D7

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

```
In [2]: df=pd.read_csv(r"C:\Users\user\Downloads\11_winequality-red.csv")
        df
```

Out[2]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alco
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.head(10)`

Out[3]:

	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
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5
9	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5

In [4]: `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
```
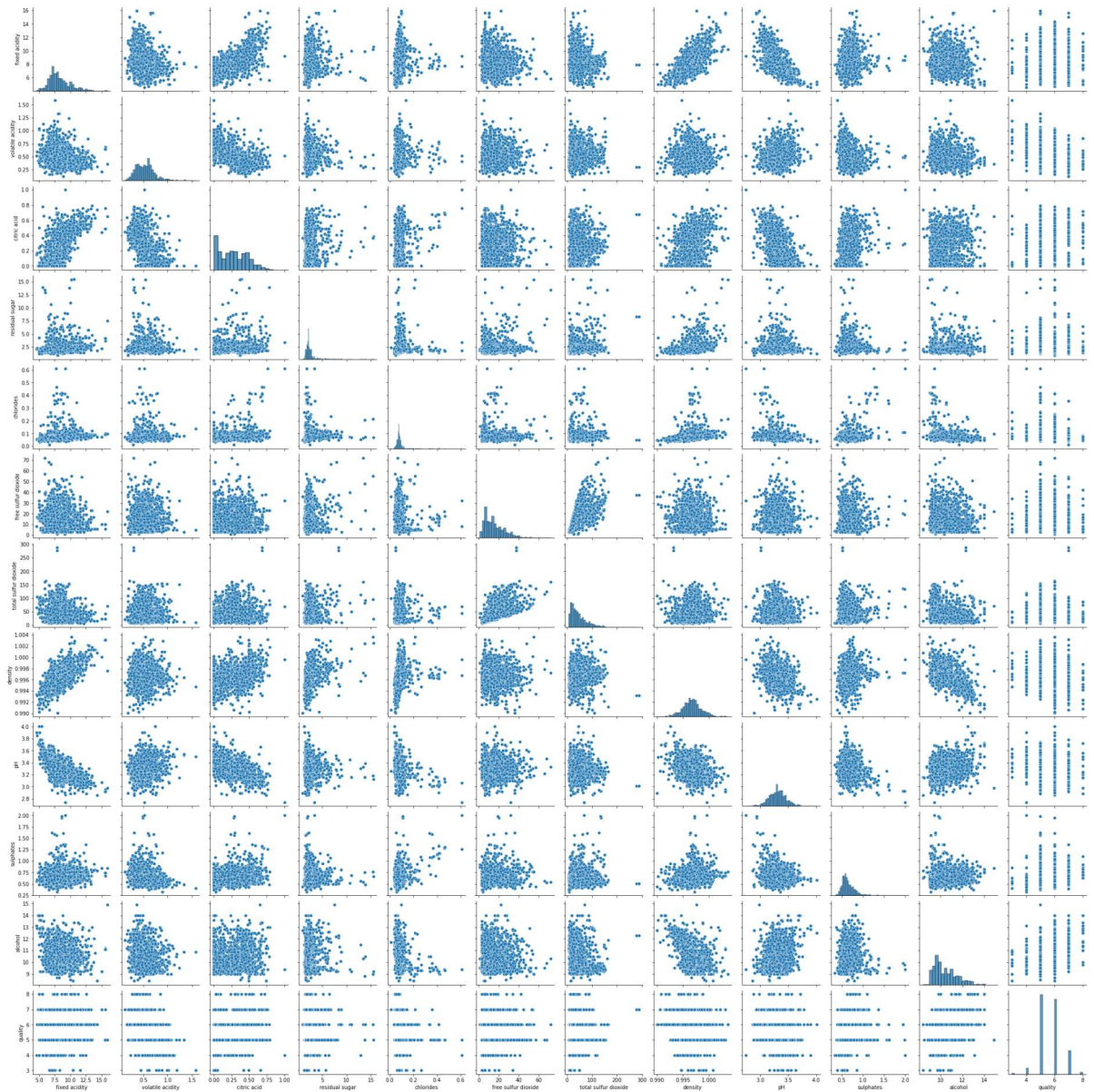
In [5]: `df.describe()`

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur 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	46.46779
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 [6]: `df.columns`

Out[6]: Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')

In [7]: `sns.pairplot(df)`

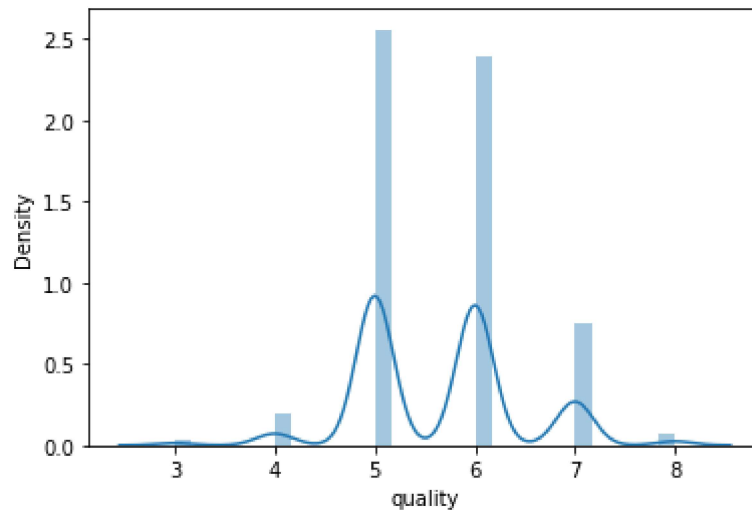Out[7]: `<seaborn.axisgrid.PairGrid at 0x26e27b9b9d0>`

In [8]:
```python
sns.distplot(df['quality'])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
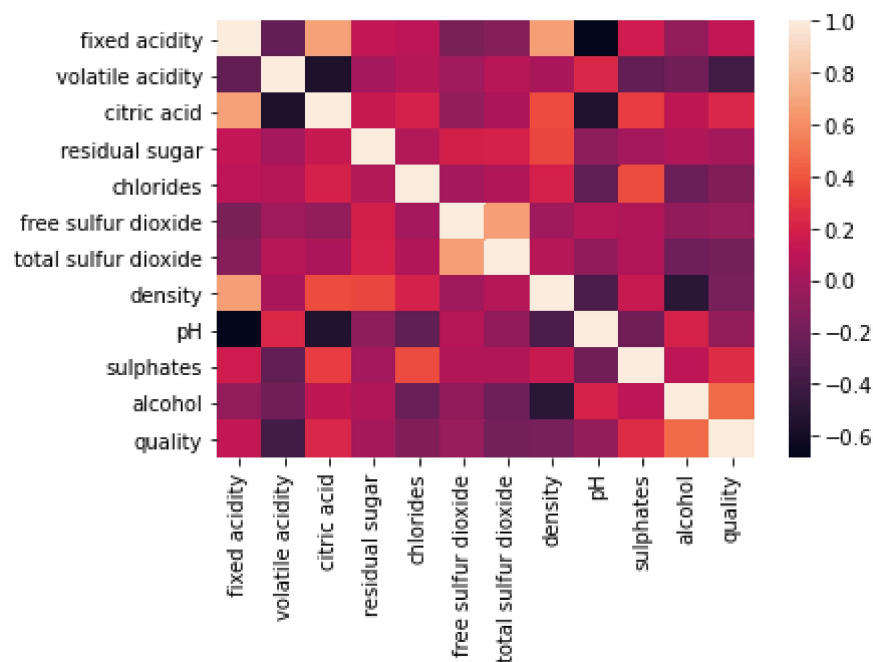stograms).
  warnings.warn(msg, FutureWarning)

Out[8]: <AxesSubplot:xlabel='quality', ylabel='Density'>

In [9]:
```python
df1=df[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
        'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
        'pH', 'sulphates', 'alcohol', 'quality']]
```

In [10]:
```python
sns.heatmap(df1.corr())
```

Out[10]: <AxesSubplot:>

In [11]:
```python
x=df1[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol']]
y=df1['quality']
```

In [12]:
```python
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 [13]:
```python
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[13]: LinearRegression()

In [14]:
```python
print(lr.intercept_)
```

5.845811866151241

In [15]:
```python
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
fixed acidity	-0.007488
volatile acidity	-1.026444
citric acid	-0.054716
residual sugar	0.000125
chlorides	-1.842561
free sulfur dioxide	0.005322
total sulfur dioxide	-0.003336
density	-0.883610
pH	-0.648912
sulphates	0.905897
alcohol	0.295879

In [16]:
```python
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[16]: <matplotlib.collections.PathCollection at 0x26e3046c400>
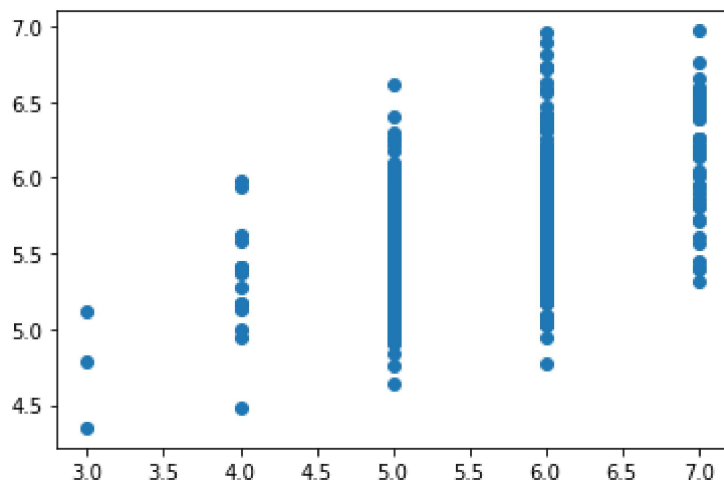
In [17]:
```python
print(lr.score(x_test,y_test))
```

0.3504856066186194

In [18]:
```python
from sklearn.linear_model import Ridge,Lasso
```

In [19]:
```python
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[19]: Ridge(alpha=10)

In [20]:
```python
rr.score(x_test,y_test)
```

Out[20]: 0.33712542505350773

In [21]:
```python
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[21]: Lasso(alpha=10)

In [22]:
```python
la.score(x_test,y_test)
```

Out[22]: -0.004986667842787673

In [23]:
```python
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[23]: ElasticNet()

In [24]:
```python
print(en.coef_)
```

```
[ 0.         -0.          0.          0.         -0.          0.00152978
 -0.00433797 -0.         -0.          0.          0.         ]
```

In [25]: `print(en.intercept_)`

5.828127534502547

In [26]: `print(en.predict(x_train))`

[5.67933677 5.64616282 5.7765747 ... 5.76967992 5.68162068 5.62753255]

In [27]: `print(en.score(x_train,y_train))`

0.034199718376713095

In [28]: `from sklearn import metrics`

In [29]: `print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))`

Mean Absolytre Error: 0.48957558832521975

In [30]: `print("Mean Square Error:",metrics.mean_squared_error(y_test,prediction))`

Mean Square Error: 0.3861876139917913

In [31]: `print("Root Mean Square Error:",np.sqrt(metrics.mean_absolute_error(y_test,pre`

Root Mean Square Error: 0.6996967831319648

In [ ]: