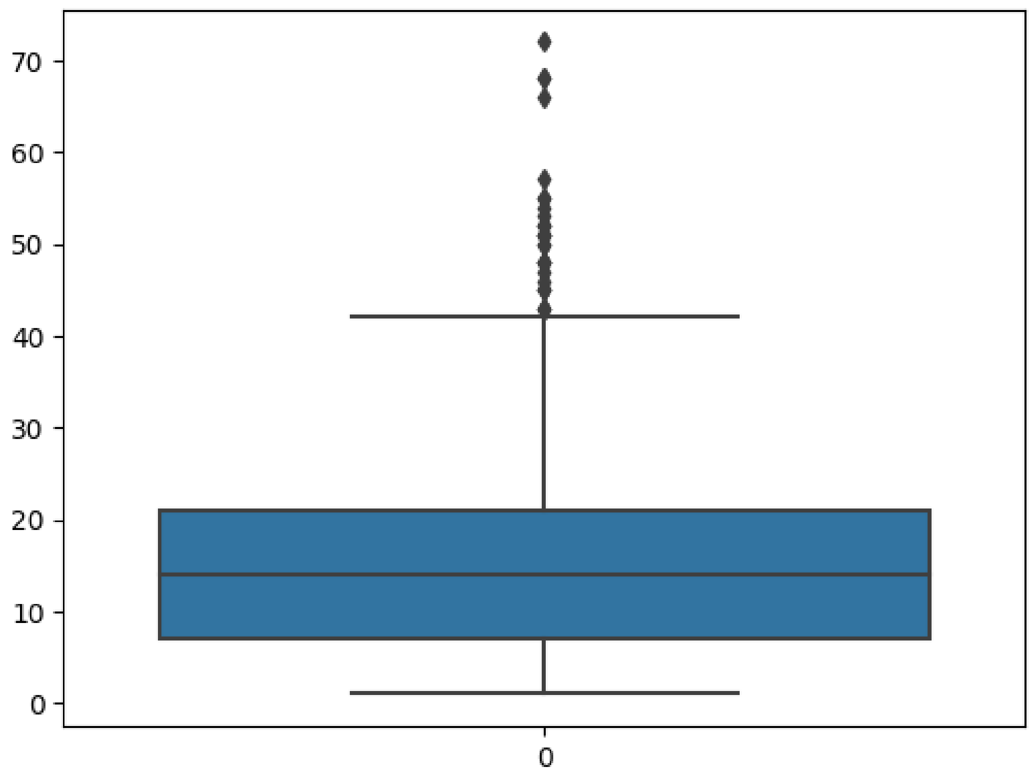



```
q1=df['chlorides'].quantile(0.25)
q3=df['chlorides'].quantile(0.75)
IQR = q3-q1
upper_limit = q3 + 1.5*IQR
lower_limit = q1 - 1.5*IQR
df['chlorides'] = np.where((df['chlorides']>upper_limit) |
(df['chlorides']<lower_limit),df['chlorides'].median(),df['chlorides']
)

sns.boxplot(df['chlorides'])

<Axes: >
```



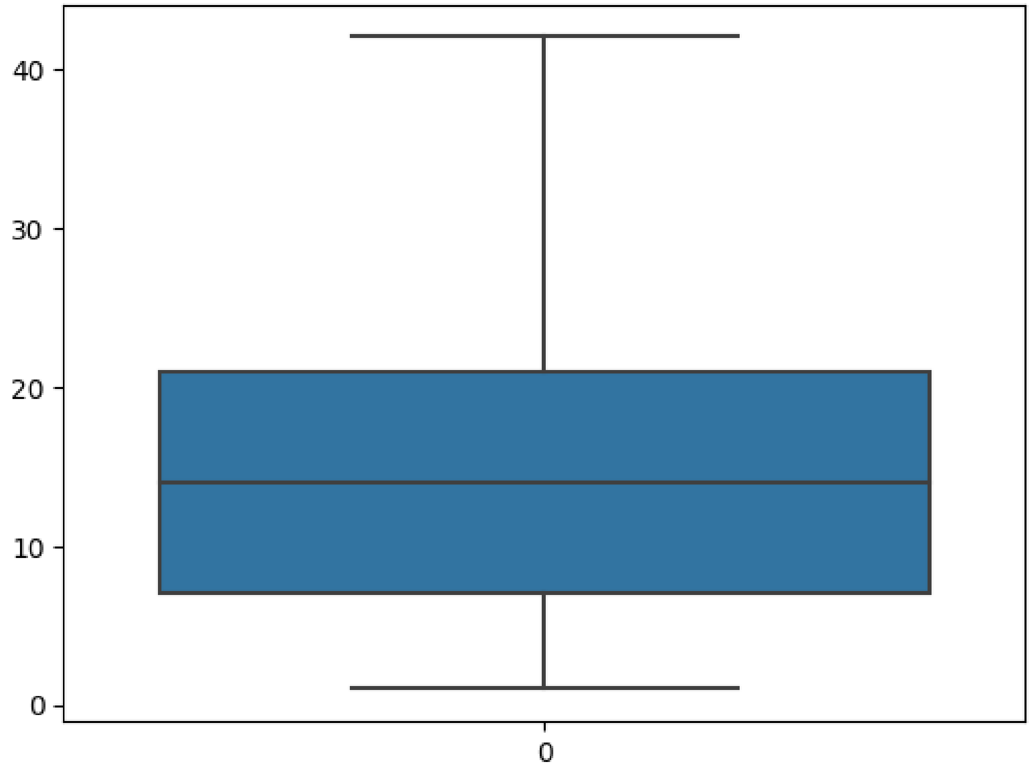

```
sns.boxplot(df['free sulfur dioxide'])
```

```
<Axes: >
```



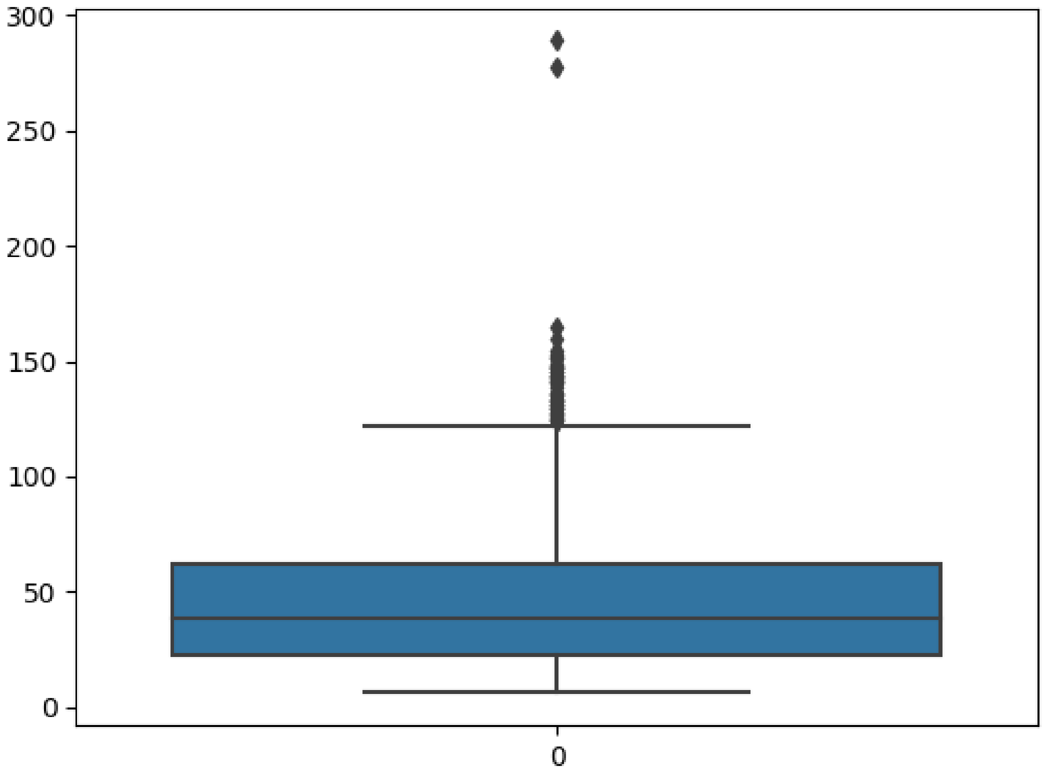

```
q1=df['free sulfur dioxide'].quantile(0.25)
q3=df['free sulfur dioxide'].quantile(0.75)
IQR = q3-q1
upper_limit = q3 + 1.5*IQR
lower_limit = q1 - 1.5*IQR
df['free sulfur dioxide'] = np.where((df['free sulfur 
dioxide']>upper_limit) | (df['free sulfur 
dioxide']<lower_limit),df['free sulfur dioxide'].median(),df['free 
sulfur dioxide'])

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

```
<Axes: >
```

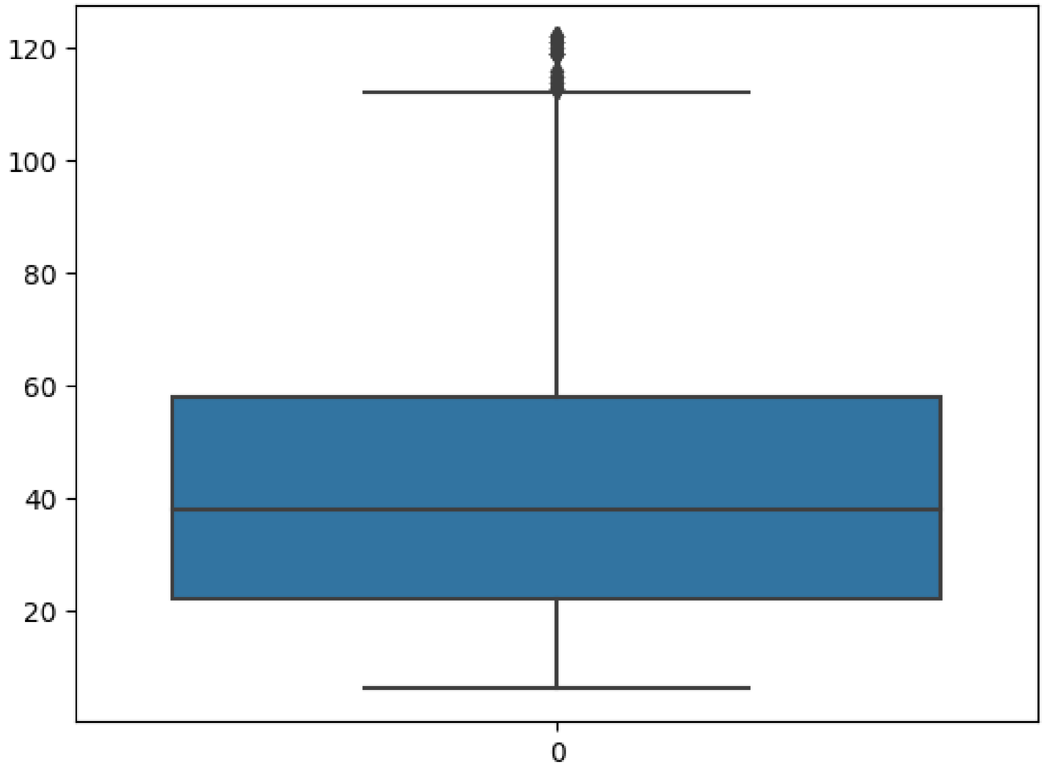



```
sns.boxplot(df['total sulfur dioxide'])
```

```
<Axes: >
```

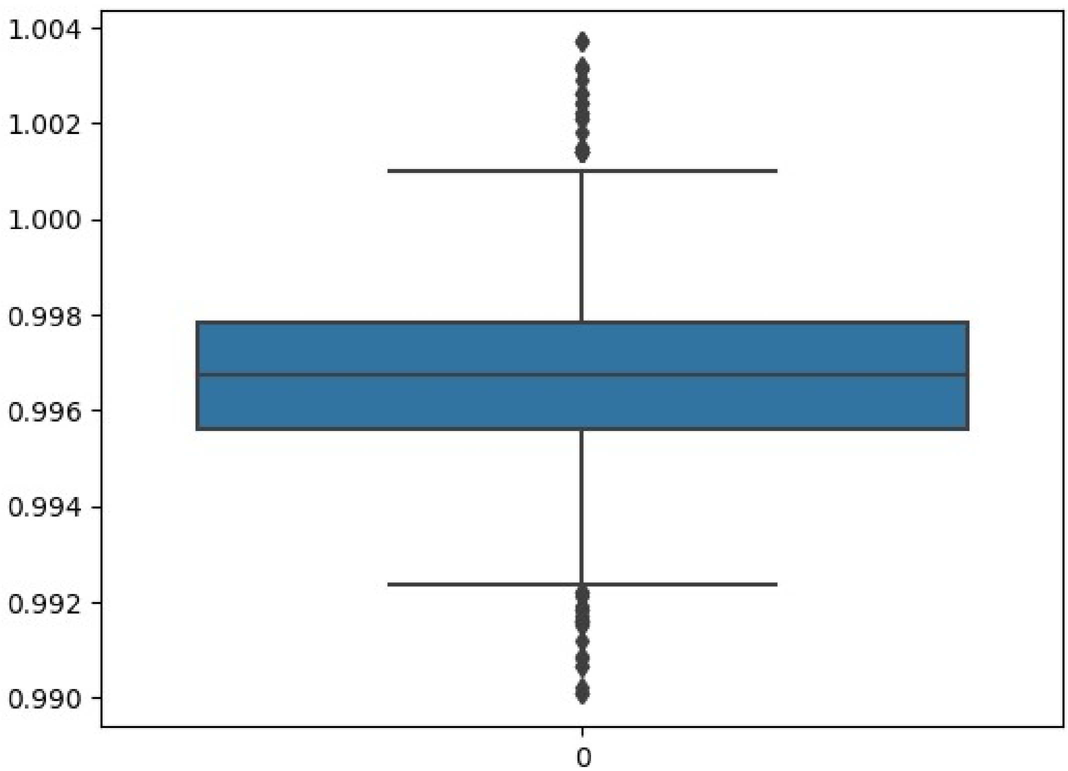



```
q1=df['total sulfur dioxide'].quantile(0.25)
q3=df['total sulfur dioxide'].quantile(0.75)
IQR = q3-q1
upper_limit = q3 + 1.5*IQR
lower_limit = q1 - 1.5*IQR
df['total sulfur dioxide'] = np.where((df['total sulfur
dioxide']>upper_limit) | (df['total sulfur
dioxide']<lower_limit),df['total sulfur dioxide'].median(),df['total
sulfur dioxide'])

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

<Axes: >
```

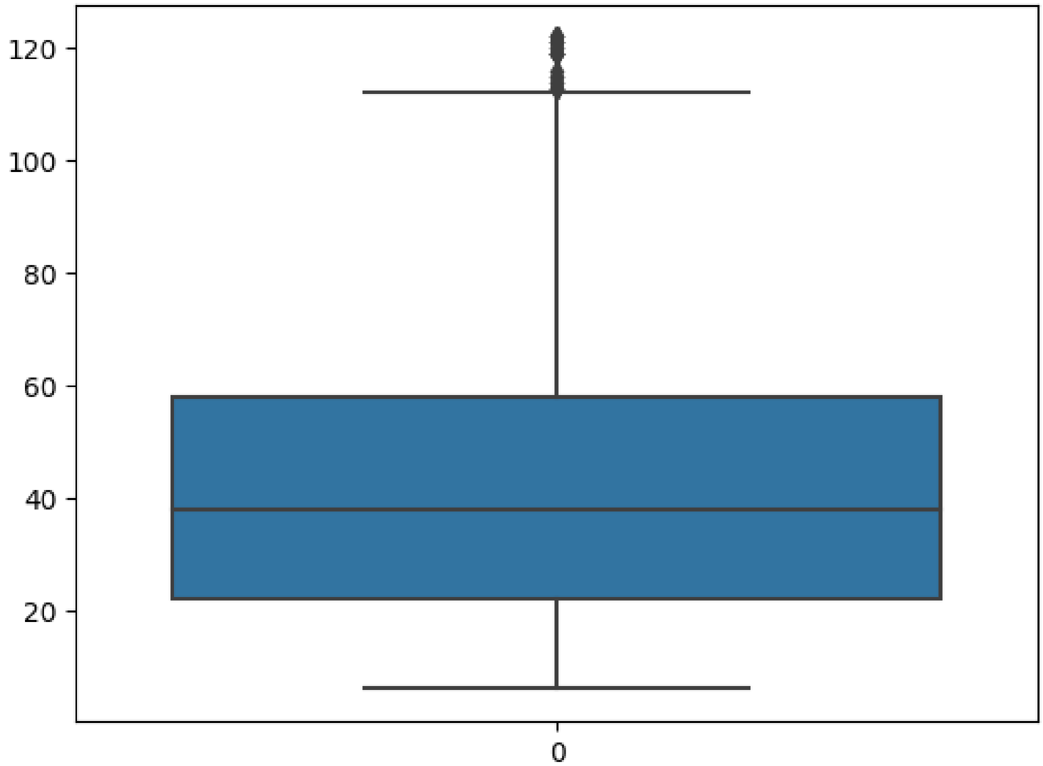



```
sns.boxplot(df['density'])
```

```
<Axes: >
```

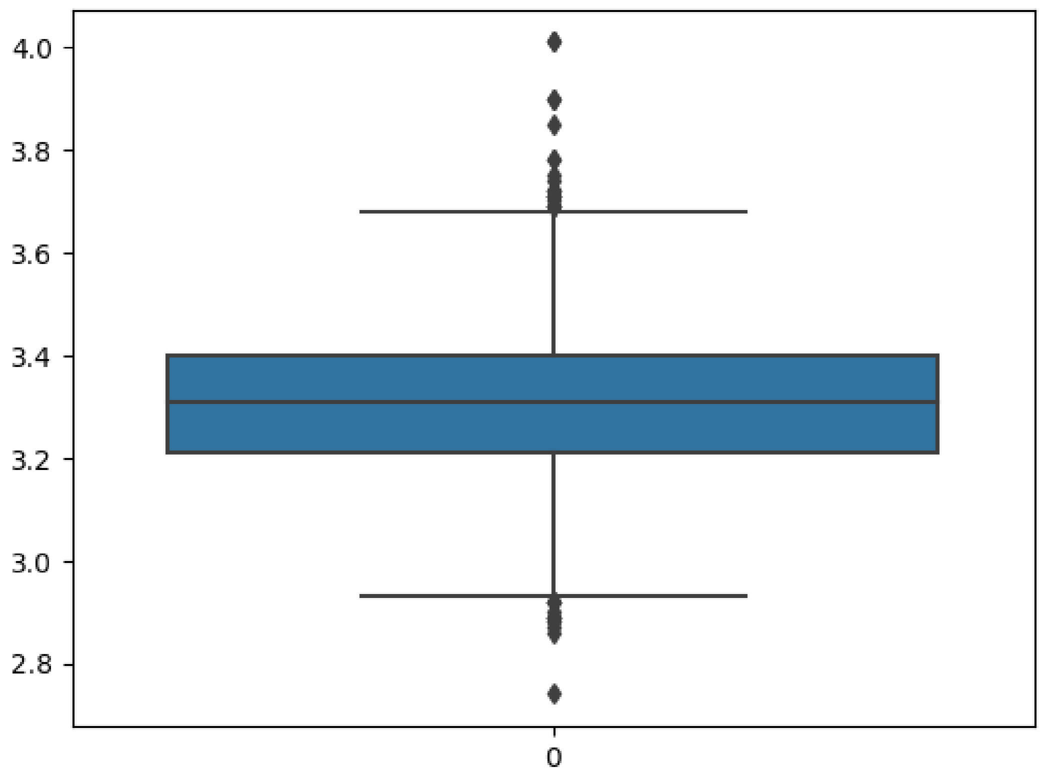



```
q1=df['density'].quantile(0.25)
q3=df['density'].quantile(0.75)
IQR = q3-q1
upper_limit = q3 + 1.5*IQR
lower_limit = q1 - 1.5*IQR
df['density'] = np.where((df['density']>upper_limit) |
(df['density']<lower_limit),df['density'].median(),df['density'])

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

<Axes: >
```



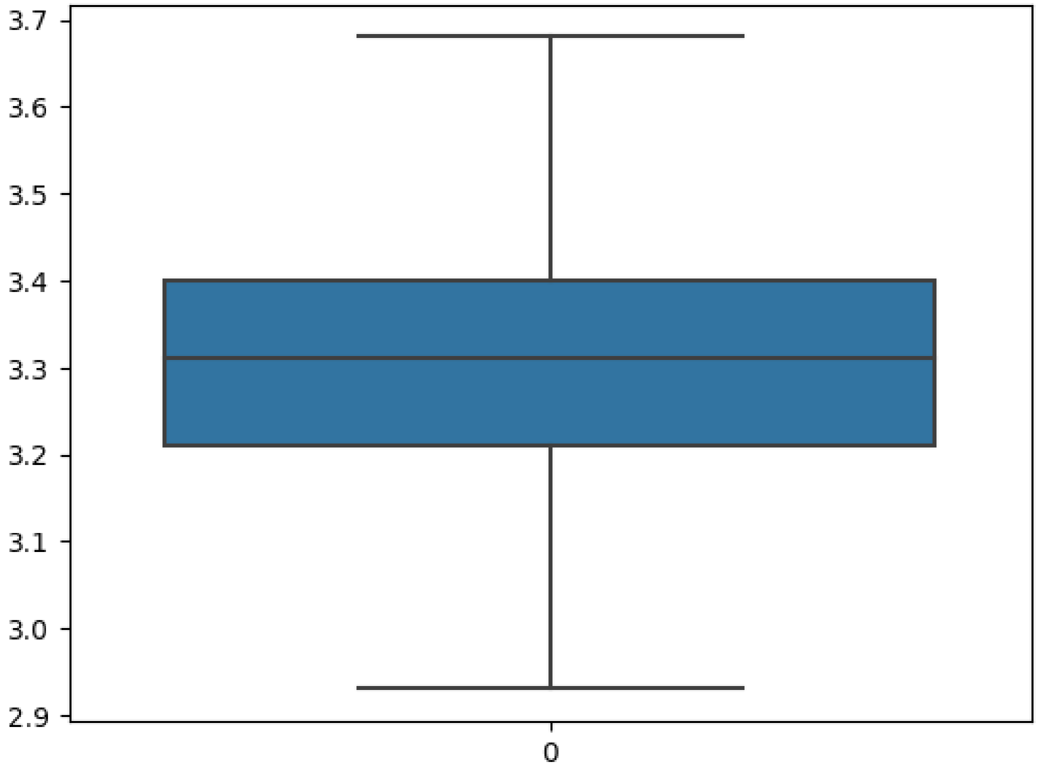

```
sns.boxplot(df['pH'])
```

```
<Axes: >
```



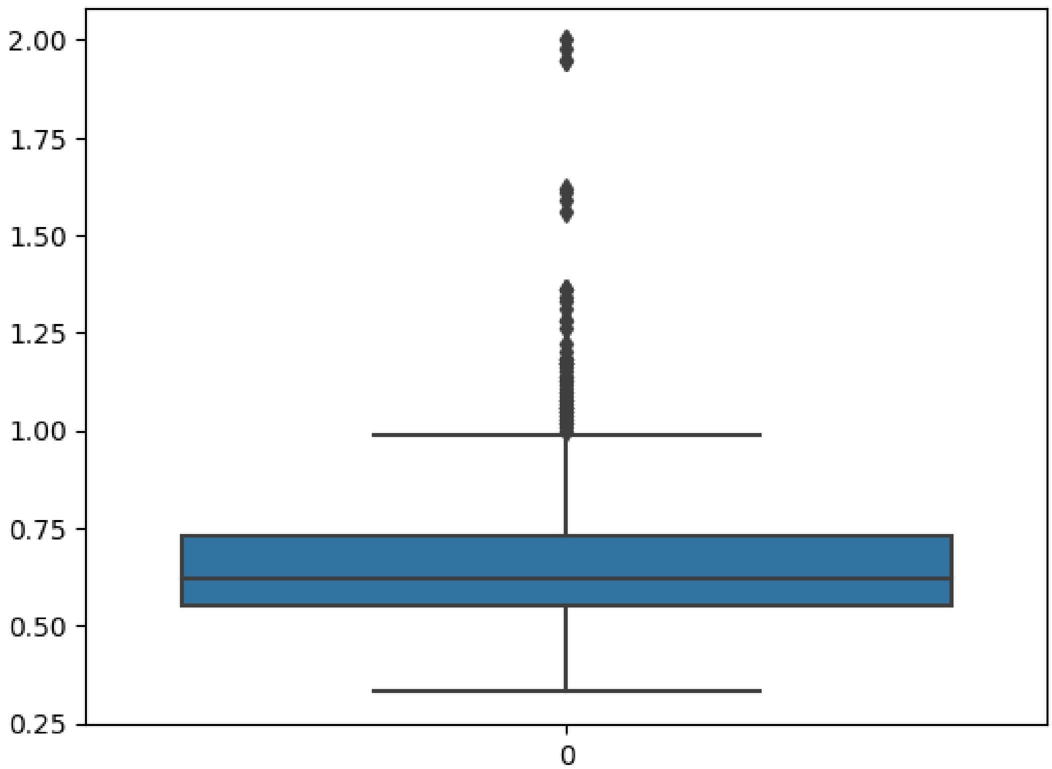

```
q1=df['pH'].quantile(0.25)
q3=df['pH'].quantile(0.75)
IQR = q3-q1
upper_limit = q3 + 1.5*IQR
lower_limit = q1 - 1.5*IQR
df['pH'] = np.where((df['pH']>upper_limit) |
(df['pH']<lower_limit),df['pH'].median(),df['pH'])

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

```
<Axes: >
```

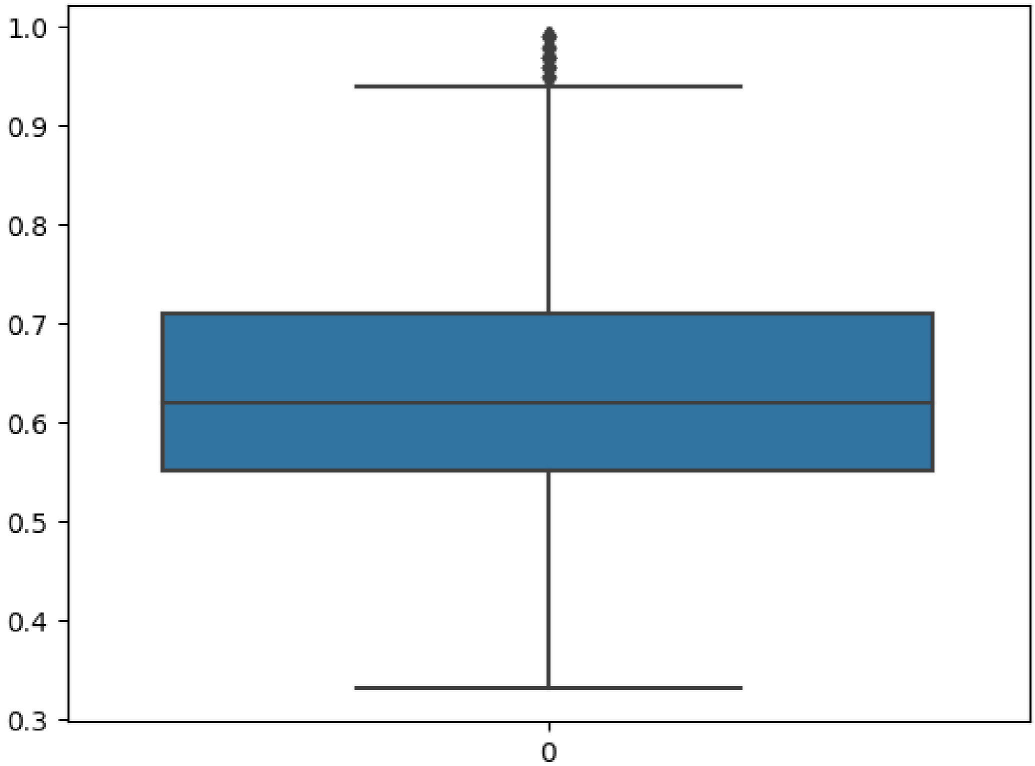



```
sns.boxplot(df['sulphates'])
```

```
<Axes: >
```

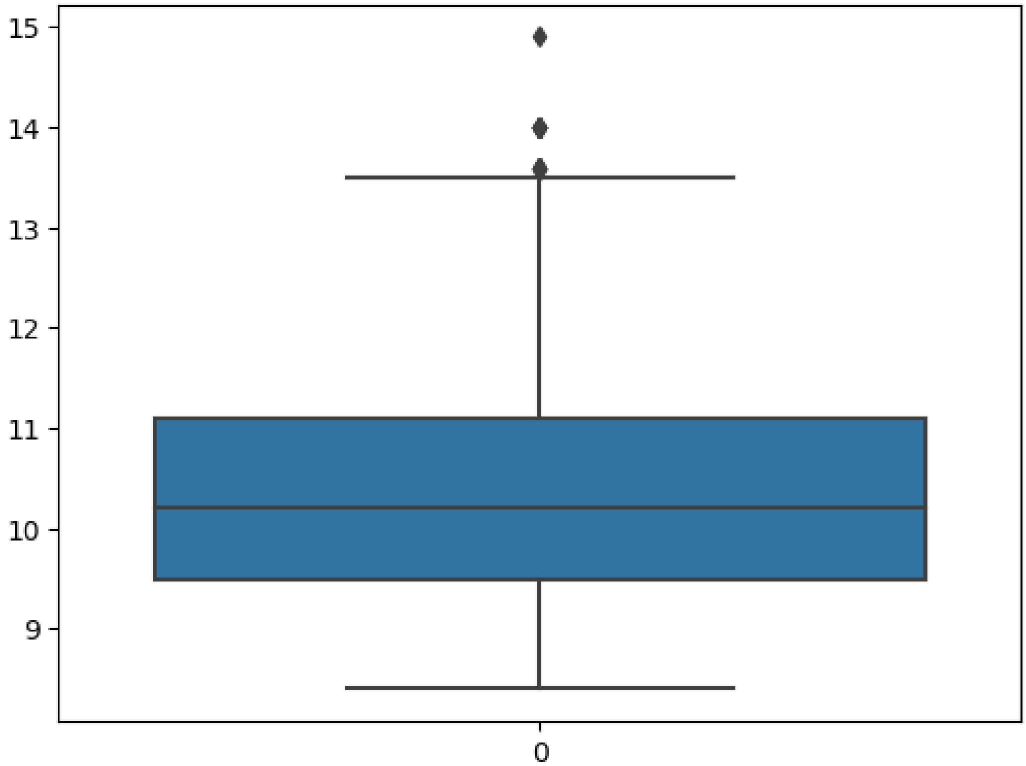



```
q1=df['sulphates'].quantile(0.25)
q3=df['sulphates'].quantile(0.75)
IQR = q3-q1
upper_limit = q3 + 1.5*IQR
lower_limit = q1 - 1.5*IQR
df['sulphates'] = np.where((df['sulphates']>upper_limit) |
(df['sulphates']<lower_limit),df['sulphates'].median(),df['sulphates']
)

sns.boxplot(df['sulphates'])

<Axes: >
```

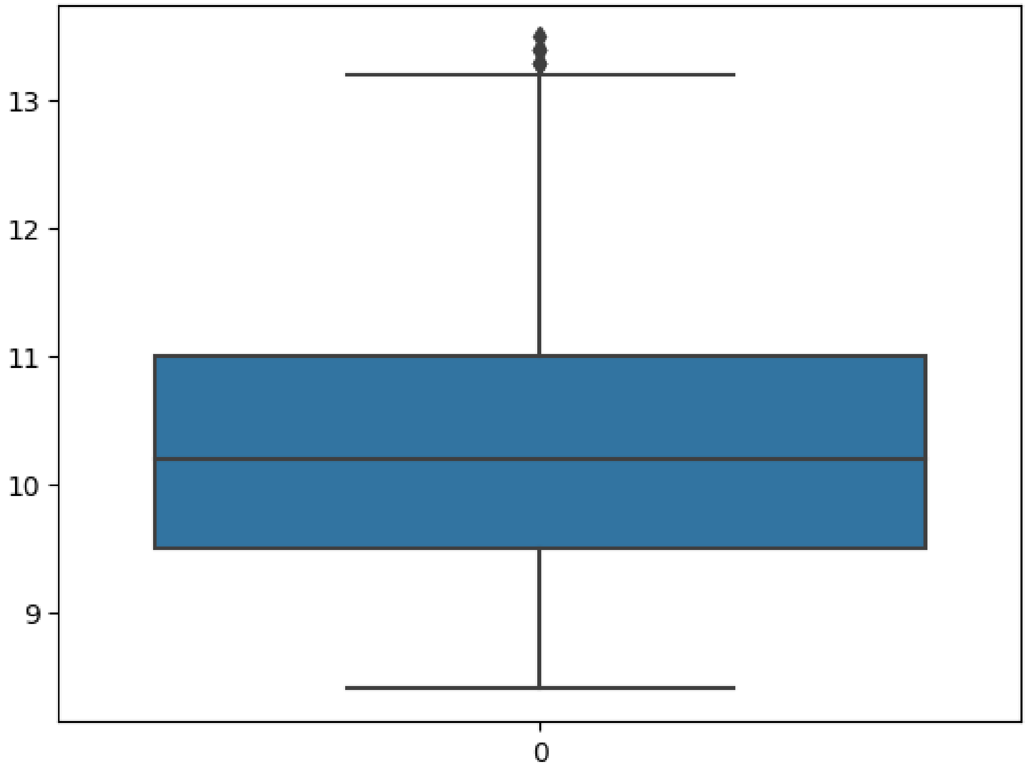



```
sns.boxplot(df['alcohol'])
```

```
<Axes: >
```

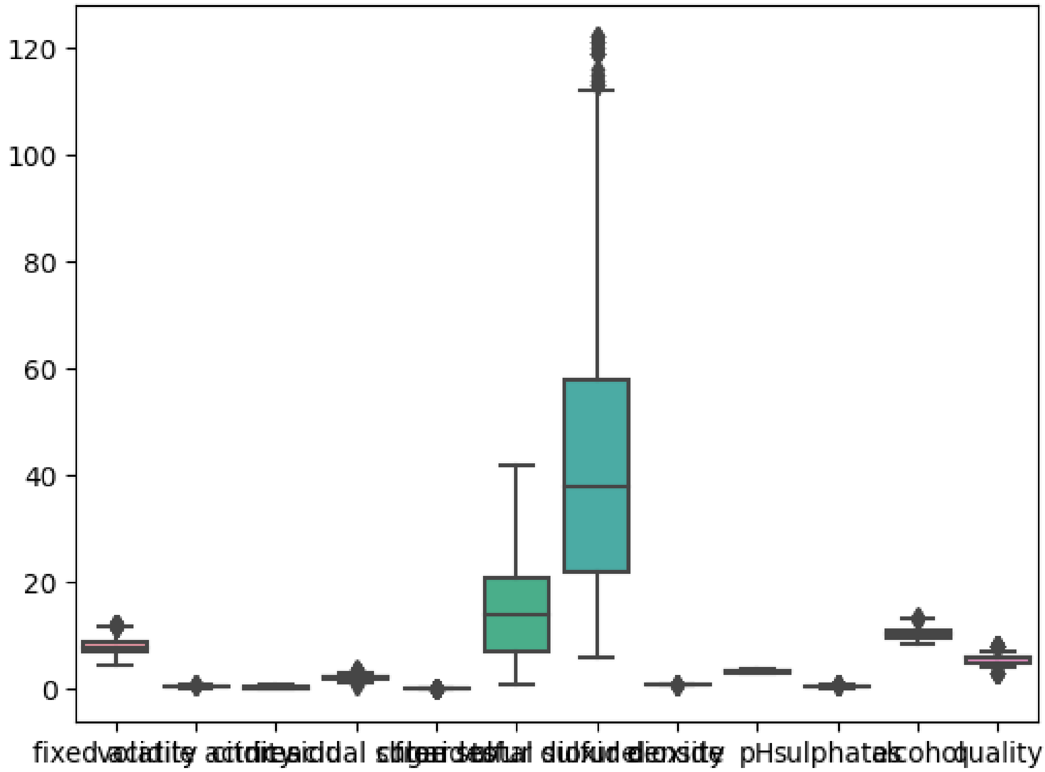



```
q1=df['alcohol'].quantile(0.25)
q3=df['alcohol'].quantile(0.75)
IQR = q3-q1
upper_limit = q3 + 1.5*IQR
lower_limit = q1 - 1.5*IQR
df['alcohol'] = np.where((df['alcohol']>upper_limit) |
(df['alcohol']<lower_limit),df['alcohol'].median(),df['alcohol'])

sns.boxplot(df['alcohol'])

<Axes: >
```




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

```
<Axes: >
```



Splitting the data into features and target

```
X=df.drop(columns=['quality'],axis=1)
X.head(10)
```

```
   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
5            7.4              0.66         0.00             1.8
0.075
6            7.9              0.60         0.06             1.6
0.069
7            7.3              0.65         0.00             1.2
0.065
8            7.8              0.58         0.02             2.0
0.073
9            7.5              0.50         0.36             2.2
```

```
0.071

   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

5                 13.0                  40.0   0.9978  3.51       0.56

6                 15.0                  59.0   0.9964  3.30       0.46

7                 15.0                  21.0   0.9946  3.39       0.47

8                  9.0                  18.0   0.9968  3.36       0.57

9                 17.0                 102.0   0.9978  3.35       0.80


   alcohol
0      9.4
1      9.8
2      9.8
3      9.8
4      9.4
5      9.4
6      9.4
7     10.0
8      9.5
9     10.5
```

```
y=df['quality']
y.head(10)
```

```
0    5
1    5
2    5
3    6
4    5
5    5
6    5
7    7
8    7
9    5
Name: quality, dtype: int64
```

Balancing the dataset

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(sampling_strategy='minority', random_state=42)
X_resampled, Y_resampled = sm.fit_resample(X,y)

X_resampled.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 the data

```
from sklearn.preprocessing import StandardScaler

scale=StandardScaler()

X_scaled=pd.DataFrame(scale.fit_transform(X_resampled),columns=X_resam
pled.columns)
X_scaled.head(10)
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  
chlorides  \
0      -0.557461          0.836418    -1.201523       -0.621499  -
0.256246
1      -0.280978          1.915642    -1.201523        1.020535   
1.484148
2      -0.280978          1.196159    -1.001027        0.316806   
1.009495
3       2.069129         -1.681771     1.605421       -0.621499  -
0.335355
4      -0.557461          0.836418    -1.201523       -0.621499  -
0.256246
5      -0.557461          0.596591    -1.201523       -0.856075  -
0.335355
6      -0.211857          0.236849    -0.900779       -1.325228  -
0.810008
7      -0.626582          0.536634    -1.201523       -2.263533  -
1.126444
8      -0.280978          0.116936    -1.101275       -0.386923  -
0.493573
9      -0.488340         -0.362719     0.602941        0.082230  -
0.651791

   free sulfur dioxide  total sulfur dioxide   density        pH  
sulphates  \
0            -0.273094             -0.109066  0.512630  1.225184  -
0.469038
1             1.286876              1.187198 -0.099961 -0.969023   
0.577948
2             0.172612              0.676548  0.022557 -0.544338   
0.316202
3             0.395465              0.912233  0.635149 -1.252147  -
0.294540
4            -0.273094             -0.109066  0.512630  1.225184  -
0.469038
5            -0.050241              0.126618  0.512630  1.225184  -
0.469038
6             0.172612              0.872952 -0.344998 -0.261215  -
1.341527
7             0.172612             -0.619716 -1.447662  0.375813  -
1.254278
8            -0.495946             -0.737558 -0.099961  0.163471  -
0.381789
9             0.395465              2.562023  0.512630  0.092690   
1.624935

    alcohol  
0 -0.907428  
1 -0.487303  
2 -0.487303  
```

```
3 -0.487303
4 -0.907428
5 -0.907428
6 -0.907428
7 -0.277241
8 -0.802397
9  0.247916
```

Train-Test-Split for Model - 1

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test =
train_test_split(X_scaled,Y_resampled,test_size=0.2,random_state=42)

X_train.shape

(1816, 11)

X_test.shape

(454, 11)

Y_train.shape

(1816,)

Y_test.shape

(454,)
```

Model Training - 1 (Linear Regression)

```
from sklearn.linear_model import LinearRegression

LinReg=LinearRegression()

LinReg.fit(X_train,Y_train)

LinearRegression()

Y_pred_1=LinReg.predict(X_test)
Y_pred_1

array([5.02726067, 4.09327914, 3.79222271, 5.7697455 , 4.95621419,
       4.8764308 , 6.7669688 , 5.18800074, 5.1286554 , 3.27335691,
       6.05434038, 3.81316565, 3.55631923, 5.86933722, 3.75363426,
       3.73563697, 3.80419401, 5.29750095, 5.39782959, 6.27465925,
       4.7976689 , 6.5291879 , 3.90050452, 4.97788385, 4.177395  ,
       5.56906896, 3.70738333, 4.65386128, 3.7464695 , 4.14824489,
       4.51757566, 4.43666482, 3.81562677, 4.51079308, 3.88368174,
       3.86106089, 6.03815831, 4.06684654, 3.61120231, 3.92721958,
```

```
       5.41459307, 3.81674754, 3.8164644 , 4.87630756, 4.3384424 ,
       4.73061212, 5.67852234, 4.72985685, 4.55603596, 3.92405506,
       3.97830868, 3.61524614, 3.60008007, 3.75180924, 6.6971253 ,
       5.83104316, 6.55932802, 5.68590963, 6.79684662, 5.13319146,
       5.46668465, 5.12630187, 4.42514098, 4.78776279, 5.43738783,
       6.85901268, 4.51079308, 4.74550129, 5.54649335, 6.16065682,
       3.59461948, 5.81685712, 5.62216515, 3.83692923, 3.69687779,
       4.82304888, 4.05667905, 3.58448767, 3.63528992, 4.66393196,
       5.73889251, 3.72510798, 4.19389034, 4.09114426, 3.68533006,
       5.19550969, 4.69196058, 3.87548034, 3.78295838, 6.10899617,
       5.87228031, 3.8798717 , 3.98242027, 4.02749154, 4.69377462,
       3.60319833, 3.44475546, 4.98805155, 4.77405897, 3.55552211,
       3.53287027, 5.14635516, 4.75351337, 5.80816376, 3.86411723,
       6.00914029, 4.75396939, 5.94496939, 5.20872152, 4.80636412,
       3.92228059, 5.28507298, 3.70934417, 4.22319824, 4.98088433,
       3.90198056, 6.32907061, 5.43646407, 5.12951664, 4.79871411,
       6.27409071, 4.03847709, 3.90840519, 3.6871529 , 3.34742   ,
       4.41692165, 4.60211259, 3.81580456, 4.52230042, 4.88575474,
       3.89672496, 6.3729031 , 4.73501902, 6.19658263, 3.90282362,
       4.91823954, 3.87651855, 4.78010581, 5.66562563, 4.85313993,
       4.7409428 , 6.85575175, 3.76218288, 5.34894889, 4.0224938 ,
       5.91012637, 3.79441941, 3.86867602, 4.20022974, 3.65578812,
       5.08471643, 4.84400533, 3.6374376 , 4.42360726, 5.58996291,
       3.90588293, 7.04906305, 6.14317544, 3.91354081, 4.92498657,
       3.85815981, 3.63954782, 3.90824917, 6.28717498, 4.77613682,
       6.09006275, 3.99257813, 5.45567049, 4.09419189, 4.14807601,
       6.20877588, 5.21735984, 5.861678  , 5.5474441 , 3.96629375,
       4.80034684, 3.63814892, 4.85275761, 4.74743157, 6.05887431,
       3.81417697, 4.72764543, 4.05998048, 3.82717898, 5.14252653,
       4.57021613, 6.81124042, 5.56809399, 5.89930521, 3.96756721,
       5.41903066, 3.94129817, 6.21709314, 3.95006075, 4.03519643,
       6.45992919, 6.2459964 , 4.14882513, 3.70811563, 5.49520864,
       3.78314396, 6.81865121, 3.82582756, 3.77085697, 4.97857596,
       4.97853531, 4.66506403, 6.90657299, 4.29380961, 5.12883364,
       4.08346398, 6.52640756, 5.43542177, 4.72288735, 4.62812284,
       4.07219149, 4.58267084, 6.019496  , 5.07218453, 6.31971191,
       5.36374679, 3.6385272 , 4.48751974, 5.47408305, 3.81985096,
       5.27556177, 5.3616683 , 4.80582684, 4.9217133 , 5.37640882,
       3.38480937, 6.83221322, 4.9058687 , 4.18822673, 6.03647384,
       6.60452465, 5.53483373, 6.74654199, 6.84858904, 4.61358551,
       4.62007622, 3.85553402, 5.34257688, 4.78058369, 4.18192818,
       5.54759001, 4.89337402, 3.75256894, 3.81306442, 4.90212683,
       3.61204068, 6.6706726 , 3.86291923, 3.80406208, 7.03822381,
       4.86761056, 4.93017755, 4.72288735, 4.67763742, 5.46411233,
       3.91157347, 5.38491541, 4.25921808, 4.80077285, 5.11187913,
       4.26302103, 6.20982069, 3.69384176, 3.7243979 , 4.03894484,
       4.01525162, 4.67230338, 6.88642707, 3.9260423 , 5.34276246,
       3.59895294, 4.0428277 , 3.89866803, 5.1286554 , 5.58614029,
       5.60194323, 4.05787009, 5.31690733, 4.02591086, 5.68828458,
```

```
       6.60612818, 5.02410545, 6.83606978, 5.08176092, 6.83226998,
       5.05603013, 3.77348732, 4.57250334, 3.83674248, 3.77821226,
       4.37529801, 4.08359094, 6.14366791, 6.02729905, 4.07613711,
       4.707273  , 3.91257032, 4.99162091, 3.74984923, 4.89118298,
       3.53987846, 5.7623143 , 4.15816367, 5.50287224, 3.66553435,
       5.29063841, 3.64085282, 5.58614029, 5.05262274, 3.7419973 ,
       3.87738943, 3.95771349, 5.2199185 , 3.82186852, 5.47315613,
       5.50550025, 4.70813192, 5.60232414, 3.73979037, 4.61600927,
       5.92755423, 5.04300382, 3.49088059, 4.96472934, 5.58864868,
       4.84828639, 3.78560675, 6.55790797, 4.60340474, 3.57485049,
       4.96645787, 6.28302371, 4.44606253, 5.23251427, 3.86933165,
       5.18453078, 5.12268573, 6.09150201, 3.553885  , 4.14082978,
       3.32906056, 4.36492319, 4.99472362, 5.87042867, 4.28820362,
       3.66818259, 3.98061781, 4.4853977 , 4.9444642 , 3.70739311,
       4.95821719, 4.6501794 , 3.87257097, 5.40416232, 3.77124548,
       4.00803117, 3.79257172, 5.88733168, 3.82607971, 3.75460604,
       7.47227245, 5.54705234, 5.49520864, 3.84176809, 6.78104942,
       6.7908776 , 5.90430697, 4.99689697, 3.71346863, 5.98826607,
       4.95992187, 5.50755632, 6.00266836, 5.20991425, 5.99698667,
       4.67930774, 5.17977542, 4.64039337, 4.0808636 , 6.67752977,
       4.02087523, 3.96866858, 3.98846811, 6.02884356, 6.41287872,
       4.02548464, 4.02883501, 5.29993228, 5.63219382, 4.22952486,
       4.8762535 , 4.50401913, 6.25162955, 6.51924451, 3.48206464,
       4.30923736, 6.16226474, 4.03257773, 6.31183543, 5.64000449,
       5.21392511, 5.46298792, 6.48811376, 3.64208819, 4.22048689,
       4.72176406, 3.86306898, 5.16189063, 5.03787134, 6.46214881,
       3.93439481, 5.26393663, 3.99831768, 5.1377641 , 3.695829  ,
       4.40677869, 5.80576394, 5.42878736, 4.28022108, 3.89077479,
       5.64303116, 3.78560675, 5.784532  , 5.4978573 , 3.70859059,
       4.72197402, 4.18020017, 5.58233651, 6.26057536, 3.27980893,
       6.00914029, 4.99253595, 4.0269963 , 5.76060322, 5.47315613,
       4.00724435, 3.63349804, 4.29890616, 5.05115735, 3.56334557,
       4.75403951, 4.12066649, 5.38719776, 3.91354081, 6.17216038,
       5.07218453, 4.53387021, 3.46440905, 4.18175253])
```

```
Y_pred_1_train=LinReg.predict(X_train)
Y_pred_1_train
```

```
array([5.36140723, 4.88567986, 5.38440789, ..., 4.83099135,
5.80418812,
       4.2617047 ])
```

```
Quality=pd.DataFrame({'Actual quality:':Y_test,'Predicted
quality':Y_pred_1})
Quality
```

	Actual quality:	Predicted quality
188	5	5.027261
2083	3	4.093279
1675	3	3.792223

```
1089                  7           5.769745
1378                  6           4.956214
...                 ...                ...
1580                  6           6.172160
1047                  5           5.072185
892                   6           4.533870
1674                  3           3.464409
8                     7           4.181753

[454 rows x 2 columns]
```

Evaluation of Model-1

```
from sklearn import metrics

print("Testing Accuracy : ",metrics.r2_score(Y_test,Y_pred_1))
print("Training Accuracy : ",metrics.r2_score(Y_train,Y_pred_1_train))
print("Mean Squared Error :
",metrics.mean_squared_error(Y_test,Y_pred_1))
print("Root Mean Squared Error :
",np.sqrt(metrics.mean_squared_error(Y_test,Y_pred_1)))

Testing Accuracy :  0.5196342611962517
Training Accuracy :  0.530025681114703
Mean Squared Error :  0.9026562488758573
Root Mean Squared Error :  0.9500822326913904
```

Testing Model-1 with random values

```
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
```

```
LinReg.predict([[7.2,0.54,0.02,1.14,0.074,7,31,0.9946,3.48,0.67,9.25]]
)
```

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

array([10.71390243])
```

```
LinReg.predict([[8.9,0.23,0.03,1.5,0.062,5,28,0.9918,3.14,0.51,9.44]])
```

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

array([9.86629743])
```

Train - Test split for Model - 2

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test =
train_test_split(X_scaled,Y_resampled,test_size=0.2,random_state=42)

X_train.shape, X_test.shape

((1816, 11), (454, 11))

Y_train.shape, Y_test.shape

((1816,), (454,))
```

Model Training -2 (Logistic Regression)

```
from sklearn.linear_model import LogisticRegression

LogReg=LogisticRegression(multi_class='ovr')

LogReg.fit(X_train,Y_train)

LogisticRegression(multi_class='ovr')

Y_pred_2=LogReg.predict(X_test)
Y_pred_2

array([5, 3, 3, 6, 5, 5, 6, 5, 6, 3, 6, 3, 3, 6, 3, 3, 3, 6, 6, 6, 6,
6,
       3, 5, 3, 6, 3, 5, 3, 3, 5, 5, 3, 6, 3, 3, 6, 3, 3, 3, 6, 3, 3,
5,
       3, 6, 5, 5, 5, 3, 3, 3, 3, 3, 5, 5, 6, 6, 6, 5, 6, 6, 5, 6, 6,
6,
       6, 5, 6, 6, 3, 7, 6, 3, 3, 5, 3, 3, 3, 5, 6, 3, 3, 3, 3, 6, 5,
3,
       3, 6, 6, 3, 3, 3, 5, 3, 3, 5, 6, 3, 3, 5, 5, 6, 3, 6, 5, 5, 6,
5,
       3, 6, 3, 3, 6, 3, 6, 6, 5, 6, 6, 3, 3, 3, 3, 3, 5, 3, 3, 5, 3,
7,
       5, 6, 3, 5, 3, 3, 5, 3, 5, 6, 3, 6, 4, 5, 3, 3, 3, 5, 6, 3, 3,
5,
       6, 3, 6, 6, 3, 6, 3, 3, 3, 6, 5, 5, 3, 5, 5, 3, 6, 5, 5, 5, 5,
5,
       3, 5, 3, 6, 3, 5, 3, 3, 6, 3, 6, 6, 6, 3, 6, 3, 6, 3, 3, 6, 6,
3,
       3, 6, 3, 6, 5, 3, 5, 6, 5, 7, 3, 6, 3, 6, 5, 6, 3, 5, 5, 5, 6,
6,
       6, 3, 5, 6, 3, 5, 5, 3, 6, 6, 3, 7, 3, 5, 6, 6, 5, 6, 6, 6, 5,
3,
       5, 5, 5, 5, 5, 3, 3, 5, 3, 6, 3, 3, 6, 5, 5, 6, 5, 5, 3, 6, 3,
5,
       5, 3, 6, 3, 3, 3, 5, 3, 6, 3, 6, 3, 3, 3, 6, 6, 6, 3, 6, 3, 6,
6,
       5, 6, 5, 6, 5, 3, 5, 3, 3, 5, 3, 5, 6, 3, 5, 5, 5, 3, 6, 3, 6,
3,
       6, 3, 5, 3, 6, 5, 3, 3, 5, 5, 3, 6, 5, 5, 5, 3, 6, 6, 5, 3, 6,
5,
       6, 3, 6, 3, 3, 6, 6, 5, 5, 3, 5, 6, 6, 3, 5, 3, 5, 6, 5, 5, 3,
3,
       5, 5, 3, 5, 5, 3, 5, 3, 3, 3, 5, 3, 3, 7, 5, 6, 3, 7, 6, 5, 5,
3,
       6, 5, 6, 5, 5, 6, 3, 6, 6, 3, 6, 3, 3, 3, 6, 6, 3, 3, 5, 5, 3,
5,
       3, 7, 6, 3, 5, 6, 5, 6, 5, 5, 5, 6, 3, 5, 5, 3, 5, 5, 6, 3, 5,
```

```
3,
       6, 3, 5, 6, 6, 5, 3, 5, 3, 5, 5, 3, 5, 5, 6, 6, 3, 6, 5, 3, 6,
6,
       3, 3, 5, 6, 3, 5, 3, 6, 3, 6, 6, 3, 3, 3])
```

```
Y_pred_2_train=LogReg.predict(X_train)
Y_pred_2_train
```

```
array([5, 3, 6, ..., 6, 6, 5])
```

```
Quality_2=pd.DataFrame({'Actual quality:':Y_test,'Predicted
quality':Y_pred_2})
Quality_2
```

```
      Actual quality:  Predicted quality
188                 5                  5
2083                3                  3
1675                3                  3
1089                7                  6
1378                6                  5
...               ...                ...
1580                6                  6
1047                5                  6
892                 6                  3
1674                3                  3
8                   7                  3

[454 rows x 2 columns]
```

Evaluation of Model-2

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

print("Testing Accuracy : ",accuracy_score(Y_test,Y_pred_2))
print("Training Accuracy : ",accuracy_score(Y_train,Y_pred_2_train))
```

```
Testing Accuracy :  0.6453744493392071
Training Accuracy :  0.6618942731277533
```

```
confusion_matrix(Y_test,Y_pred_2)
```

```
array([[136,   0,   0,   0,   0,   0],
       [  1,   0,   4,   3,   0,   0],
       [ 27,   1,  83,  29,   1,   0],
       [ 12,   0,  44,  71,   3,   0],
       [  1,   0,   3,  27,   3,   0],
       [  0,   0,   0,   5,   0,   0]])
```

```
pd.crosstab(Y_test,Y_pred_2)
```

```
col_0      3  4   5   6  7
quality
3        136  0   0   0  0
4          1  0   4   3  0
5         27  1  83  29  1
6         12  0  44  71  3
7          1  0   3  27  3
8          0  0   0   5  0
```

```
print(classification_report(Y_test,Y_pred_2))
```

```
              precision    recall  f1-score   support

           3       0.77      1.00      0.87       136
           4       0.00      0.00      0.00         8
           5       0.62      0.59      0.60       141
           6       0.53      0.55      0.54       130
           7       0.43      0.09      0.15        34
           8       0.00      0.00      0.00         5

    accuracy                           0.65       454
   macro avg       0.39      0.37      0.36       454
weighted avg       0.61      0.65      0.61       454


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Testing Model-2 with random values

```
LogReg.predict([[8.7,0.5,0.03,1.1,0.08,9,31,0.9998,3.87,0.48,9.83]])
```

```
/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(
```

```
array([4])
```

```
LogReg.predict([[7.2,0.6,0.05,2.8,0.066,12,23,0.9774,3.29,0.29,9.47]])
```

```
/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(
```

```
array([8])
```

Train - Test split for Model-3

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test =
train_test_split(X_resampled,Y_resampled,test_size=0.2,random_state=30
)

X_train.shape, X_test.shape

((1816, 11), (454, 11))

Y_train.shape, Y_test.shape

((1816,), (454,))
```

Model Training - 3 (Random Forest Classifier)

```
from sklearn.ensemble import RandomForestClassifier
RanFor = RandomForestClassifier(criterion='entropy')

RanFor.fit(X_train,Y_train)

RandomForestClassifier(criterion='entropy')

Y_pred_3 = RanFor.predict(X_test)
Y_pred_3

array([5, 3, 6, 6, 3, 5, 3, 6, 5, 5, 5, 6, 3, 6, 6, 3, 3, 6, 5, 5, 5,
5,
       5, 6, 5, 6, 7, 3, 6, 5, 7, 7, 5, 5, 3, 5, 5, 6, 3, 3, 7, 3, 6,
3,
       7, 5, 6, 6, 7, 3, 3, 3, 6, 6, 3, 3, 5, 3, 5, 5, 6, 3, 5, 6, 5,
6,
       5, 3, 6, 5, 3, 7, 3, 6, 5, 6, 5, 3, 6, 5, 3, 3, 6, 6, 6, 6, 3,
5,
       6, 6, 3, 3, 5, 5, 3, 3, 6, 6, 7, 5, 3, 3, 5, 5, 5, 3, 6, 3, 6,
6,
       5, 6, 3, 3, 6, 8, 3, 6, 7, 3, 6, 3, 6, 6, 6, 5, 3, 5, 6, 5, 5,
3,
```

```
       3, 6, 5, 3, 7, 3, 6, 3, 5, 5, 5, 7, 6, 5, 5, 6, 3, 3, 3, 6, 3,
3,
       3, 5, 6, 7, 5, 3, 6, 5, 5, 3, 5, 5, 6, 6, 3, 6, 3, 3, 5, 6, 3,
6,
       7, 5, 3, 6, 3, 5, 5, 3, 3, 7, 5, 6, 3, 6, 5, 5, 5, 5, 5, 3, 6,
3,
       6, 3, 3, 6, 5, 3, 5, 5, 6, 3, 3, 5, 6, 3, 6, 5, 3, 5, 6, 3, 5,
6,
       5, 5, 5, 3, 6, 6, 5, 5, 6, 5, 5, 5, 5, 6, 5, 5, 3, 3, 5, 5, 5,
6,
       3, 6, 3, 3, 5, 6, 6, 5, 3, 5, 7, 3, 6, 5, 3, 5, 6, 3, 3, 5, 3,
3,
       5, 6, 6, 5, 5, 7, 5, 6, 3, 6, 3, 3, 3, 5, 3, 3, 5, 3, 7, 6, 5,
3,
       5, 5, 6, 6, 5, 6, 6, 6, 5, 6, 3, 3, 5, 3, 5, 6, 3, 5, 5, 5, 7,
3,
       3, 3, 3, 6, 6, 3, 6, 5, 3, 3, 5, 5, 6, 5, 5, 6, 6, 3, 3, 3, 3,
6,
       3, 3, 6, 5, 5, 7, 3, 3, 6, 3, 6, 5, 3, 6, 3, 3, 6, 5, 6, 5, 5,
5,
       6, 6, 7, 6, 5, 5, 6, 3, 3, 3, 5, 3, 6, 3, 5, 3, 3, 3, 3, 3, 6,
6,
       3, 6, 6, 6, 5, 5, 3, 6, 5, 6, 3, 3, 6, 6, 7, 7, 6, 5, 6, 3, 6,
5,
       3, 3, 7, 3, 6, 6, 7, 3, 5, 3, 5, 5, 6, 5, 5, 6, 6, 5, 5, 5, 3,
3,
       7, 5, 7, 6, 3, 5, 5, 3, 5, 3, 5, 3, 3, 5, 5, 6, 5, 3, 3, 3, 3,
3,
       3, 5, 3, 7, 3, 5, 6, 3, 3, 3, 6, 5, 5, 6])
```

```
Y_pred_3_train = RanFor.predict(X_train)
Y_pred_3_train
```

```
array([5, 5, 3, ..., 6, 7, 3])
```

Evaluation for Model-3

```
print("Testing Accuracy = ", accuracy_score(Y_test,Y_pred_3))
print("Training Accuracy = ",accuracy_score(Y_train,Y_pred_3_train))
```

```
Testing Accuracy =  0.801762114537445
Training Accuracy =  1.0
```

```
pd.crosstab(Y_test,Y_pred_3)
```

col_0	3	5	6	7	8
quality					
3	151	1	0	0	0
4	2	6	7	0	0
5	1	106	23	1	0

```
6           0   29  90   7  0
7           0    3   6  17  1
8           0    0   1   2  0
```

```python
print(classification_report(Y_test,Y_pred_3))
```

```
              precision    recall  f1-score   support

           3       0.98      0.99      0.99       152
           4       0.00      0.00      0.00        15
           5       0.73      0.81      0.77       131
           6       0.71      0.71      0.71       126
           7       0.63      0.63      0.63        27
           8       0.00      0.00      0.00         3

    accuracy                           0.80       454
   macro avg       0.51      0.52      0.52       454
weighted avg       0.77      0.80      0.79       454


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```python
print(confusion_matrix(Y_test,Y_pred_3))
```

```
[[151   0   1   0   0   0]
 [  2   0   6   7   0   0]
 [  1   0 106  23   1   0]
 [  0   0  29  90   7   0]
 [  0   0   3   6  17   1]
 [  0   0   0   1   2   0]]
```

Testing Model-3 with random values

```python
RanFor.predict([[7.6,0.81,0.34,2.5,0.052,14,26,0.9936,3.45,0.59,10.23]
])
```

```
/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(
```

```
array([6])
```

```
RanFor.predict([[6.5,0.75,0.29,1.3,0.074,21,29,0.9946,3.28,0.62,9.19]]
)
```

```
/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(
```

```
array([3])
```