In [45]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [46]:
```
s=pd.read_csv(r"C:\Users\user\Downloads\11_winequality-red - 11_winequality-red.csv")
s
```

Out[46]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	qualit
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	
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	

1599 rows × 12 columns

In [47]:
```
s.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
```

In [48]:
```
s.describe()
```

Out[48]:

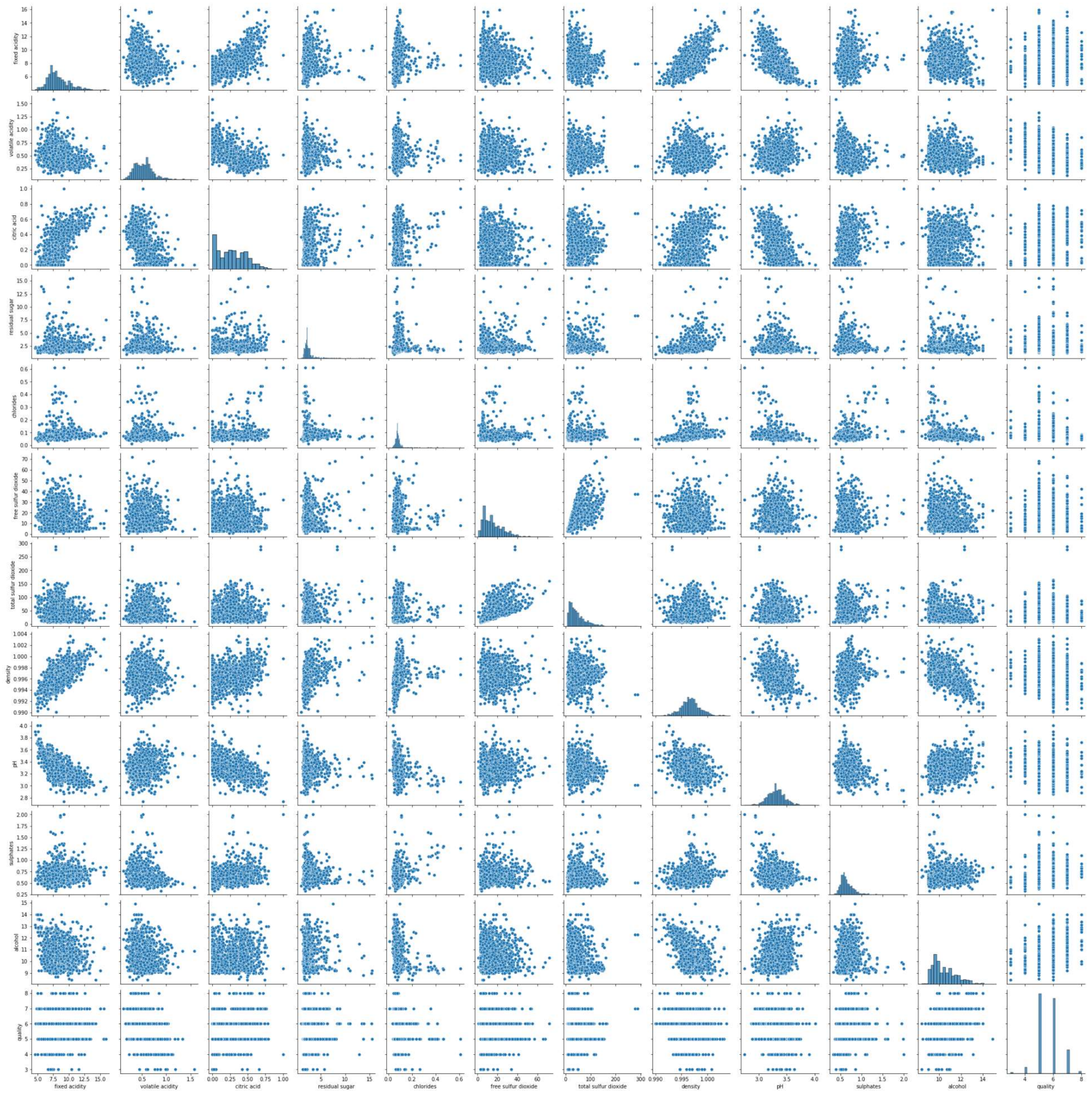
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	de
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.0(
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.9!
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.0(
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.9!
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.9!
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.9!
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.9!
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.0(

In [49]:
```
s.columns
```

Out[49]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

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

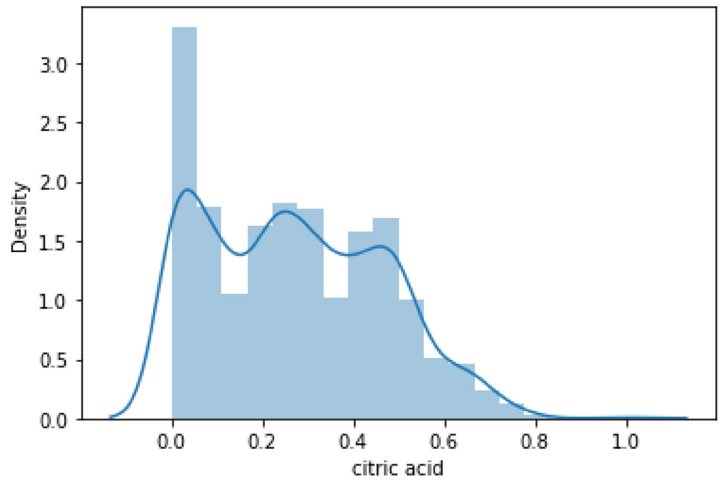
Out[50]:
```
<seaborn.axisgrid.PairGrid at 0x28aeb538880>
```



In [51]:
```
sns.distplot(s['citric acid'])
```

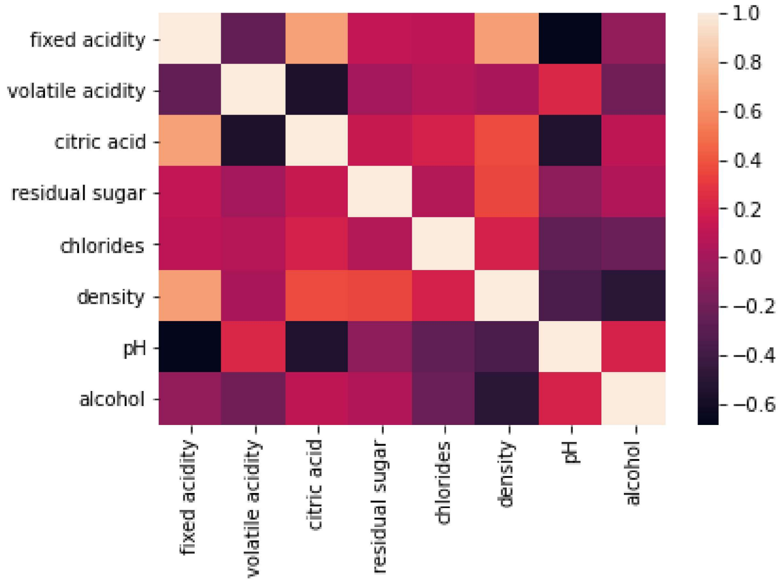
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function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

Out[51]:
```
<AxesSubplot:xlabel='citric acid', ylabel='Density'>
```



In [52]:
```
s1=s[['fixed acidity','volatile acidity','citric acid','residual sugar','chlorides','de
sns.heatmap(s1.corr())
```

Out[52]:
```
<AxesSubplot:>
```



In [53]:
```
x=s1[['fixed acidity','volatile acidity','citric acid','residual sugar','chlorides','de
y=s1['alcohol']
```

In [54]:
```
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 [55]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[55]:
```
LinearRegression()
```

In [56]:
```
lr.intercept_
```

Out[56]:
```
603.4357739265045
```

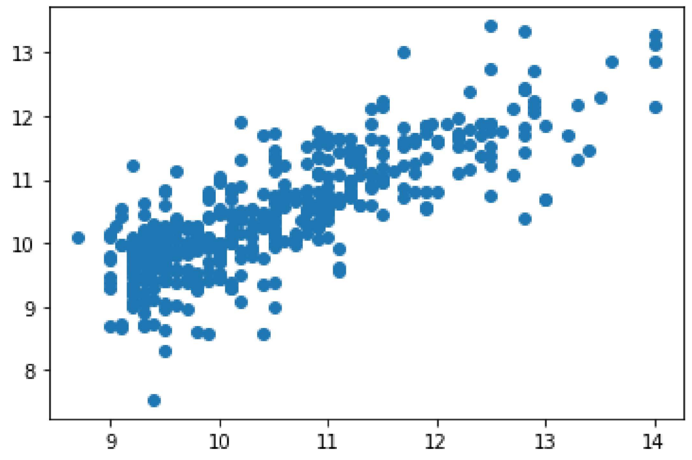
In [57]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[57]:

	Co-efficient
fixed acidity	0.566435
volatile acidity	0.101828
citric acid	0.866467
residual sugar	0.242012
chlorides	-0.260133
density	-614.127347
pH	4.081877

In [58]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[58]:
```
<matplotlib.collections.PathCollection at 0x28af3f7c9a0>
```



In [59]:
```
print(lr.score(x_test,y_test))
```

```
0.6512578308050023
```

In [60]:
```
from sklearn.linear_model import Ridge,Lasso
```

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

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

Out[62]:
```
0.13510342225848648
```

In [63]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_test,y_test)
```

Out[63]:
```
-0.004113723787898316
```

In [64]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[64]:
```
ElasticNet()
```

In [65]:
```
print(en.coef_)
```

```
[-0. -0.  0.  0. -0. -0.  0.]
```

In [66]:
```
print(en.intercept_)
```

```
10.401340482573726
```

```
In [67]: print(en.predict(x_test))
```

```
[10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048
 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048 10.40134048]
```

In [68]:
```
print(en.score(x_test,y_test))
```

```
-0.004113723787898316
```

Evaluation metrics

In [69]:
```
from sklearn import metrics
```

In [70]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 0.5198131321895176
```

In [71]:
```
print("Mean squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean squared Error 0.44066030847307186
```

In [72]:
```
print("Root Mean squared Error",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Root Mean squared Error 0.6638224977153696
```

In [ ]: