In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```

df=pd.read_csv(r"C:\Users\Admin\Downloads\11_winequality-red - 11_winequality-r
df
```

Out[2]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
...	...	...	...	...	...	...	...	...	...	...	
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	1
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	1
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	1
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	1
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	1

1599 rows × 12 columns

In [3]:
```
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
```

In [4]:
```
df.describe()
```

Out[4]:

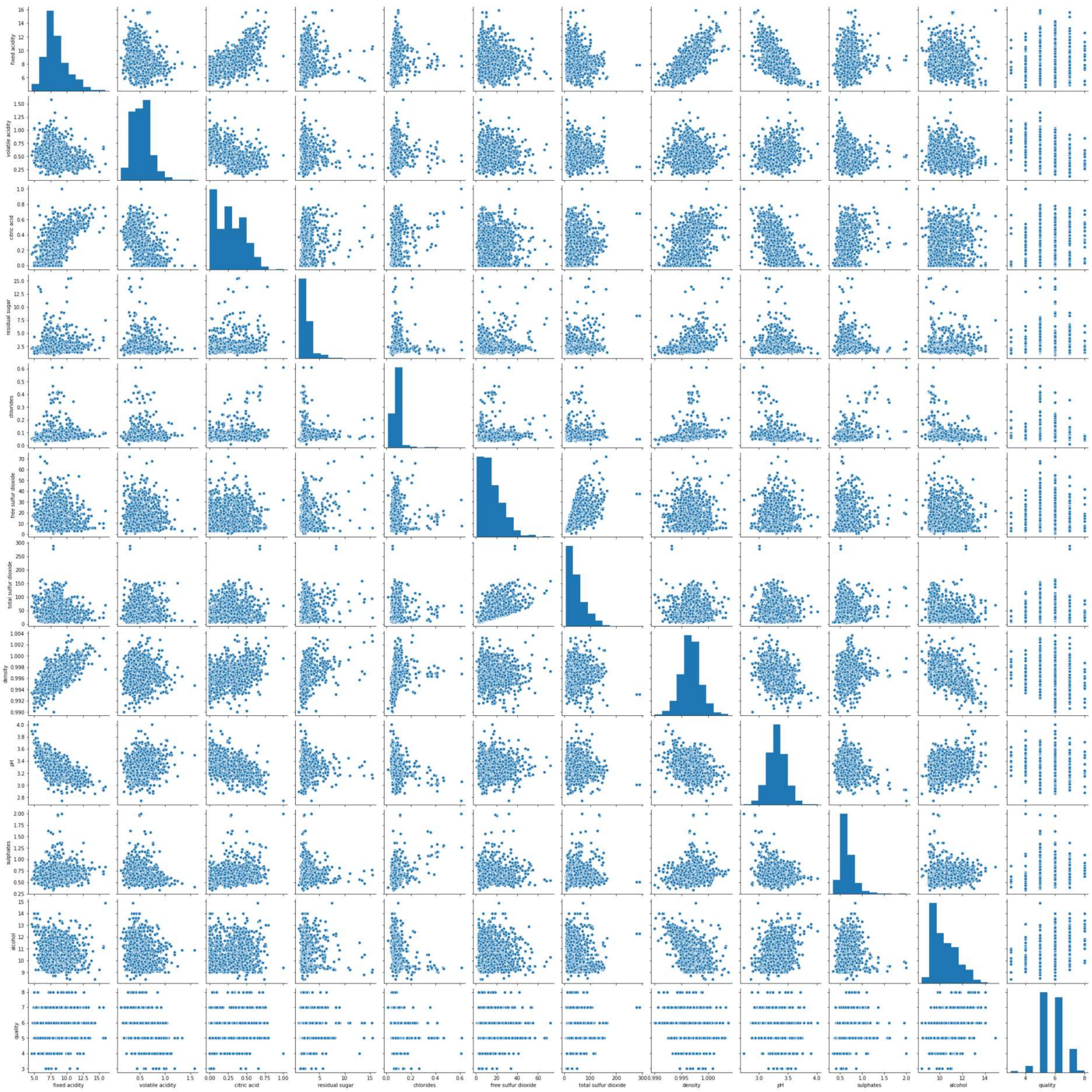
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfu dioxid
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.00000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.4677
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.89532
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.00000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.00000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.00000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.00000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.00000

In [5]:
```
df.columns
```

Out[5]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

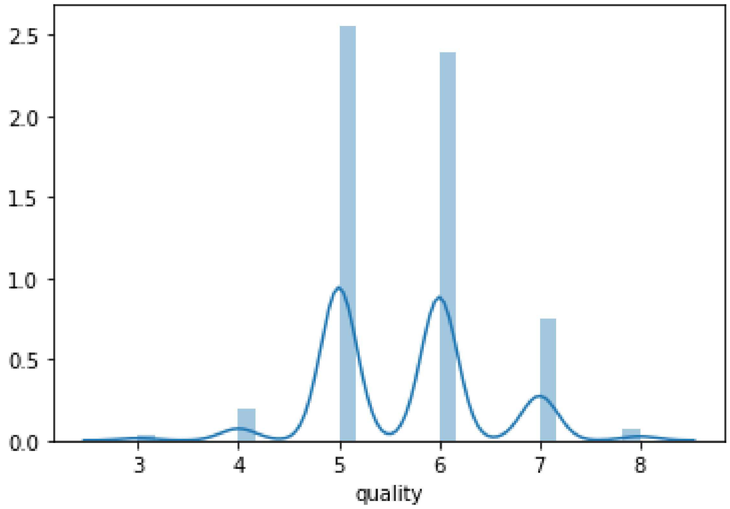
In [6]:
```
sns.pairplot(df)
```

Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x21fb67e2a00>
```



In [7]:
```
sns.distplot(df['quality'])
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x21fbcbd70d0>



In [8]:
```
df1=df[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality']]
df1
```

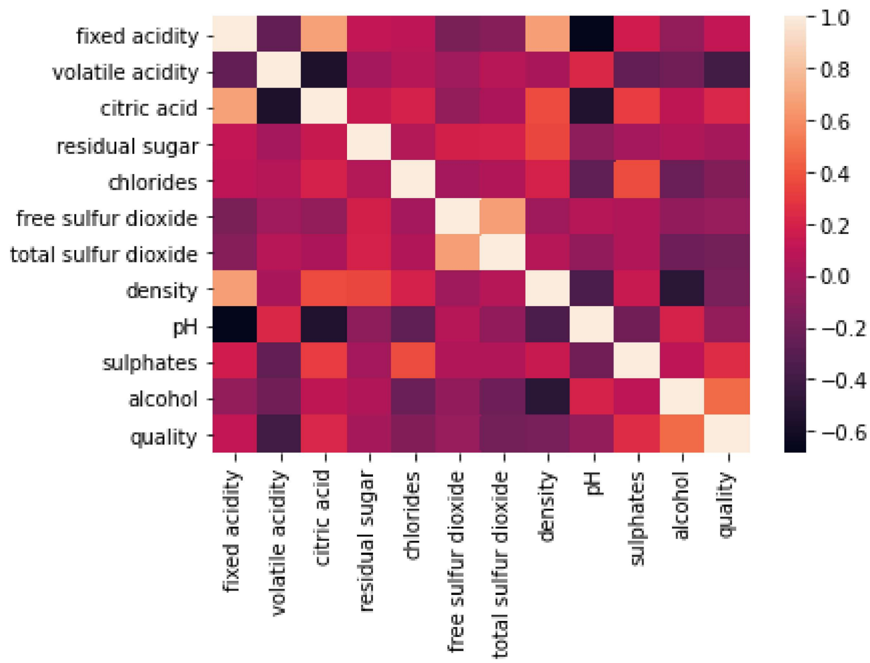
Out[8]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
...	...	...	...	...	...	...	...	...	...	...	
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	1
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	1
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	1
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	1
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	1

1599 rows × 12 columns

In [9]:
```
sns.heatmap(df1.corr())
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x21fbdfe1af0>



In [10]:
```
x=df1[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol']]
y=df[['quality']]
```

In [11]:
```
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 [12]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

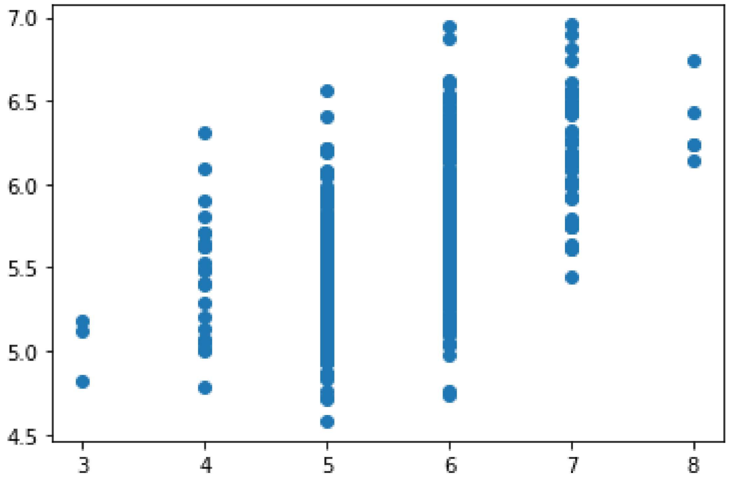
Out[12]: LinearRegression()

In [13]:
```
print(lr.intercept_)
```

[22.61662826]

In [14]:
```
prediction= lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[14]: <matplotlib.collections.PathCollection at 0x21fbe446940>



In [15]:
```
print(lr.score(x_test,y_test))
```

0.33993421063834894

In [16]:
```
print(lr.score(x_train,y_train))
```

0.36613729345061574

In [17]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [18]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[18]: Ridge(alpha=10)

In [19]:
```
rr.score(x_test,y_test)
```

Out[19]: 0.3278779220835216

In [20]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[20]: Lasso(alpha=10)

In [21]:
```
la.score(x_test,y_test)
```

Out[21]: -0.008380755702723786

In [22]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[22]:
```
ElasticNet()
```

In [23]:
```
print(en.intercept_)
```

```
[5.83046024]
```

In [24]:
```
print(en.predict(x_test))
```

```
[5.49960095 5.57395135 5.78213248 5.6371492  5.75610984 5.67804192
 5.6371492  5.7114996  5.6557368  5.40294543 5.77097992 5.7300872
 5.51075351 5.75239232 5.76354488 5.7672624  5.72636968 5.6371492
 5.60740904 5.58510392 5.77841496 5.38435783 5.69662952 5.6557368
 5.52934111 5.692912   5.7672624  5.59625648 5.75982736 5.34346511
 5.58882144 5.77841496 5.77097992 5.71521712 5.68175944 5.77097992
 5.72636968 5.57766888 5.78213248 5.6557368  5.55536375 5.68919448
 5.55908127 5.57395135 5.46614327 5.74123976 5.6185616  5.71521712
 5.72636968 5.7486748  5.78956752 5.7672624  5.78585    5.75610984
 5.58510392 5.68919448 5.7672624  5.74495728 5.76354488 5.51075351
 5.70778208 5.44012063 5.79328504 5.52934111 5.53677615 5.78956752
 5.72636968 5.58882144 5.64086672 5.75239232 5.74495728 5.72636968
 5.7486748  5.77469744 5.6743244  5.75610984 5.51447103 5.77841496
 5.43640311 5.71521712 5.75239232 5.49960095 5.72636968 5.78213248
 5.72265216 5.66688936 5.692912   5.7300872  5.75982736 5.61484408
 5.66317184 5.75239232 5.41781551 5.6743244  5.78213248 5.54421119
 5.51818855 5.77097992 5.72636968 5.599974   5.64830176 5.66317184
 5.45870823 5.60740904 5.74495728 5.56279879 5.80072008 5.62971416
 5.5813864  5.78956752 5.50703599 5.66688936 5.47357831 5.78585
 5.7672624  5.62227912 5.77097992 5.77469744 5.64830176 5.62599664
 5.67804192 5.74123976 5.76354488 5.80072008 5.78956752 5.72636968
 5.65201928 5.71521712 5.68919448 5.7486748  5.7300872  5.7672624
 5.62971416 5.76354488 5.67804192 5.36577023 5.7672624  5.7672624
 5.69662952 5.77469744 5.79700256 5.68919448 5.76354488 5.51818855
 5.73380472 5.57766888 5.62971416 5.50331847 5.56279879 5.58510392
 5.76354488 5.7486748  5.26539719 5.74495728 5.65945432 5.70406456
 5.49216591 5.52562359 5.7486748  5.70406456 5.74123976 5.7486748
 5.38807535 5.7114996  5.66688936 5.79328504 5.73380472 5.70034704
 5.64830176 5.74123976 5.59253896 5.48844839 5.70778208 5.72636968
 5.692912   5.76354488 5.57766888 5.74123976 5.79328504 5.74495728
 5.60740904 5.77097992 5.64830176 5.8044376  5.72265216 5.74123976
 5.72636968 5.61484408 5.63343168 5.6743244  5.7114996  5.70406456
 5.7300872  5.70034704 5.73380472 5.7114996  5.77469744 5.692912
 5.72265216 5.77469744 5.62971416 5.62971416 5.25424463 5.68547696
 5.68175944 5.64086672 5.7300872  5.76354488 5.50703599 5.77841496
 5.72636968 5.51447103 5.36948775 5.78585    5.47729583 5.692912
 5.75610984 5.49216591 5.7300872  5.57766888 5.77841496 5.73380472
 5.80072008 5.34346511 5.66688936 5.76354488 5.72265216 5.63343168
 5.77097992 5.45127319 5.77469744 5.62971416 5.75982736 5.55536375
 5.55164623 5.37692279 5.61112656 5.28398479 5.65201928 5.68919448
 5.78213248 5.33231255 5.72636968 5.46242575 5.692912   5.60740904
 5.72265216 5.62227912 5.71521712 5.7114996  5.70034704 5.58882144
 5.57395135 5.72636968 5.48101335 5.60369152 5.74123976 5.70034704
 5.68175944 5.77469744 5.40294543 5.7114996  5.42896807 5.73752224
 5.54792871 5.74495728 5.70406456 5.29141983 5.7672624  5.66317184
 5.77469744 5.64086672 5.48101335 5.70778208 5.55908127 5.67060688
 5.73752224 5.6743244  5.7672624  5.57023383 5.78213248 5.47729583
 5.6557368  5.62971416 5.55536375 5.63343168 5.40666295 5.75610984
 5.7114996  5.77841496 5.60740904 5.70034704 5.64086672 5.68175944
 5.33603007 5.7114996  5.48844839 5.56651631 5.58882144 5.73752224
 5.77469744 5.78585    5.68919448 5.35090015 5.75239232 5.73752224
 5.59625648 5.7486748  5.50331847 5.61484408 5.75610984 5.71893464
 5.77841496 5.64086672 5.75982736 5.57023383 5.73380472 5.692912
 5.6743244  5.21706943 5.692912   5.70778208 5.75982736 5.80072008
 5.79328504 5.45499071 5.71521712 5.73752224 5.75982736 5.6557368
 5.67060688 5.7300872  5.7300872  5.54792871 5.5813864  5.64830176
 5.54049367 5.64830176 5.59625648 5.46242575 5.54421119 5.76354488
```

```
 5.55908127 5.76354488 5.29513735 5.7300872  5.32487751 5.56279879
 5.75239232 5.7114996  5.68547696 5.44383815 5.42153303 5.77469744
 5.79328504 5.61112656 5.50331847 5.6557368  5.60740904 5.55536375
 5.74495728 5.65201928 5.70034704 5.33231255 5.66317184 5.79328504
 5.692912   5.68919448 5.62599664 5.61112656 5.77841496 5.74123976
 5.63343168 5.7114996  5.62227912 5.56651631 5.76354488 5.28398479
 5.6557368  5.6743244  5.38807535 5.64830176 5.75982736 5.7486748
 5.73752224 5.32859503 5.78585    5.79328504 5.79328504 5.54049367
 5.66688936 5.66317184 5.80072008 5.60740904 5.60369152 5.78956752
 5.7114996  5.73380472 5.68919448 5.54421119 5.74123976 5.51818855
 5.50703599 5.77469744 5.51447103 5.75239232 5.58882144 5.692912
 5.76354488 5.76354488 5.44012063 5.75610984 5.78585    5.58882144
 5.72265216 5.46242575 5.34718263 5.73380472 5.64830176 5.79328504
 5.64458424 5.46242575 5.60369152 5.74495728 5.46614327 5.77841496
 5.60740904 5.64830176 5.6743244  5.7672624  5.60369152 5.46614327
 5.57023383 5.64830176 5.65201928 5.64830176 5.70034704 5.79700256
 5.76354488 5.75239232 5.75239232 5.71521712 5.71893464 5.77841496
 5.7486748  5.64830176 5.48844839 5.59625648 5.57766888 5.80072008
 5.5813864  5.52934111 5.51447103 5.70778208 5.75982736 5.73752224
 5.77469744 5.7486748  5.70406456 5.80072008 5.77097992 5.65945432
 5.56651631 5.32859503 5.76354488 5.67060688 5.50331847 5.48101335
 5.53305863 5.66688936 5.58882144 5.70034704 5.77097992 5.78585
 5.57023383 5.71893464 5.64458424 5.66317184 5.692912   5.66317184]
```

In [25]:
```
print(en.score(x_test,y_test))
```

```
0.033163424256753005
```

Evaluation

In [26]:
```
from sklearn import metrics
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Absolute Error 0.5178352605495445
Mean Squared Error 0.4464611658598944
```

In [27]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prec
```

```
Root Mean Squared Error: 0.6681774957748087
```

In [ ]: