Load the Dataset

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

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

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	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

Data preprocessing including visualization

```
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.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
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

```
df.describe()
```

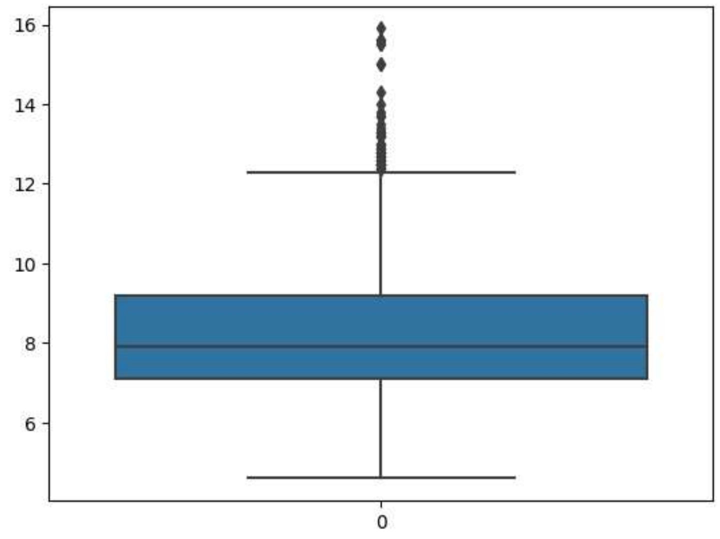
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	su
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	159
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2

```
df.shape
```

```
(1599, 12)
```

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

```
<Axes: >
```

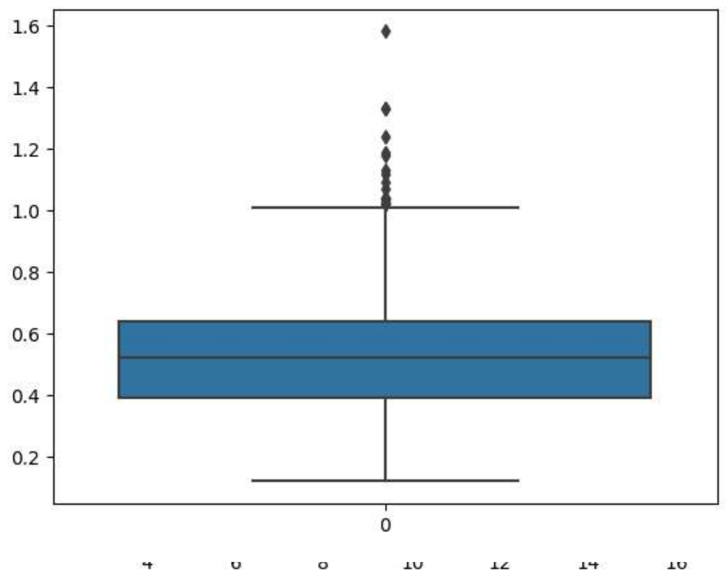


```
sns.distplot(df['fixed acidity'])
```

```
<ipython-input-12-52a4a49dcd39>: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).
```

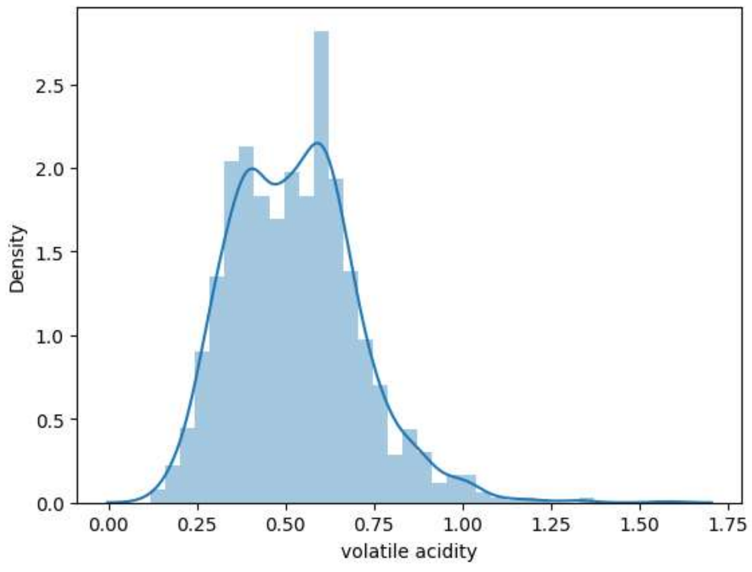
```
sns.boxplot(df['volatile acidity'])
```

```
<Axes: >
```



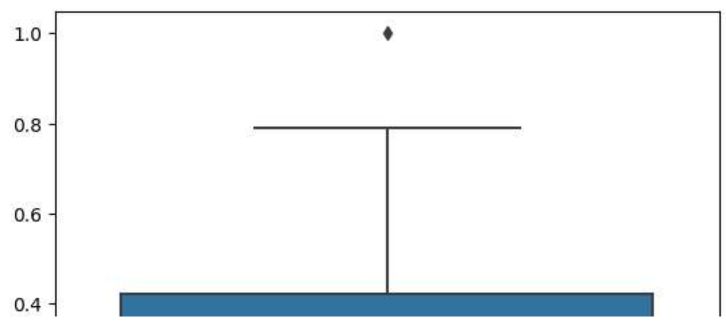
```
sns.distplot(df['volatile acidity'])
```

```
<ipython-input-14-6077730c287e>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['volatile acidity'])
<Axes: xlabel='volatile acidity', ylabel='Density'>
```



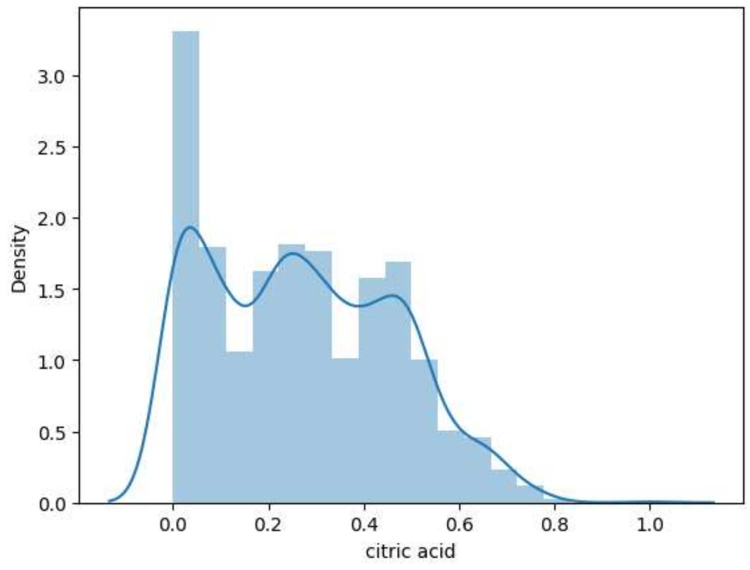
```
sns.boxplot(df['citric acid'])
```

```
<Axes: >
```



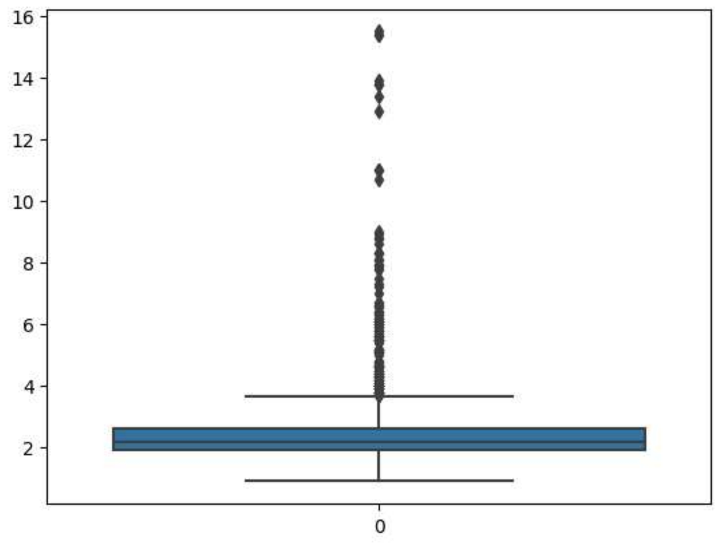
```
sns.distplot(df['citric acid'])
```

```
<ipython-input-16-1324198882c2>: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['citric acid'])
<Axes: xlabel='citric acid', ylabel='Density'>
```



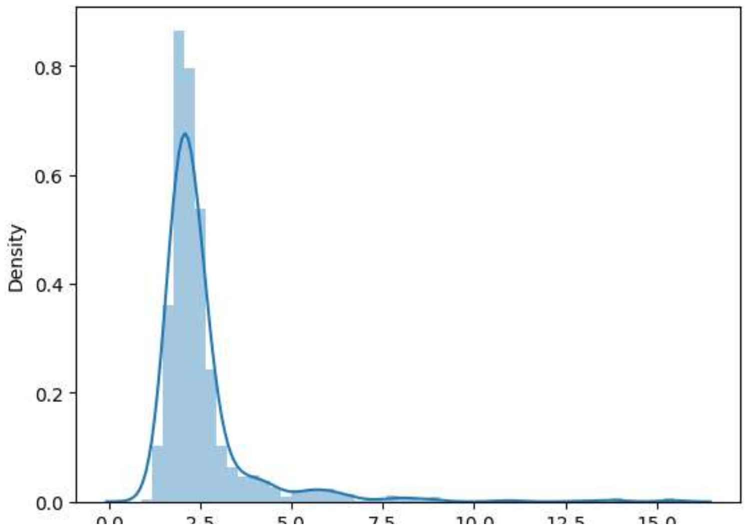
```
sns.boxplot(df['residual sugar'])
```

```
<Axes: >
```



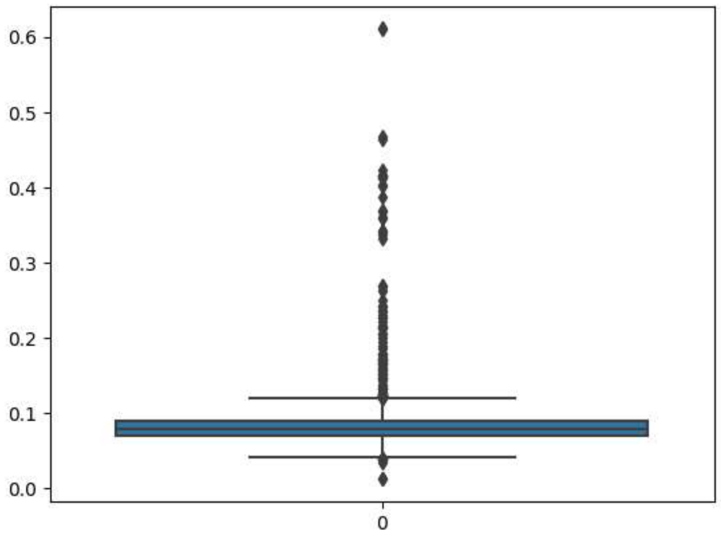
```
sns.distplot(df['residual sugar'])
```

```
<ipython-input-18-17c4014efccf>: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'])
<Axes: xlabel='residual sugar', ylabel='Density'>
```



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

```
<Axes: >
```

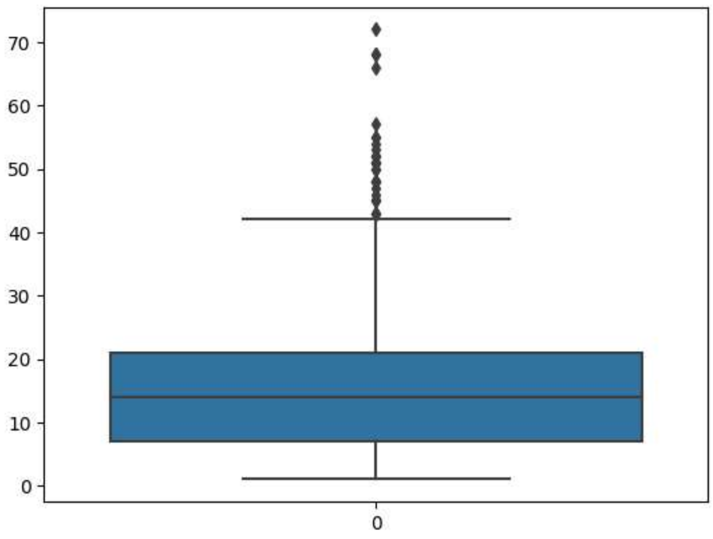


```
sns.distplot(df['chlorides'])
```

```
<ipython-input-20-fdc4bb1ed131>: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['chlorides'])
<Axes: xlabel='chlorides', ylabel='Density'>
```

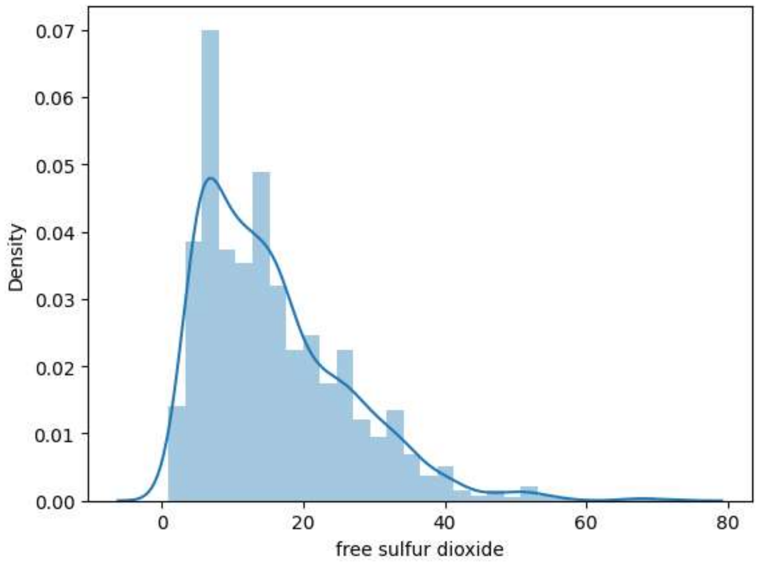
```
sns.boxplot(df['free sulfur dioxide'])
```

```
<Axes: >
```



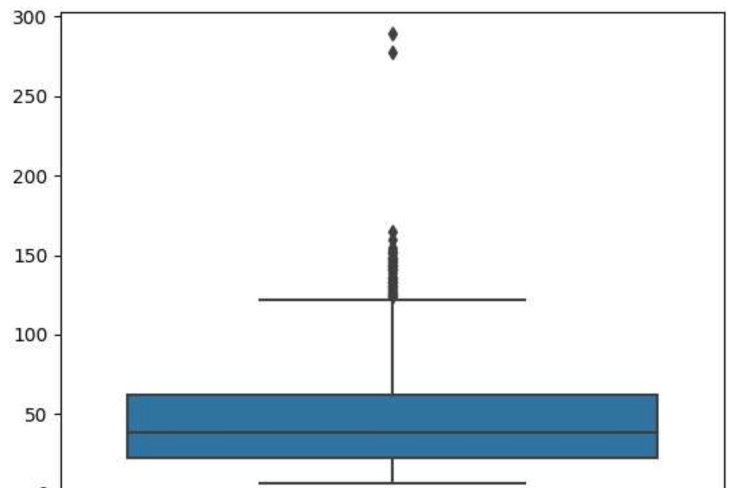
```
sns.distplot(df['free sulfur dioxide'])
```

```
<ipython-input-24-3dee0624d434>: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'])
<Axes: xlabel='free sulfur dioxide', ylabel='Density'>
```



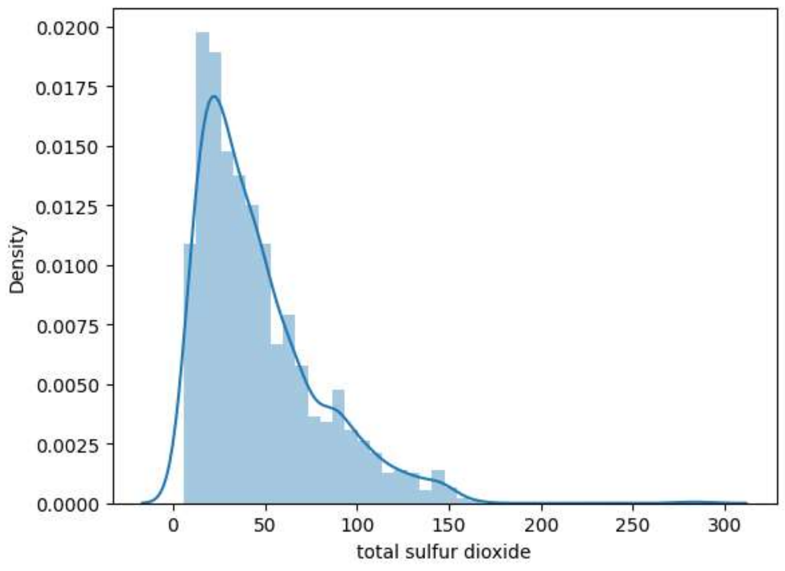
```
sns.boxplot(df['total sulfur dioxide'])
```

```
<Axes: >
```



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

```
<ipython-input-26-a53ba4eac084>: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'])
<Axes: xlabel='total sulfur dioxide', ylabel='Density'>
```

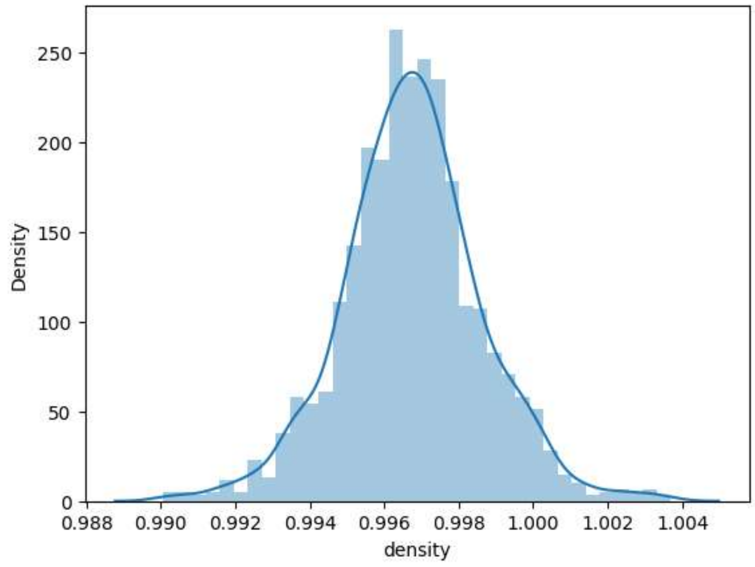


```
sns.boxplot(df['density'])
```

```
<Axes: >
```

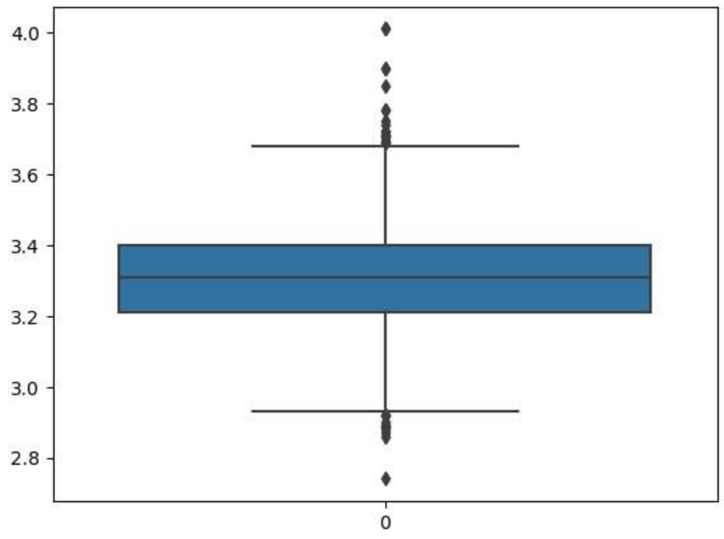
```
sns.distplot(df['density'])
```

```
<ipython-input-28-cffea316cede>: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['density'])
<Axes: xlabel='density', ylabel='Density'>
```



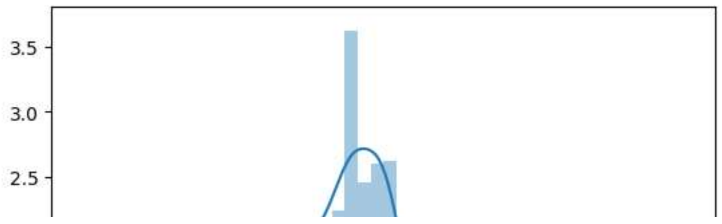
```
sns.boxplot(df['pH'])
```

```
<Axes: >
```



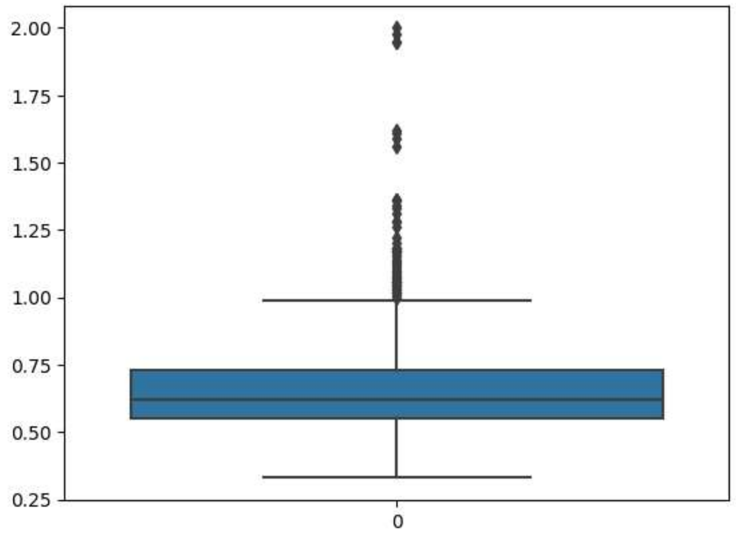
```
sns.distplot(df['pH'])
```

```
<ipython-input-30-d020e64af2d2>: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['pH'])
<Axes: xlabel='pH', ylabel='Density'>
```



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

```
<Axes: >
```

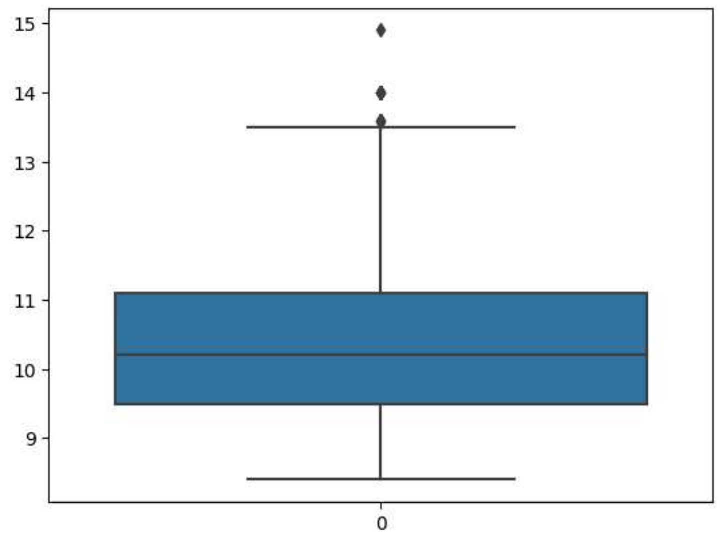


```
sns.distplot(df['sulphates'])
```

```
<ipython-input-32-3a090c5692ad>:1: UserWarning:
```

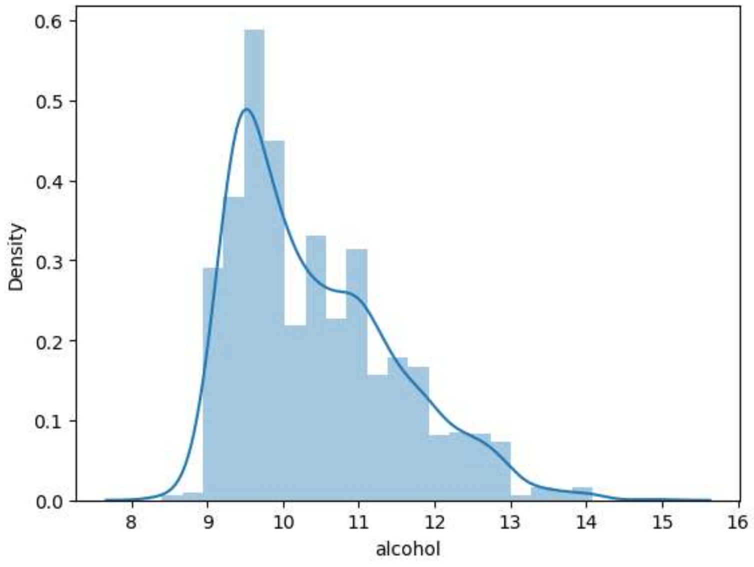
```
sns.boxplot(df['alcohol'])
```

```
<Axes: >
```



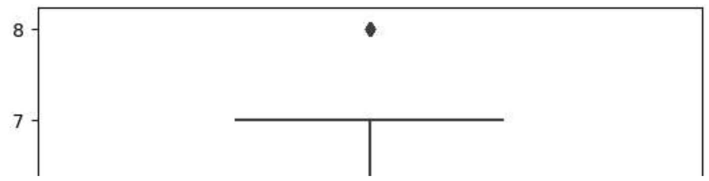
```
sns.distplot(df['alcohol'])
```

```
<ipython-input-34-570de8ff0310>: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'>
```



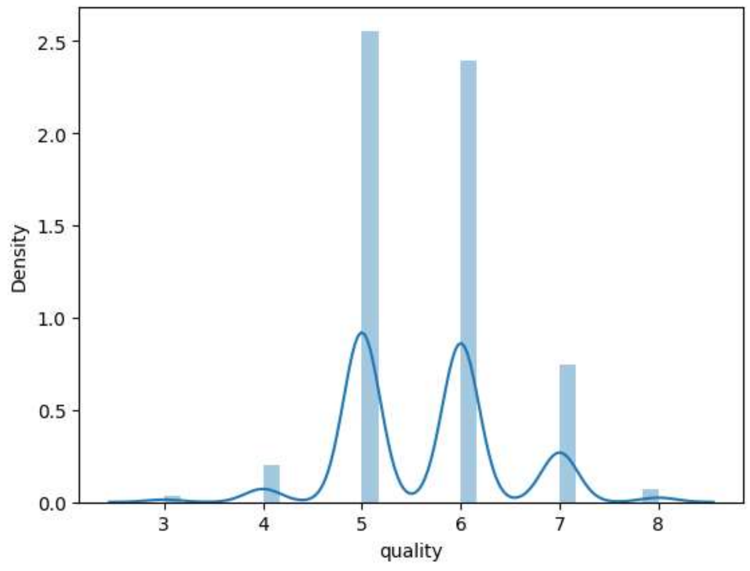
```
sns.boxplot(df['quality'])
```

```
<Axes: >
```



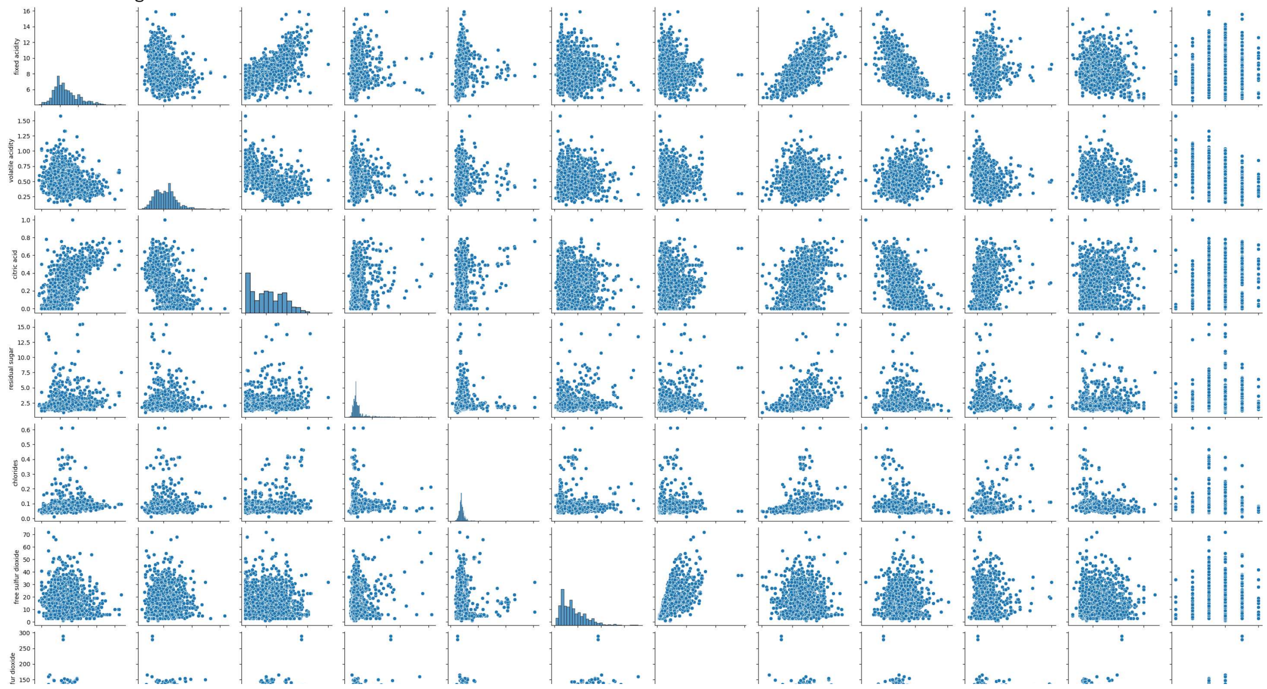
```
sns.distplot(df['quality'])
```

```
<ipython-input-36-e9b2f3ff6ab5>: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['quality'])
<Axes: xlabel='quality', ylabel='Density'>
```



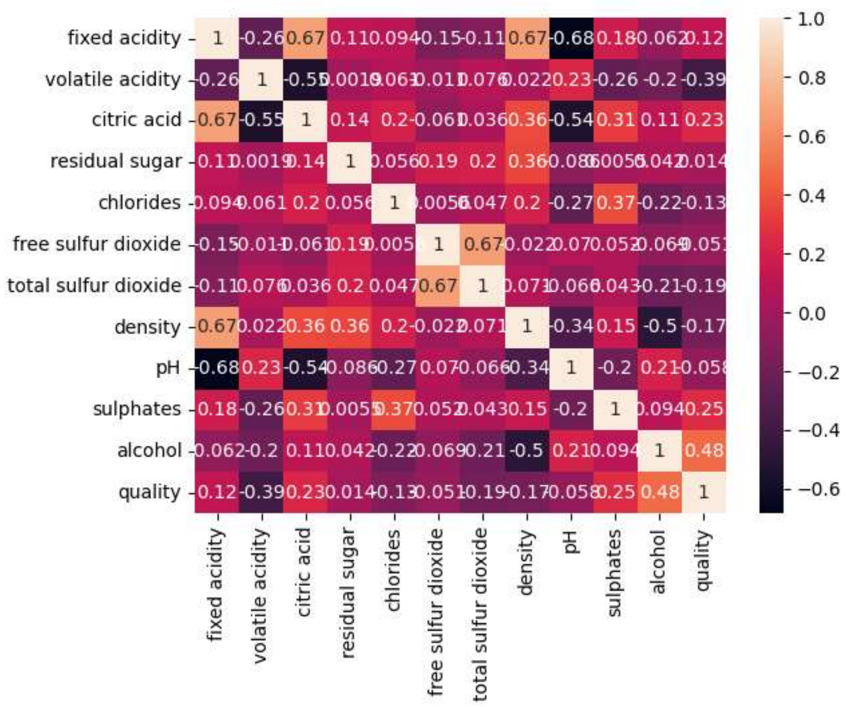
```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7eac54bd0f40>
```



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

```
<Axes: >
```



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

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

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

```
0    5
1    5
2    5
```

```
3    6
4    5
Name: quality, dtype: int64
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train, y_test = train_test_split(X,y, test_size= 0.2, random_state = 0)
```

```
X_train.shape
```

```
(1279, 11)
```

```
X_test.shape
```

```
(320, 11)
```

```
y_train.shape
```

```
(1279,)
```

```
y_test.shape
```

```
(320,)
```

Machine Learning Model building

```
from sklearn.linear_model import LinearRegression, LogisticRegression
lr = LinearRegression()
lor = LogisticRegression()
```

```
lr.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
lor.fit(X_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Conver
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

```
y_predict_lr = np.round(lr.predict(X_test))
y_predict_lr
```

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

```
y_predict_lor = np.round(lor.predict(X_test))
y_predict_lor
```

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

Evaluate the model

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

```
as1 = accuracy_score(y_test, y_predict_lr)
as1
```

```
    0.63125
```

```
as2 = accuracy_score(y_test, y_predict_lor)
as2
```

```
    0.625
```

```
r2s1 = metrics.r2_score(y_test,y_predict_lr)
r2s1
```

```
    0.20846127601501196
```

```
r2s2 = metrics.r2_score(y_test, y_predict_lor)
r2s2
```

```
    0.1647901740020471
```

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

col_0	5.0	6.0	7.0
quality			
3	2	0	0
4	6	5	0
5	97	38	0
6	37	98	7
7	1	19	7
8	0	1	2

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

col_0	5	6	7
quality			
3	2	0	0
4	7	4	0
5	101	33	1
6	42	96	4
7	2	22	3
8	0	2	1

```
classification_report(y_test, y_predict_lr)
```

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

```
classification_report(y_test,y_predict_lor)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-sc
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-sc
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-sc
  _warn_prf(average, modifier, msg_start, len(result))
'              precision    recall  f1-score   support\n\n           3       0.00      0.00      0.00         2\n
4       0.00      0.00      0.00        11\n           5       0.66      0.75      0.70       135\n           6       0.61
0.68      0.64       142\n           7       0.33      0.11      0.17        27\n           8       0.00      0.00      0.
00         3\n\n    accuracy                           0.62       320\n   macro avg       0.27      0.26      0.25       3
20\nweighted avg       0.58      0.62      0.59       320\n'
```

Test with random observation

```
random_input_1 = np.round(lr.predict([[9.0,0.60,0.05,1.8,0.087,11.0,25.0,0.8888,8.90,0.90,7.4]]))
random_input_1
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
array([7.])
```

```
random_input_2 = np.round(lr.predict([[9.4,0.90,0.10,3.0,0.155,12.0,90.0,0.8888,2.50,0.68,9.6]]))
random_input_2
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
array([8.])
```