```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

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

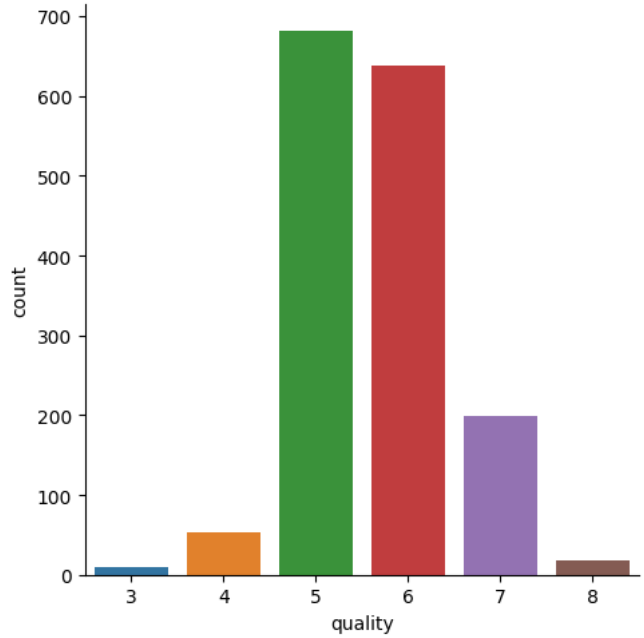
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

VISUALIZATION

Univariate analysis

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

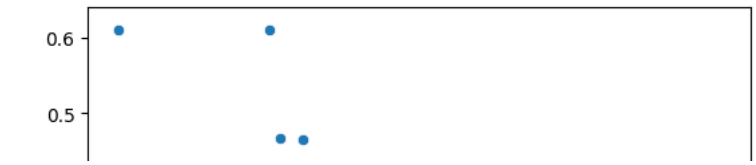
```
<seaborn.axisgrid.FacetGrid at 0x7d36d631b910>
```



Bivariate analysis

```
sns.scatterplot(x=df.pH,y=df.chlorides)
```

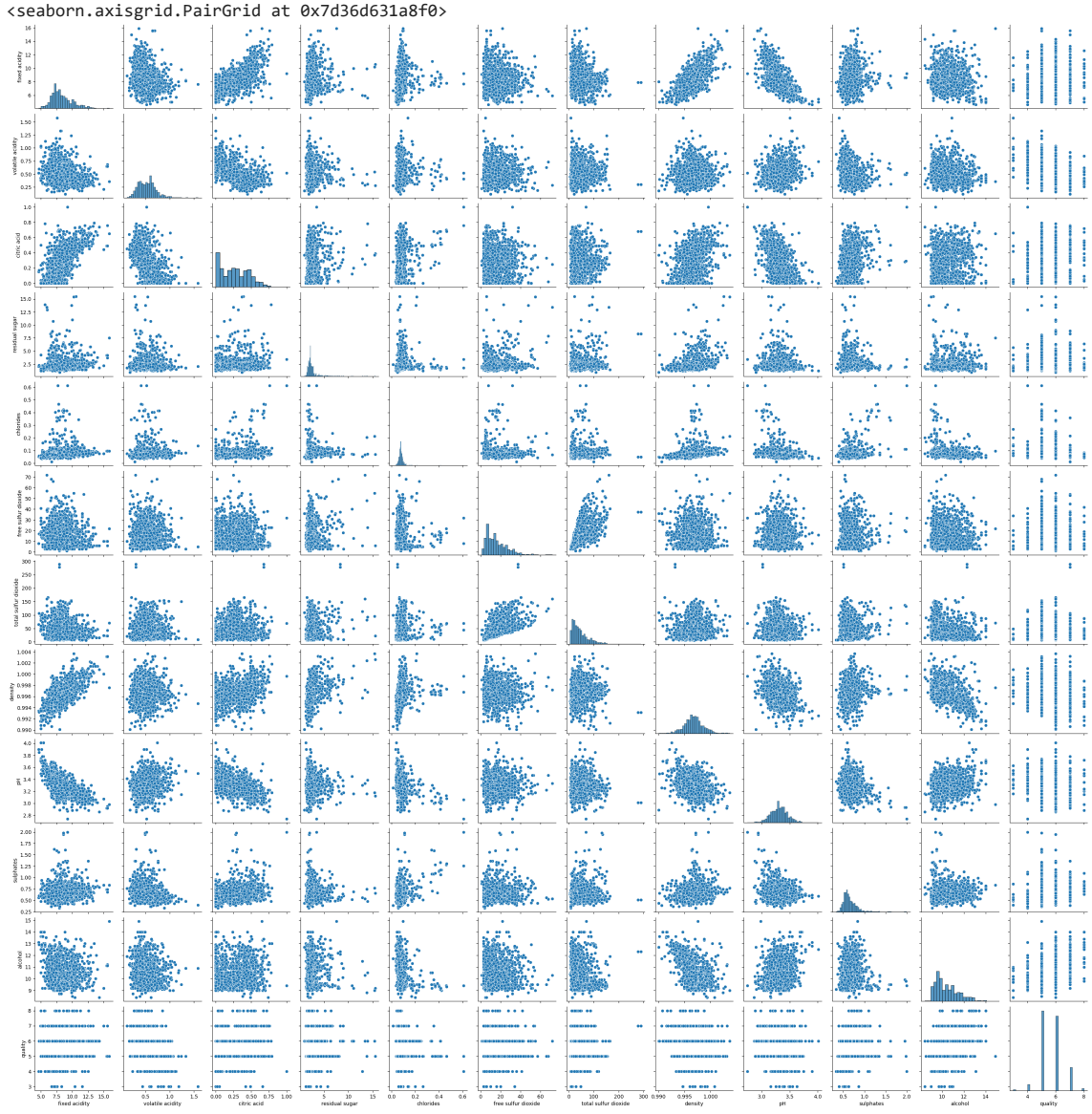
```
<Axes: xlabel='pH', ylabel='chlorides'>
```



Multivariate analysis

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

```
<seaborn.axisgrid.PairGrid at 0x7d36d631a8f0>
```



Check for missing 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
```

```
df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	159
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	

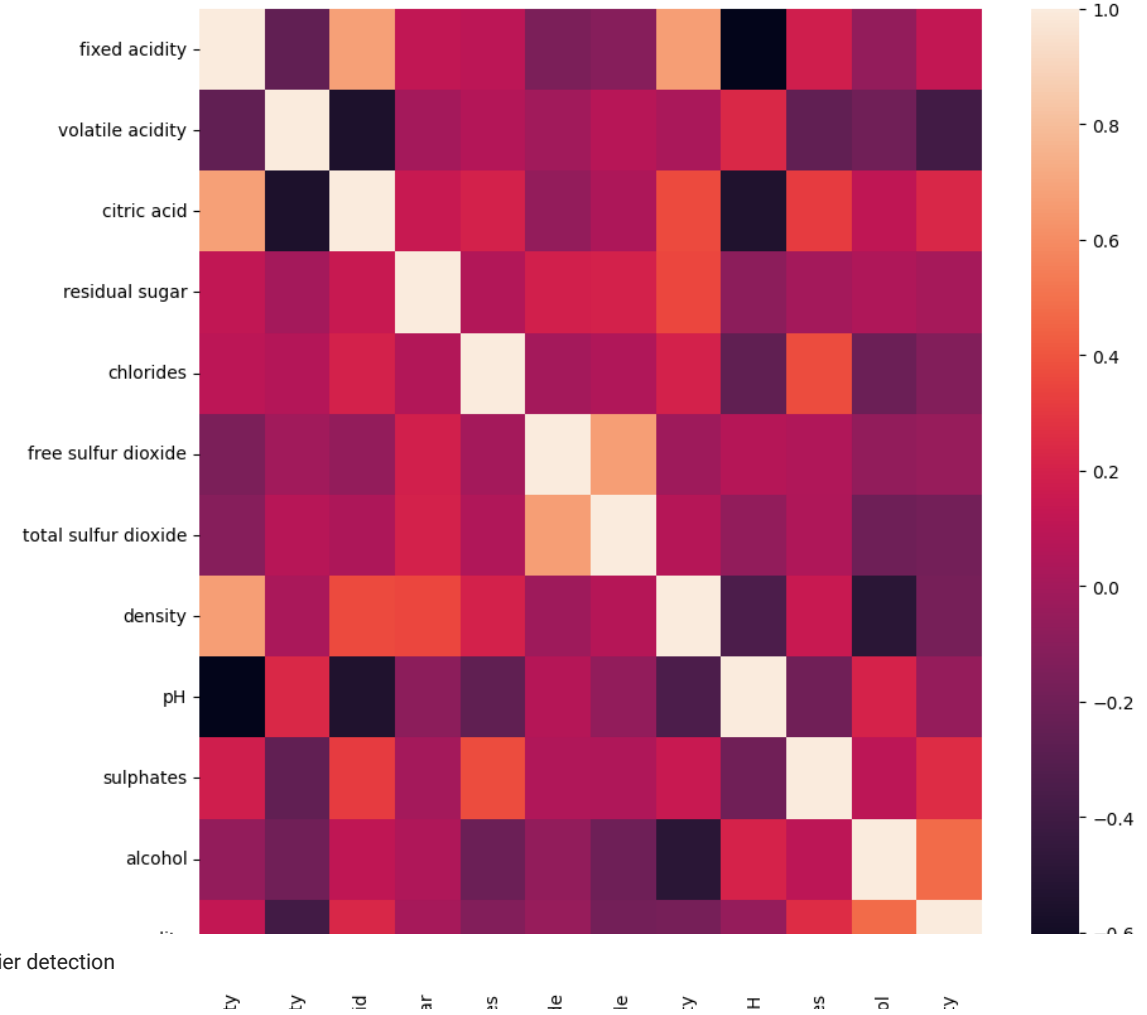
```
correlation=df.corr()
correlation
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.668047	-0.682978	0.18300
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234937	-0.26098
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904	0.31277
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.355283	-0.085652	0.00552
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.200632	-0.265026	0.37126
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666	-0.021946	0.070377	0.05165
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000	0.071269	-0.066495	0.04294
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269	1.000000	-0.341699	0.14850
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070377	-0.066495	-0.341699	1.000000	-0.19664
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051658	0.042947	0.148506	-0.196648	1.00000
alcohol	-0.061668	-0.202288	0.109903	0.042075	-0.221141	-0.069408	-0.205654	-0.496180	0.205633	0.09359
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050656	-0.185100	-0.174919	-0.057731	0.25139

Constructing heatmap

```
plt.figure(figsize=(10,10))
sns.heatmap(correlation)
```

```
<Axes: >
```



outlier detection

```
df.columns = df.columns.str.replace(' ', '_')
df.head()
```

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_d
0	7.4	0.70	0.00	1.9	0.076	11.0	
1	7.8	0.88	0.00	2.6	0.098	25.0	
2	7.8	0.76	0.04	2.3	0.092	15.0	
3	11.2	0.28	0.56	1.9	0.075	17.0	
4	7.4	0.70	0.00	1.9	0.076	11.0	

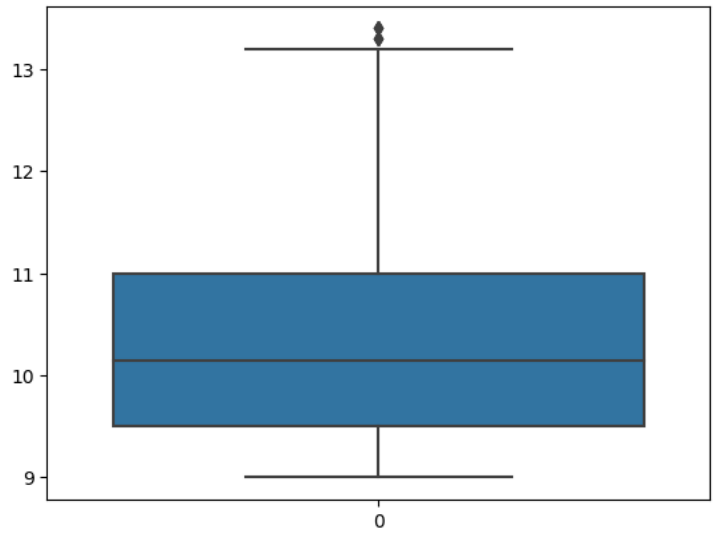
```
sns.boxplot(df.alcohol)
```

```
a_01=df.alcohol.quantile(0.01)
a_99=df.alcohol.quantile(0.99)
print(a_01)
print(a_99)
```

```
9.0
13.4
```

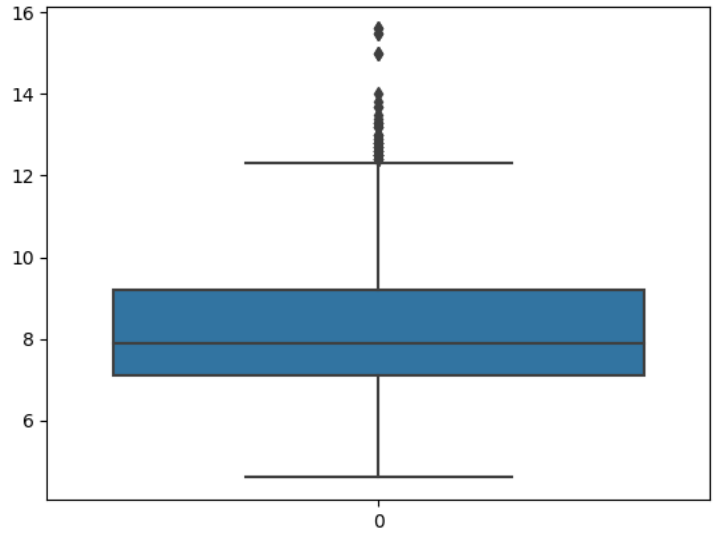
```
df=df[(df.alcohol>=a_01) & (df.alcohol<=a_99)]
sns.boxplot(df.alcohol)
```

```
<Axes: >
```




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

```
<Axes: >
```

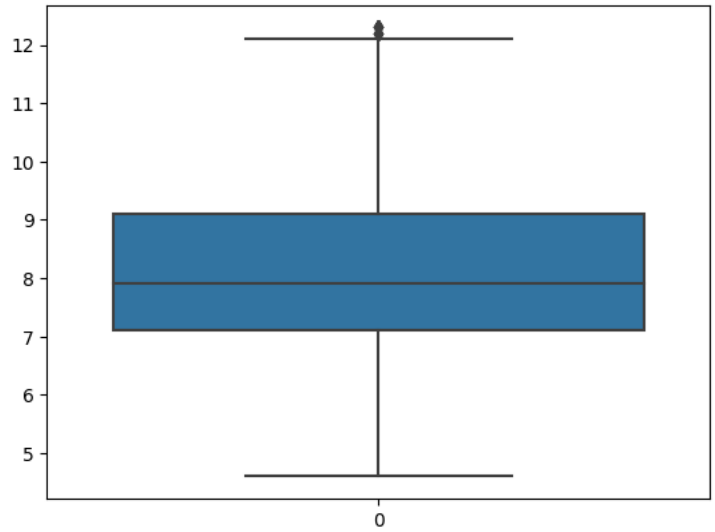



```
f1 = df.fixed_acidity.quantile(0.25) #Q1
f3 = df.fixed_acidity.quantile(0.75) #Q3
IQR_f = f3 - f1
upper_limit_f = f3+(1.5)*(IQR_f)
lower_limit_f = f1-(1.5)*(IQR_f)
print(f1)
print(f3)
print(IQR_f)
print(upper_limit_f)
print(lower_limit_f)
```

```
7.1
9.2
2.0999999999999996
12.349999999999998
3.95
```

```
df=df[(df.fixed_acidity<upper_limit_f) & (df.fixed_acidity>lower_limit_f)]
sns.boxplot(df.fixed_acidity)
```

```
<Axes: >
```

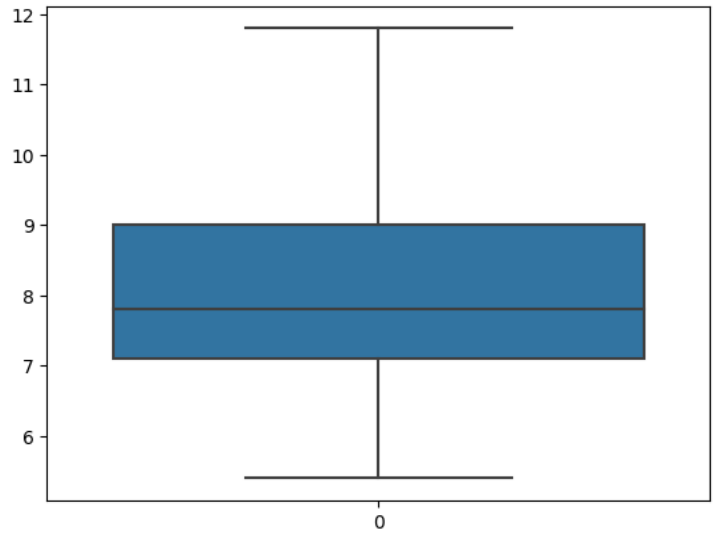



```
fa_01=df.fixed_acidity.quantile(0.01)
fa_98=df.fixed_acidity.quantile(0.98)
print(fa_01)
print(fa_98)
```

```
5.4
11.8
```

```
df=df[(df.fixed_acidity>=fa_01) & (df.fixed_acidity<=fa_98)]
sns.boxplot(df.fixed_acidity)
```

```
<Axes: >
```




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

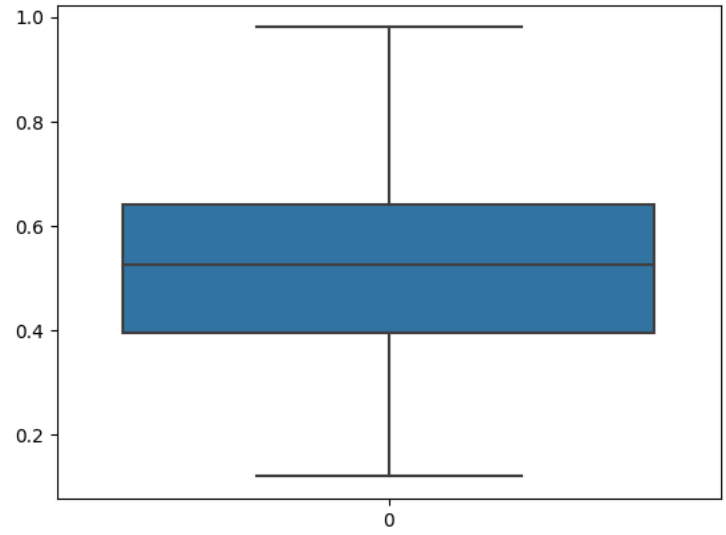
```
<Axes: >
```


```
v1 = df.volatile_acidity.quantile(0.25) #Q1
v3 = df.volatile_acidity.quantile(0.75) #Q3
IQR_v = v3 - v1
upper_limit_v = v3+(1.5)*(IQR_v)
lower_limit_v = v1-(1.5)*(IQR_v)
print(v1)
print(v3)
print(IQR_v)
print(upper_limit_v)
print(lower_limit_v)
```

```
0.4
0.64
0.24
1.0
0.040000000000000036
```

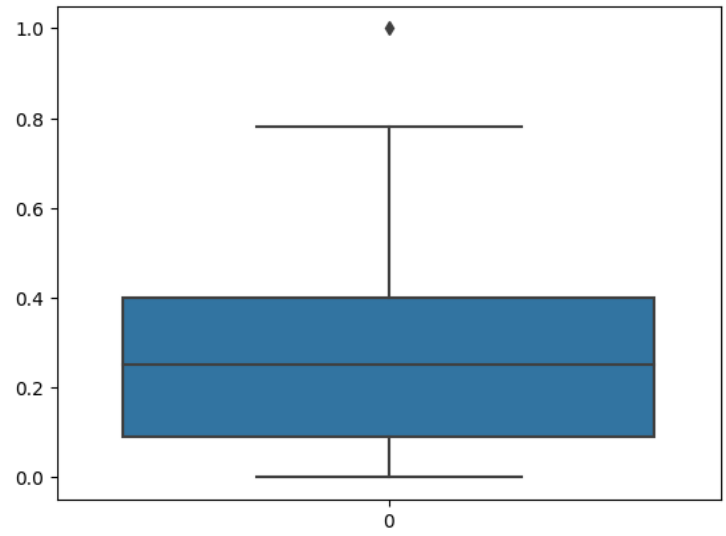
```
df=df[(df.volatile_acidity<upper_limit_v) & (df.volatile_acidity>lower_limit_v)]
sns.boxplot(df.volatile_acidity)
```

```
<Axes: >
```




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

```
<Axes: >
```



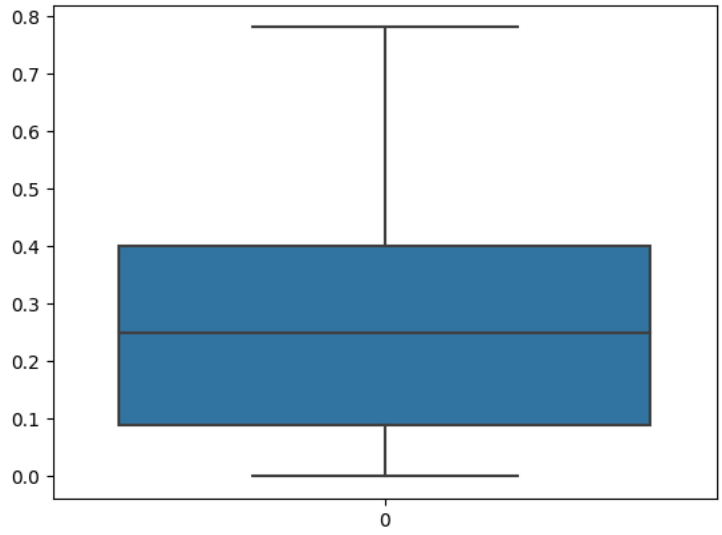

```
c1 = df.citric_acid.quantile(0.25) #Q1
c3 = df.citric_acid.quantile(0.75) #Q3
IQR_c = c3 - c1
upper_limit_c = c3+(1.5)*(IQR_c)
lower_limit_c = c1-(1.5)*(IQR_c)
print(c1)
```

```
print(c3)
print(IQR_c)
print(upper_limit_c)
print(lower_limit_c)
```

```
0.09
0.4
0.31000000000000005
0.8650000000000001
-0.3750000000000001
```

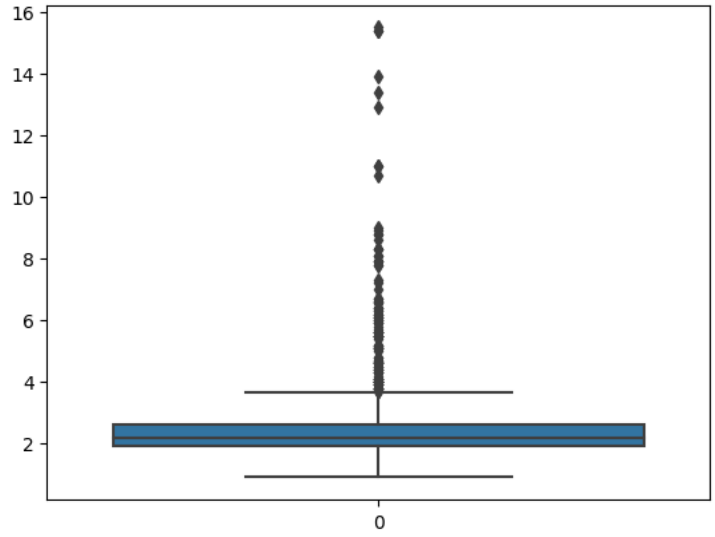
```
df=df[(df.citric_acid<upper_limit_c) & (df.citric_acid>lower_limit_c)]
sns.boxplot(df.citric_acid)
```

```
<Axes: >
```




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

```
<Axes: >
```



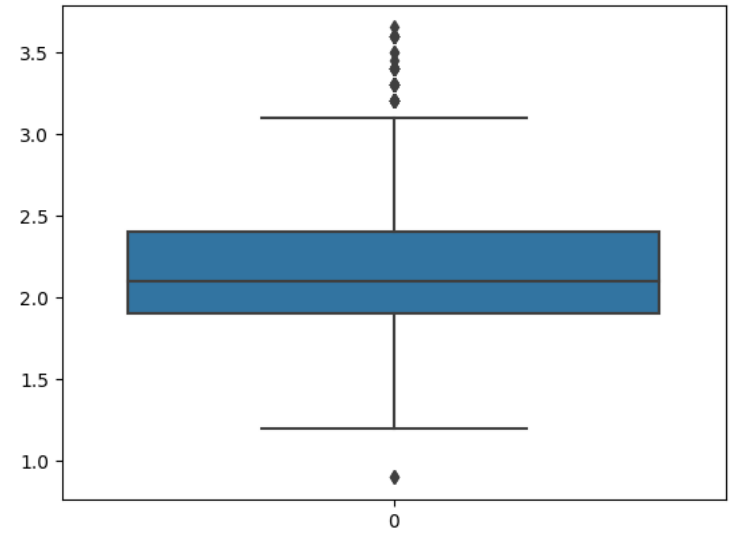

```
r1 = df.residual_sugar.quantile(0.25) #Q1
r3 = df.residual_sugar.quantile(0.75) #Q3
IQR_r = r3 - r1
upper_limit_r = r3+(1.5)*(IQR_r)
lower_limit_r = r1-(1.5)*(IQR_r)
print(r1)
print(r3)
print(IQR_r)
print(upper_limit_r)
print(lower_limit_r)
```

```
1.9
2.6
0.7000000000000002
3.6500000000000004
0.8499999999999996
```

```
df=df[(df.residual_sugar<upper_limit_r) & (df.residual_sugar>lower_limit_r)]
```

```
df=df[(df.residual_sugar<upper_limit_r) & (df.residual_sugar>lower_limit_r)]
sns.boxplot(df.residual_sugar)
```

```
<Axes: >
```

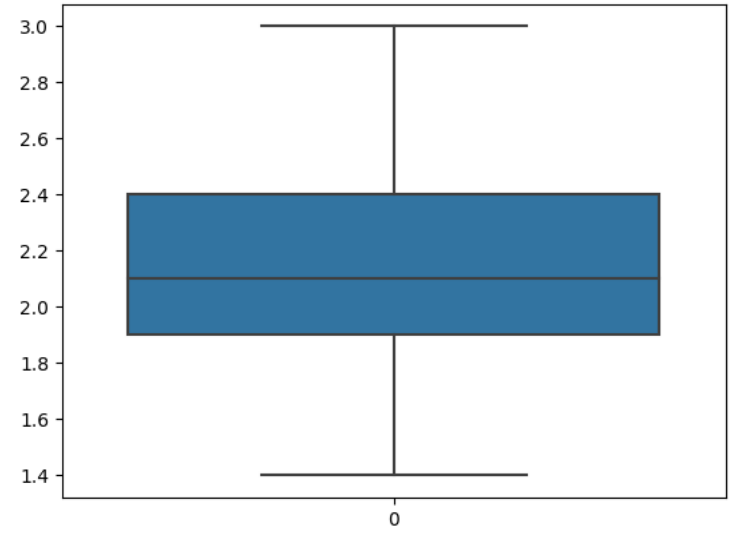



```
rs_02=df.residual_sugar.quantile(0.02)
rs_96=df.residual_sugar.quantile(0.96)
print(rs_02)
print(rs_96)
```

```
1.4
3.0
```

```
df=df[(df.residual_sugar>=rs_02) & (df.residual_sugar<=rs_96)]
sns.boxplot(df.residual_sugar)
```

```
<Axes: >
```




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

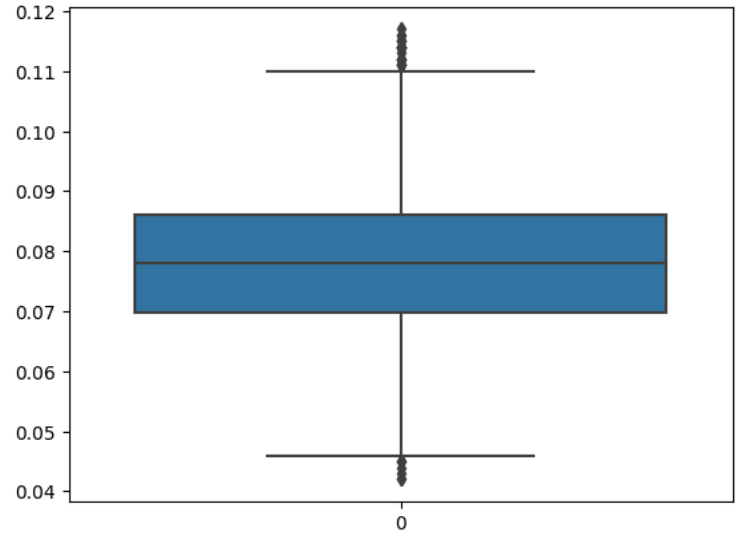
```
<Axes: >
```


```
ch1 = df.chlorides.quantile(0.25) #Q1
ch3 = df.chlorides.quantile(0.75) #Q3
IQR_ch = ch3 - ch1
upper_limit_ch = ch3+(1.5)*(IQR_ch)
lower_limit_ch = ch1-(1.5)*(IQR_ch)
print(ch1)
print(ch3)
print(IQR_ch)
print(upper_limit_ch)
print(lower_limit_ch)
```

```
0.07
0.089
0.01899999999999999
0.11749999999999998
0.04150000000000002
```

```
df=df[(df.chlorides<upper_limit_ch) & (df.chlorides>lower_limit_ch)]
sns.boxplot(df.chlorides)
```

```
<Axes: >
```




```
ch_01=df.chlorides.quantile(0.01)
ch_97=df.chlorides.quantile(0.97)
print(ch_01)
print(ch_97)
```

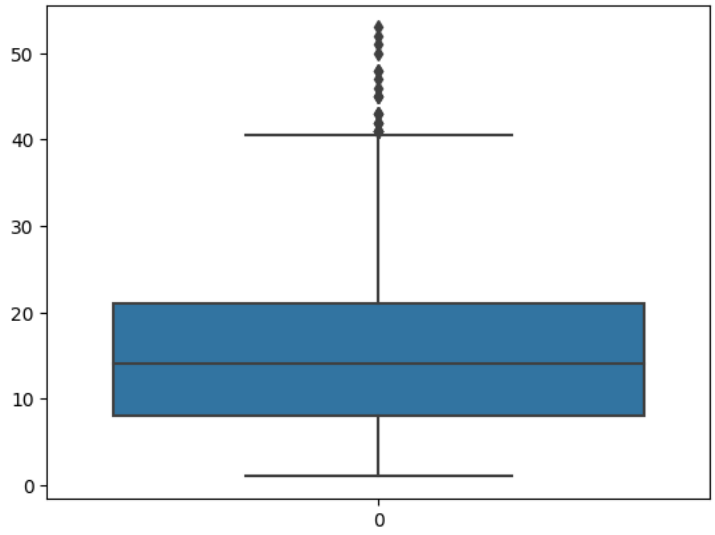
```
0.04775
0.107
```

```
df=df[(df.chlorides>=ch_01) & (df.chlorides<=ch_97)]
sns.boxplot(df.chlorides)
```

```
<Axes: >
```

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

```
<Axes: >
```

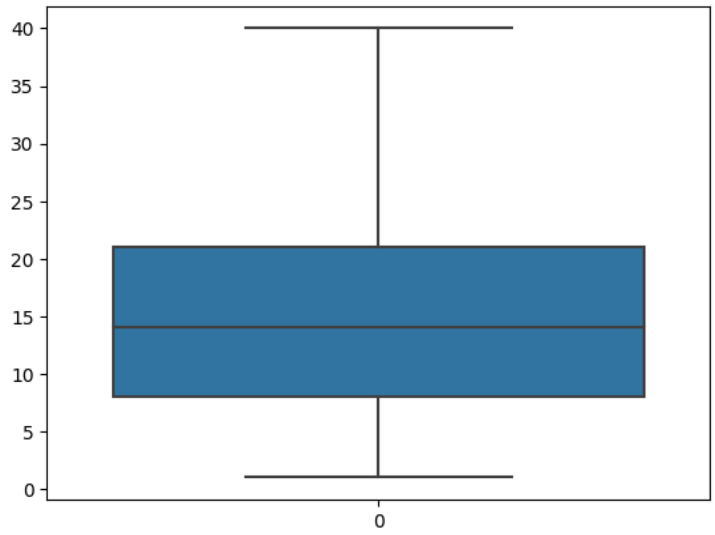



```
fs1 = df.free_sulfur_dioxide.quantile(0.25) #Q1
fs3 = df.free_sulfur_dioxide.quantile(0.75) #Q3
IQR_fs = fs3 - fs1
upper_limit_fs = fs3+(1.5)*(IQR_fs)
lower_limit_fs = fs1-(1.5)*(IQR_fs)
print(fs1)
print(fs3)
print(IQR_fs)
print(upper_limit_fs)
print(lower_limit_fs)
```

```
8.0
21.0
13.0
40.5
-11.5
```

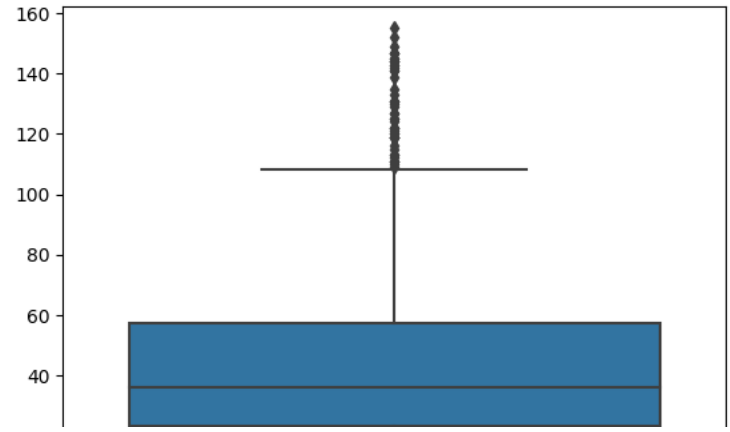
```
df=df[(df.free_sulfur_dioxide<upper_limit_fs) & (df.free_sulfur_dioxide>lower_limit_fs)]
sns.boxplot(df.free_sulfur_dioxide)
```

```
<Axes: >
```




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

```
<Axes: >
```

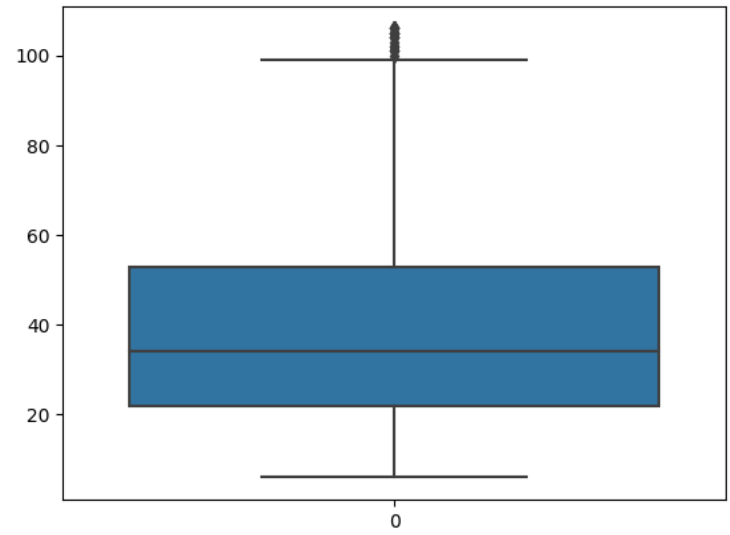



```
ts1 = df.total_sulfur_dioxide.quantile(0.25) #Q1
ts3 = df.total_sulfur_dioxide.quantile(0.75) #Q3
IQR_ts = ts3 - ts1
upper_limit_ts = ts3+(1.5)*(IQR_ts)
lower_limit_ts = ts1-(1.5)*(IQR_ts)
print(ts1)
print(ts3)
print(IQR_ts)
print(upper_limit_ts)
print(lower_limit_ts)
```

```
23.0
57.0
34.0
108.0
-28.0
```

```
df=df[(df.total_sulfur_dioxide<upper_limit_ts) & (df.total_sulfur_dioxide>lower_limit_ts)]
sns.boxplot(df.total_sulfur_dioxide)
```

```
<Axes: >
```

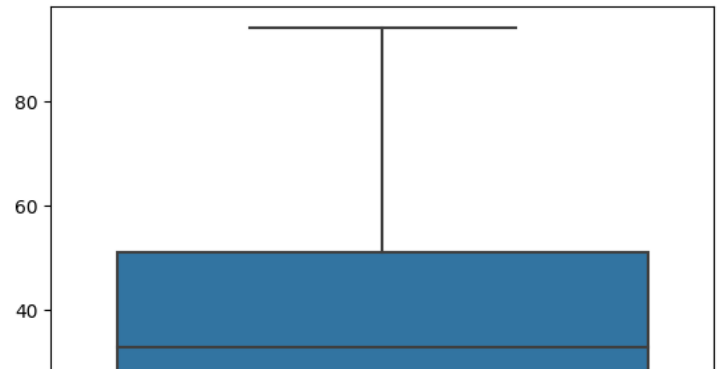



```
ts_01=df.total_sulfur_dioxide.quantile(0.01)
ts_97=df.total_sulfur_dioxide.quantile(0.97)
print(ts_01)
print(ts_97)
```

```
8.0
94.43999999999994
```

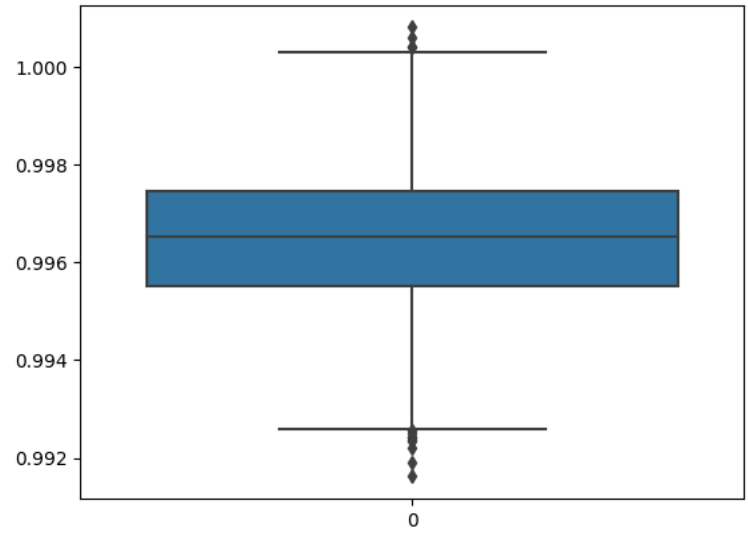
```
df=df[(df.total_sulfur_dioxide>=ts_01) & (df.total_sulfur_dioxide<=ts_97)]
sns.boxplot(df.total_sulfur_dioxide)
```

```
<Axes: >
```




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

```
<Axes: >
```




```
d1 = df.density.quantile(0.25) #Q1
d3 = df.density.quantile(0.75) #Q3
IQR_d = d3 - d1
upper_limit_d = d3+(1.5)*(IQR_d)
lower_limit_d = d1-(1.5)*(IQR_d)
print(d1)
print(d3)
print(IQR_d)
print(upper_limit_d)
print(lower_limit_d)
```

```
0.9955
0.99745
0.0019499999999998963
1.0003749999999998
0.9925750000000002
```

```
df=df[(df.density<upper_limit_d) & (df.density>lower_limit_d)]
sns.boxplot(df.density)
```

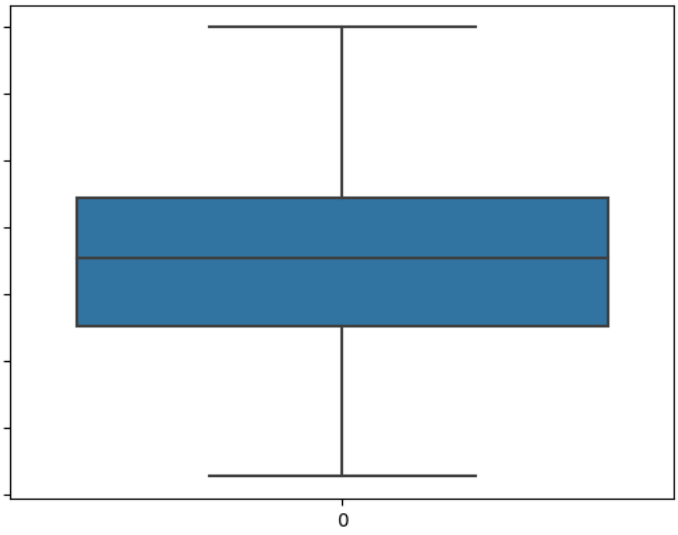
```
<Axes: >
```


```
d_01=df.density.quantile(0.01)
d_99=df.density.quantile(0.99)
print(d_01)
print(d_99)
```

```
0.9932314999999999
1.0
```

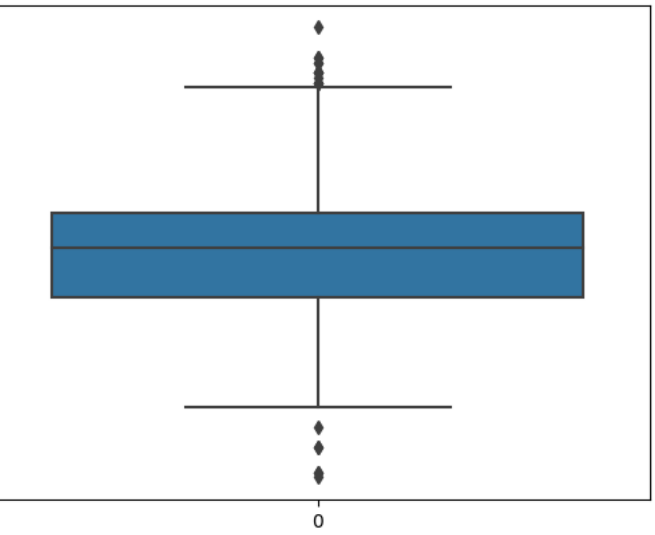

```
df=df[(df.density>=d_01) & (df.density<=d_99)]
sns.boxplot(df.density)
```

```
<Axes: >
```




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

```
<Axes: >
```



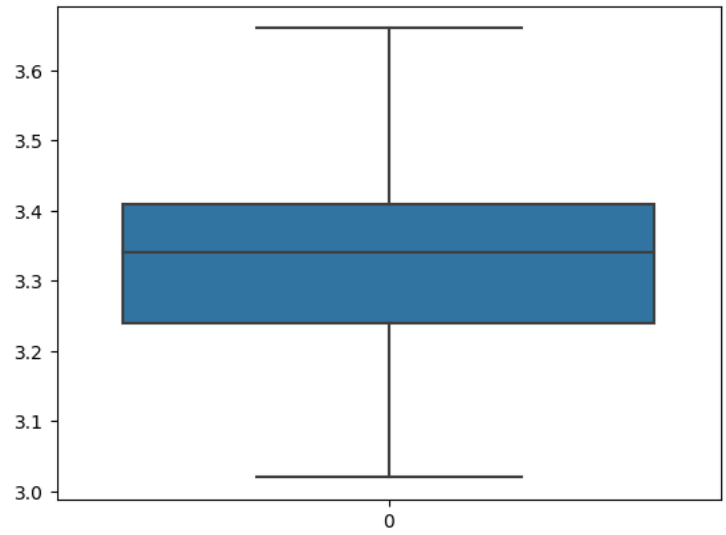

```
pH1 = df.pH.quantile(0.25) #Q1
pH3 = df.pH.quantile(0.75) #Q3
IQR_pH = pH3 - pH1
upper_limit_pH = pH3+(1.5)*(IQR_pH)
lower_limit_pH = pH1-(1.5)*(IQR_pH)
print(pH1)
print(pH3)
print(IQR_pH)
print(upper_limit_pH)
print(lower_limit_pH)
```

```
3.24
3.41
0.16999999999999993
```

```
3.665
2.9850000000000003
```

```
df=df[(df.pH<upper_limit_pH) & (df.pH>lower_limit_pH)]
sns.boxplot(df.pH)
```

```
<Axes: >
```

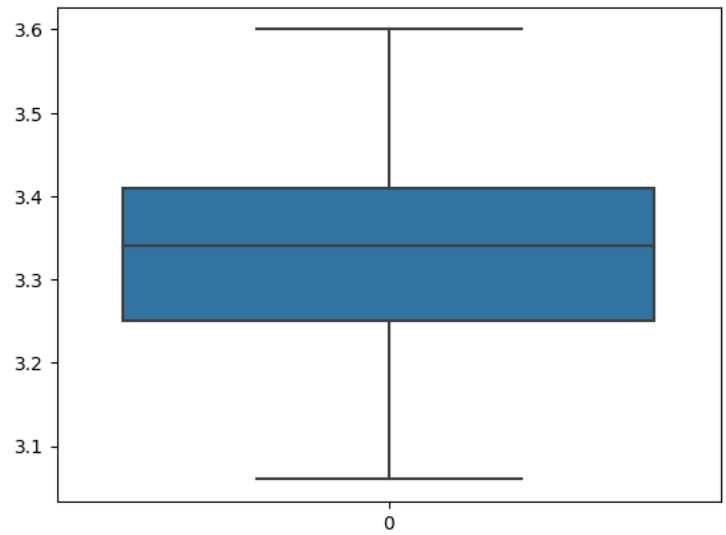



```
pH_01=df.pH.quantile(0.01)
pH_99=df.pH.quantile(0.99)
print(pH_01)
print(pH_99)
```

```
3.06
3.6034
```

```
df=df[(df.pH>=pH_01) & (df.pH<=pH_99)]
sns.boxplot(df.pH)
```

```
<Axes: >
```



ALL THE OUTLIERS ARE REMOVED

DATA PREPROCESSING

```
X=df.iloc[:,:-1]
X.head()
```

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_d
0	7.4	0.70	0.00	1.9	0.076	11.0	
1	7.8	0.88	0.00	2.6	0.098	25.0	

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

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

```
Y = df['quality'].apply(lambda y_value: 1 if y_value>=7 else 0)
print(Y)
```

```
0       0
1       0
2       0
3       0
4       0
       ..
1593    0
1594    0
1595    0
1596    0
1597    0
Name: quality, Length: 948, dtype: int64
```

Train and Test split

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=3)
```

```
X_train.shape
```

```
(758, 11)
```

```
X_test.shape
```

```
(190, 11)
```

```
Y_train.shape
```

```
(758,)
```

```
Y_test.shape
```

```
(190,)
```

Model building

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

```
model.fit(X_train,Y_train)
```

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

Model Evaluation

```
X_test_prediction = model.predict(X_test)
X_train_prediction=model.predict(X_train)
```

```
print('Testing Accuracy = ', accuracy_score(Y_test,X_test_prediction))
print('Training Accuracy = ', accuracy_score(Y_train,X_train_prediction))
```

```
Testing Accuracy =  0.9263157894736842
Training Accuracy =  1.0
```

```
input_data = [7.6, 1.0, 0, 3.0, 0.07, 30, 100, 0.9542, 3.1, 0.66, 9.6]
prediction = model.predict([input_data])
prediction
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestClas
  warnings.warn(
array([0])
```

✓ 0s completed at 10:56 PM