In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [44]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\11_winequality-red.csv")
a
```

Out[44]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	qua
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	
...	...	...	...	...	...	...	...	...	...	...	...	
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	

In [45]:
```
a=a.head(50)
a
```

Out[45]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
5	7.4	0.660	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4	5
6	7.9	0.600	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4	5
7	7.3	0.650	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0	7
8	7.8	0.580	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5	7
9	7.5	0.500	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	5
10	6.7	0.580	0.08	1.8	0.097	15.0	65.0	0.9959	3.28	0.54	9.2	5
11	7.5	0.500	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	5
12	5.6	0.615	0.00	1.6	0.089	16.0	59.0	0.9943	3.58	0.52	9.9	5
13	7.8	0.610	0.29	1.6	0.114	9.0	29.0	0.9974	3.26	1.56	9.1	5
14	8.9	0.620	0.18	3.8	0.176	52.0	145.0	0.9986	3.16	0.88	9.2	5
15	8.9	0.620	0.19	3.9	0.170	51.0	148.0	0.9986	3.17	0.93	9.2	5
16	8.5	0.280	0.56	1.8	0.092	35.0	103.0	0.9969	3.30	0.75	10.5	7
17	8.1	0.560	0.28	1.7	0.368	16.0	56.0	0.9968	3.11	1.28	9.3	5
18	7.4	0.590	0.08	4.4	0.086	6.0	29.0	0.9974	3.38	0.50	9.0	4
19	7.9	0.320	0.51	1.8	0.341	17.0	56.0	0.9969	3.04	1.08	9.2	6
20	8.9	0.220	0.48	1.8	0.077	29.0	60.0	0.9968	3.39	0.53	9.4	6
21	7.6	0.390	0.31	2.3	0.082	23.0	71.0	0.9982	3.52	0.65	9.7	5
22	7.9	0.430	0.21	1.6	0.106	10.0	37.0	0.9966	3.17	0.91	9.5	5
23	8.5	0.490	0.11	2.3	0.084	9.0	67.0	0.9968	3.17	0.53	9.4	5
24	6.9	0.400	0.14	2.4	0.085	21.0	40.0	0.9968	3.43	0.63	9.7	6
25	6.3	0.390	0.16	1.4	0.080	11.0	23.0	0.9955	3.34	0.56	9.3	5
26	7.6	0.410	0.24	1.8	0.080	4.0	11.0	0.9962	3.28	0.59	9.5	5
27	7.9	0.430	0.21	1.6	0.106	10.0	37.0	0.9966	3.17	0.91	9.5	5
28	7.1	0.710	0.00	1.9	0.080	14.0	35.0	0.9972	3.47	0.55	9.4	5
29	7.8	0.645	0.00	2.0	0.082	8.0	16.0	0.9964	3.38	0.59	9.8	6
30	6.7	0.675	0.07	2.4	0.089	17.0	82.0	0.9958	3.35	0.54	10.1	5
31	6.9	0.685	0.00	2.5	0.105	22.0	37.0	0.9966	3.46	0.57	10.6	6
32	8.3	0.655	0.12	2.3	0.083	15.0	113.0	0.9966	3.17	0.66	9.8	5
33	6.9	0.605	0.12	10.7	0.073	40.0	83.0	0.9993	3.45	0.52	9.4	6
34	5.2	0.320	0.25	1.8	0.103	13.0	50.0	0.9957	3.38	0.55	9.2	5
35	7.8	0.645	0.00	5.5	0.086	5.0	18.0	0.9986	3.40	0.55	9.6	6
36	7.8	0.600	0.14	2.4	0.086	3.0	15.0	0.9975	3.42	0.60	10.8	6

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
37	8.1	0.380	0.28	2.1	0.066	13.0	30.0	0.9968	3.23	0.73	9.7	7
38	5.7	1.130	0.09	1.5	0.172	7.0	19.0	0.9940	3.50	0.48	9.8	4
39	7.3	0.450	0.36	5.9	0.074	12.0	87.0	0.9978	3.33	0.83	10.5	5
40	7.3	0.450	0.36	5.9	0.074	12.0	87.0	0.9978	3.33	0.83	10.5	5
41	8.8	0.610	0.30	2.8	0.088	17.0	46.0	0.9976	3.26	0.51	9.3	4
42	7.5	0.490	0.20	2.6	0.332	8.0	14.0	0.9968	3.21	0.90	10.5	6
43	8.1	0.660	0.22	2.2	0.069	9.0	23.0	0.9968	3.30	1.20	10.3	5
44	6.8	0.670	0.02	1.8	0.050	5.0	11.0	0.9962	3.48	0.52	9.5	5
45	4.6	0.520	0.15	2.1	0.054	8.0	65.0	0.9934	3.90	0.56	13.1	4
46	7.7	0.935	0.43	2.2	0.114	22.0	114.0	0.9970	3.25	0.73	9.2	5
47	8.7	0.290	0.52	1.6	0.113	12.0	37.0	0.9969	3.25	0.58	9.5	5
48	6.4	0.400	0.23	1.6	0.066	5.0	12.0	0.9958	3.34	0.56	9.2	5
49	5.6	0.310	0.37	1.4	0.074	12.0	96.0	0.9954	3.32	0.58	9.2	5

In [46]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         50 non-null     float64
 1   volatile acidity      50 non-null     float64
 2   citric acid           50 non-null     float64
 3   residual sugar        50 non-null     float64
 4   chlorides             50 non-null     float64
 5   free sulfur dioxide   50 non-null     float64
 6   total sulfur dioxide  50 non-null     float64
 7   density               50 non-null     float64
 8   pH                    50 non-null     float64
 9   sulphates             50 non-null     float64
 10  alcohol               50 non-null     float64
 11  quality               50 non-null     int64  
dtypes: float64(11), int64(1)
memory usage: 4.8 KB
```

In [47]:
```
a.columns
```

Out[47]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

In [48]:
```
a.head()
```

Out[48]:

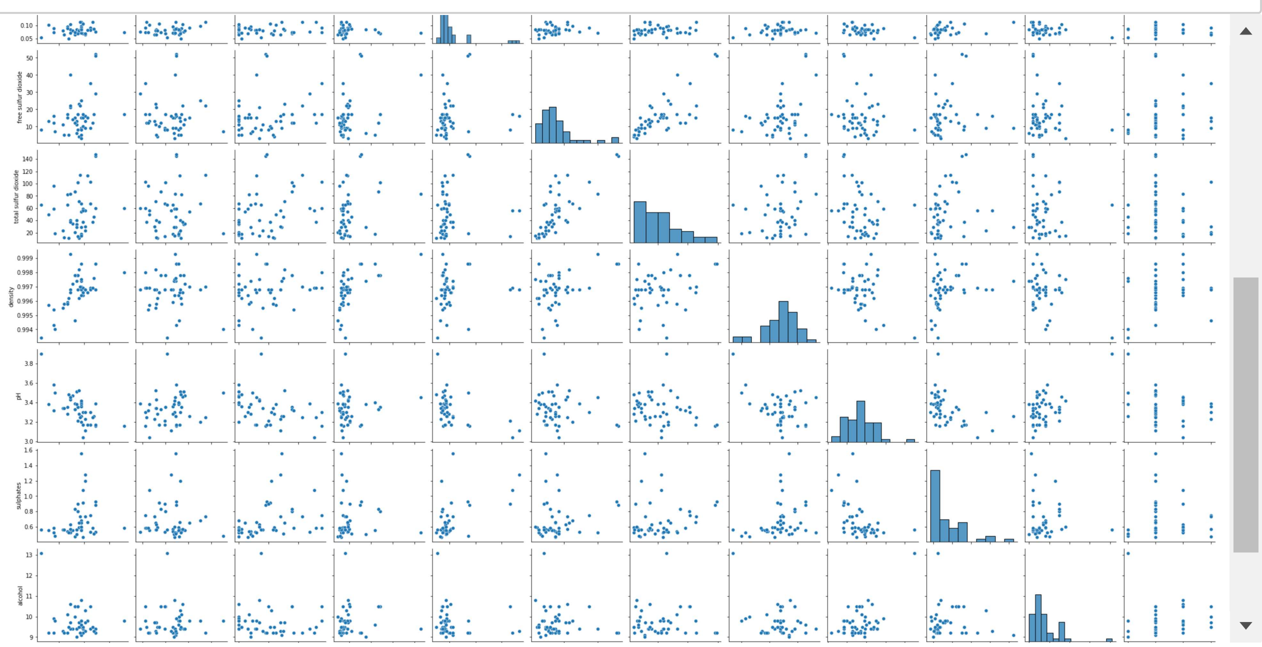
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	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

In [49]:
```
a.describe()
```

Out[49]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH
count	50.000000	50.000000	50.000000	50.000000	50.000000	50.000000	50.000000	50.000000	50.000000
mean	7.510000	0.552000	0.193200	2.644000	0.104140	15.560000	54.340000	0.996818	3.336600
std	1.085385	0.177546	0.167032	1.747203	0.067205	10.441597	34.769568	0.001177	0.146783
min	4.600000	0.220000	0.000000	1.200000	0.050000	3.000000	11.000000	0.993400	3.040000
25%	6.950000	0.415000	0.045000	1.800000	0.074000	9.000000	29.000000	0.996400	3.250000
50%	7.600000	0.585000	0.170000	2.000000	0.083500	13.000000	48.000000	0.996800	3.335000
75%	7.900000	0.653750	0.297500	2.475000	0.101750	17.000000	70.000000	0.997750	3.415000
max	11.200000	1.130000	0.560000	10.700000	0.368000	52.000000	148.000000	0.999300	3.900000

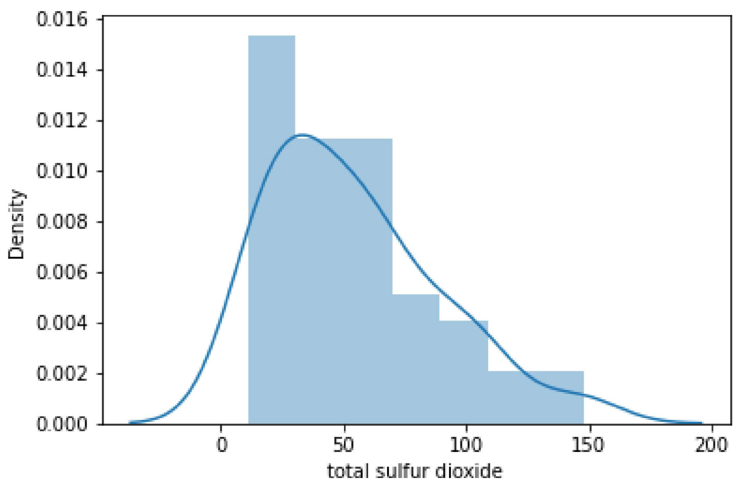
In [50]:
```
sns.pairplot(a)
```



In [51]:
```
sns.distplot(a['total sulfur dioxide'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

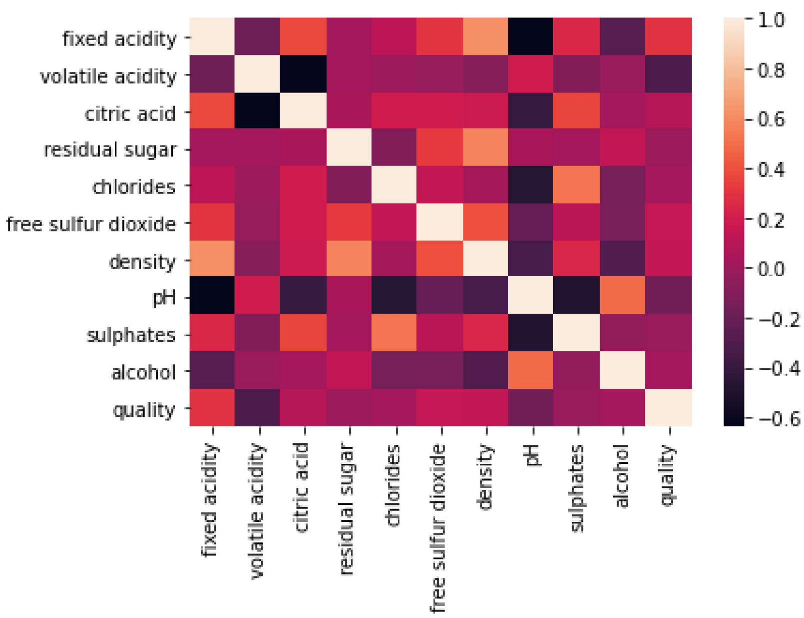
Out[51]: <AxesSubplot:xlabel='total sulfur dioxide', ylabel='Density'>



In [52]:
```
x1=a[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide',  'density',
       'pH', 'sulphates', 'alcohol', 'quality']]
```

In [53]:
```
sns.heatmap(x1.corr())
```

Out[53]: <AxesSubplot:>



In [54]:
```
x=a[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality']]
y=a['total sulfur dioxide']
```

In [55]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [56]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[56]:
```
LinearRegression()
```

In [57]:
```
print(lr.intercept_)
```

```
-4626.874052171854
```

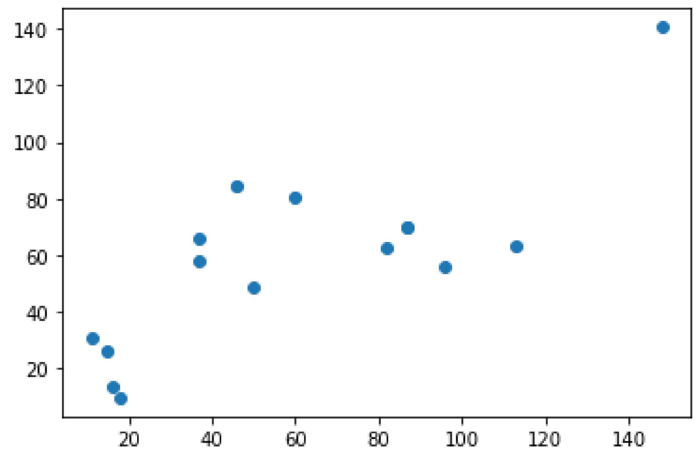
In [58]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[58]:

	Co-efficient
fixed acidity	-4.406215
volatile acidity	19.838820
citric acid	66.739654
residual sugar	-0.976906
chlorides	-55.642452
free sulfur dioxide	2.443872
density	4845.924668
pH	-67.384856
sulphates	-23.638588
alcohol	14.543185
quality	-13.410399

In [59]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[59]: <matplotlib.collections.PathCollection at 0x22eadaa2910>



In [60]:
```
print(lr.score(x_test,y_test))
```

0.618163052939029

In [61]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [62]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[62]: Ridge(alpha=10)

In [63]:
```
rr.score(x_test,y_test)
```

Out[63]: 0.4713149451319629

In [64]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[64]: Lasso(alpha=10)

In [65]:
```
la.score(x_test,y_test)
```

Out[65]: 0.4142017292677008

In [66]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[66]: ElasticNet()

In [67]:
```
print(en.coef_)
```

```
[ 2.01602538 -0.          1.19781769  1.06009876 -0.          2.50526518
 -0.         -0.          0.          4.02656888 -5.49531947]
```

In [68]:
```
print(en.intercept_)
```

```
-15.304938878432829
```

In [69]:
```
print(en.predict(x_test))
```

```
[ 21.16281322  56.61177189  23.00937576  63.81902712 144.33601825
  53.57279247  39.52230902  66.08143049  45.39249435  50.96340153
  82.65133825  29.07083628  24.4600726   50.96340153  37.54315261]
```

In [70]:
```
en.score(x_test,y_test)
```

Out[70]:
```
0.4647540720309572
```

In [71]:
```
from sklearn import metrics
```

In [72]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 20.191042512307103
```

In [73]:
```
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error 592.3665661925082
```

In [74]:
```
print(" Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
 Root Mean Squared Error 24.338581844316817
```