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("/Users/bob/Downloads/11_winequality-red.csv")
df.fillna(0,inplace=True)
df
```

Out[2]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
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
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66

1599 rows × 12 columns

In [3]:

```
df.head()
```

Out[3]:

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

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]:

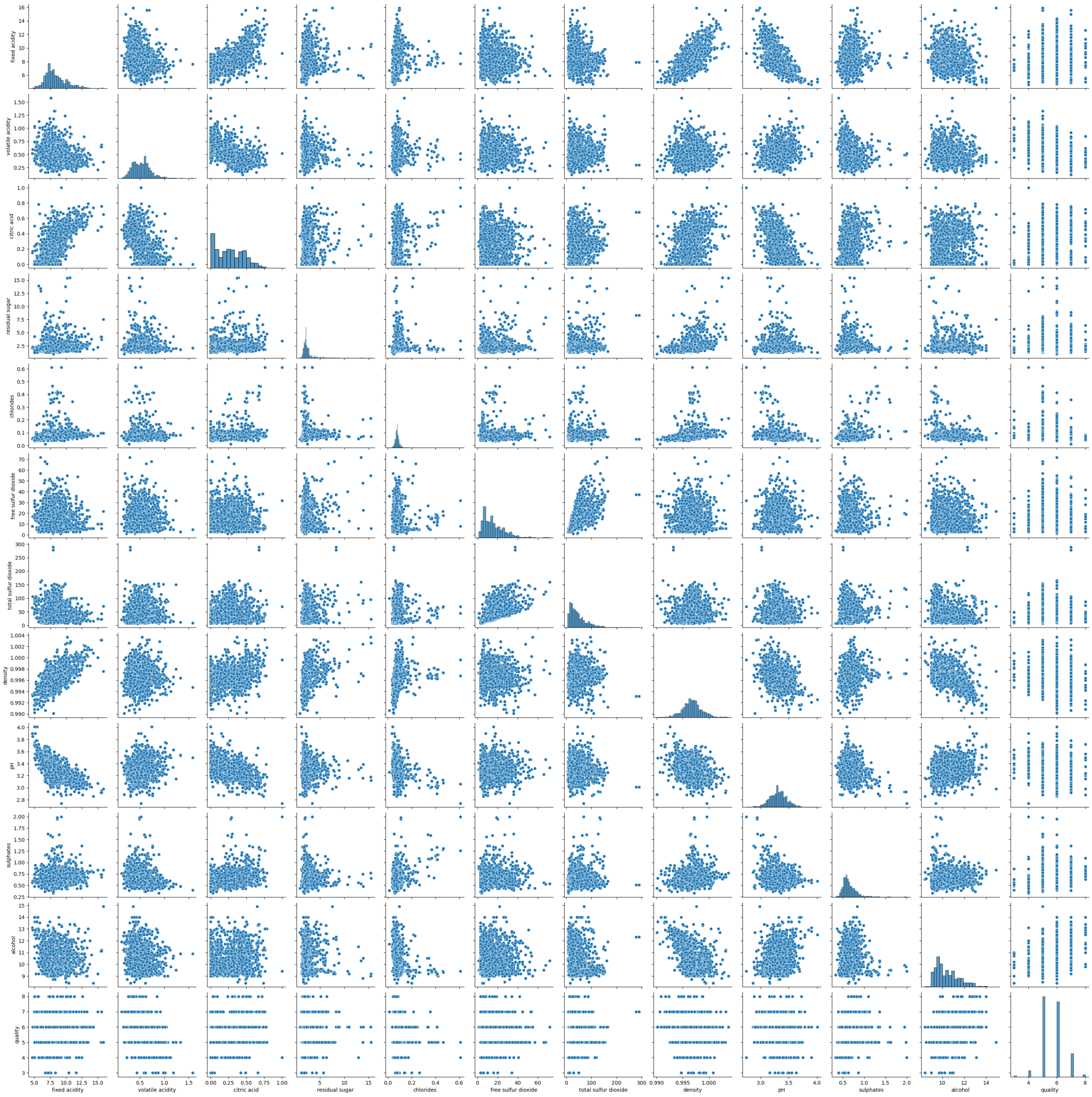
```
df.columns
```

Out[5]:

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residu
al 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 0x7f8ef1bed900>
```

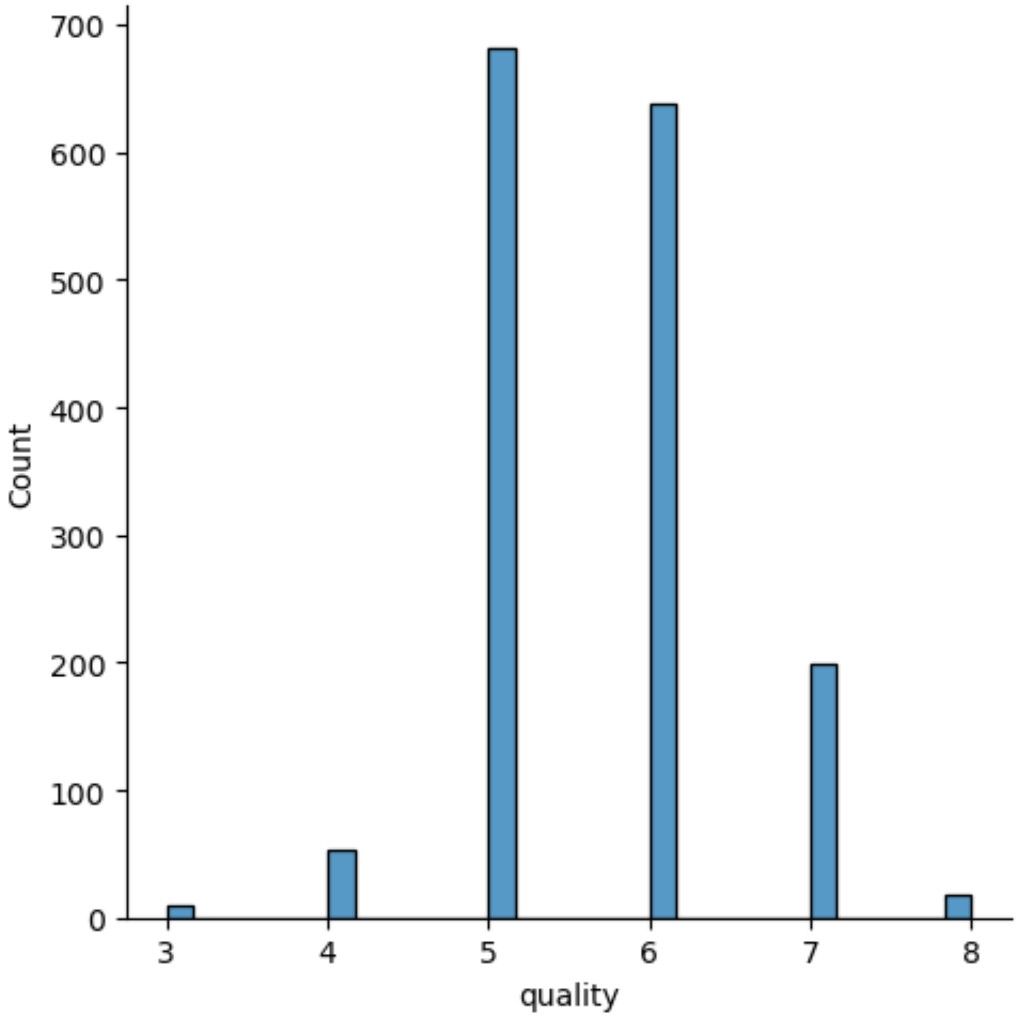


In [7]:

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

Out[7]:

```
<seaborn.axisgrid.FacetGrid at 0x7f8ec91de980>
```



In [8]:

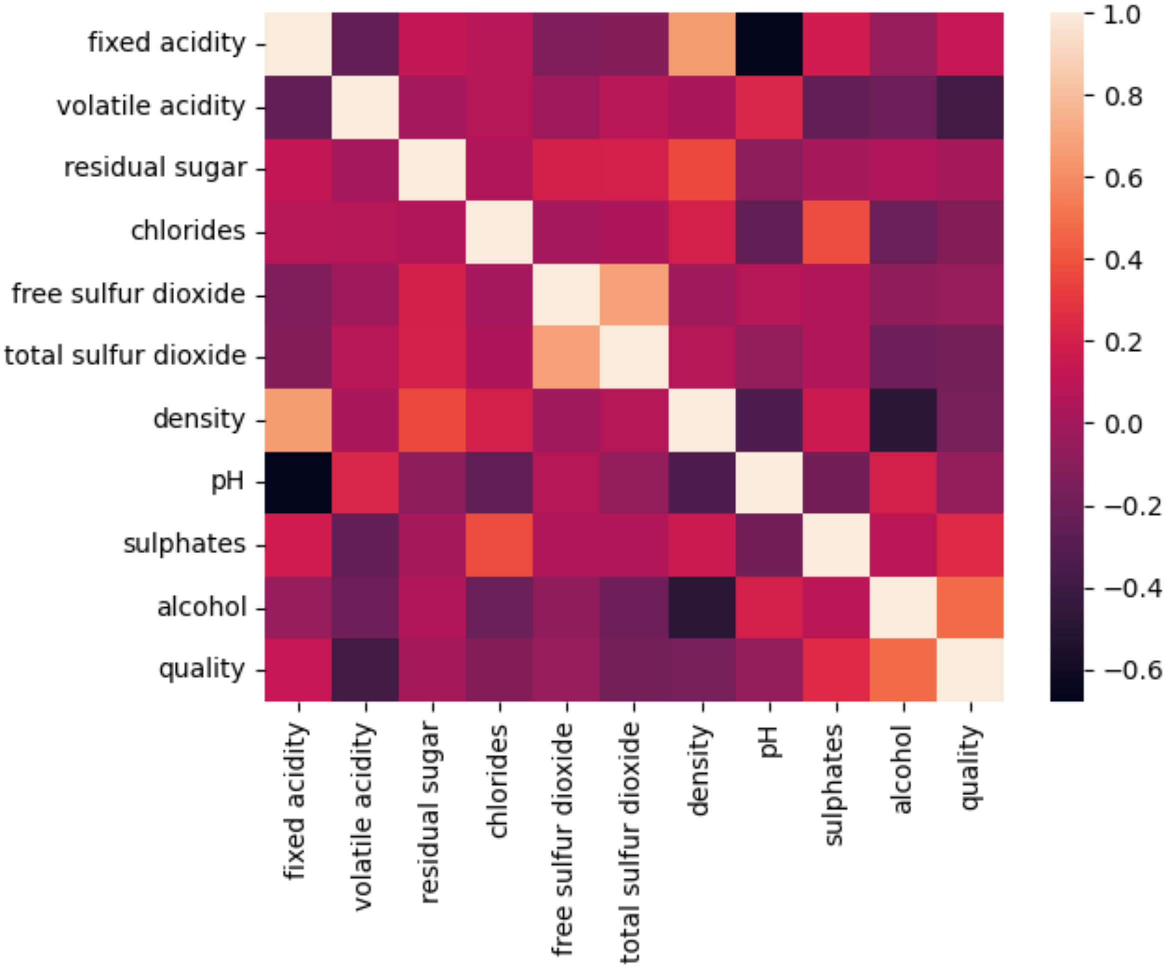
```
df1=df.drop(['citric acid'],axis=1)
df1
df1=df1.drop(df1.index[1537:])
df1.isna().sum()
```

Out[8]:

```
fixed acidity           0
volatile acidity        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
```

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

Out[9]:
```
<Axes: >
```



In [10]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [11]:
```
df1.isna().sum()
```

Out[11]:
```
fixed acidity           0
volatile acidity        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
```

In [12]:
```
y=df1['quality']
x=df1.drop(['volatile acidity'],axis=1)
```

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

```
      fixed acidity  residual sugar  chlorides  free sulfur dioxid
e  \
792             7.1             2.5      0.081                17.
0
974             8.8             5.9      0.073                 7.
0
901             7.4             2.4      0.080                16.
0
1436           10.0             1.6      0.169                27.
0
435            12.3             2.3      0.091                 6.
0
...             ...             ...        ...
...
515             8.5             6.1      0.122                34.
0
350            10.7             2.7      0.107                17.
0
460             9.2             2.3      0.083                14.
0
68              9.3             2.0      0.074                27.
0
1497            6.9             2.3      0.054                 7.
0

      total sulfur dioxide  density    pH  sulphates  alcohol  qua
lity
792                   87.0  0.99745  3.48       0.60      9.7
6
974                   13.0  0.99658  3.30       0.62     12.1
7
901                   33.0  0.99736  3.58       0.69     10.8
7
1436                  90.0  0.99914  3.15       0.65      8.5
5
435                   18.0  1.00040  3.16       0.49      9.5
5
...                    ...      ...   ...        ...      ...
...
515                  151.0  1.00100  3.31       1.14      9.3
5
350                   34.0  1.00040  3.28       0.98      9.9
6
460                   23.0  0.99760  3.35       0.61     11.3
6
68                    65.0  0.99690  3.28       0.79     10.7
5
1497                  16.0  0.99508  3.45       0.63     11.5
6

[1075 rows x 10 columns]
```

In [13]:

```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

Out[13]:

```
4.440892098500626e-14
```

In [14]:

```
model.coef_
```

Out[14]:

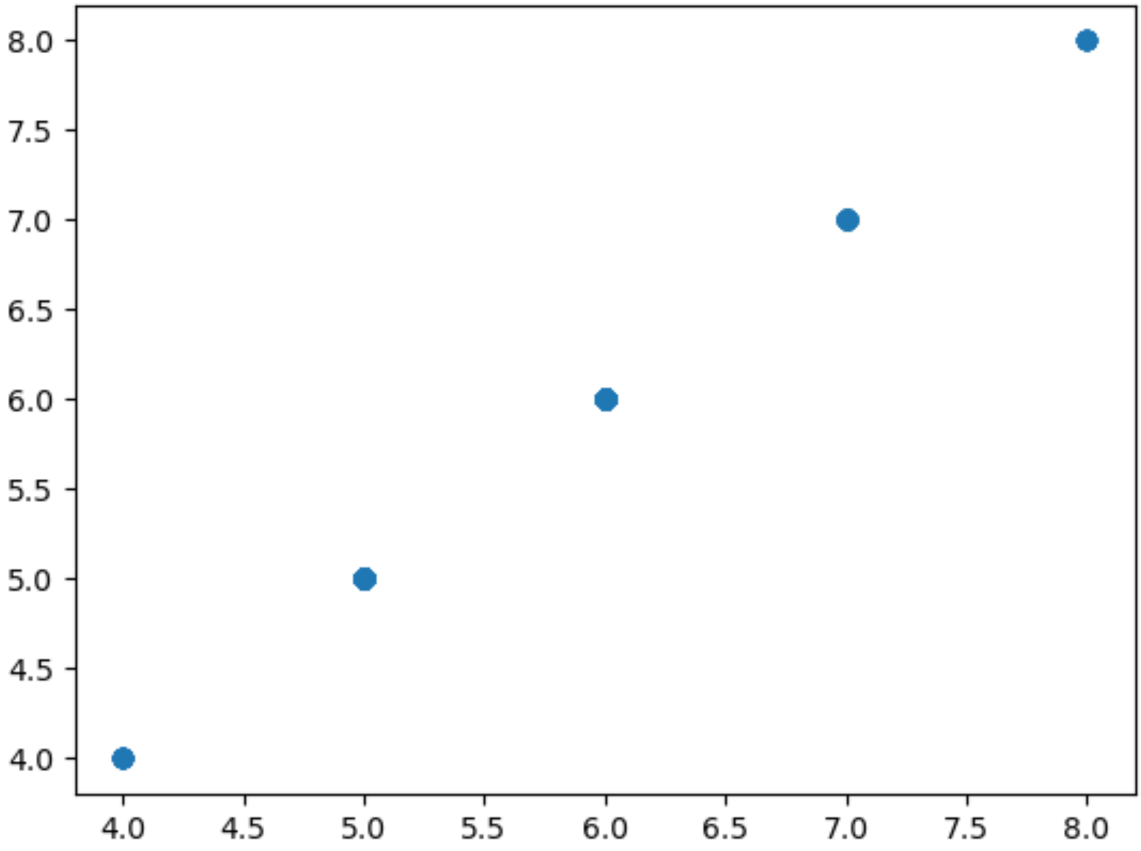
```
array([ 7.33711071e-17, -3.50414142e-16, -2.27913527e-15,  6.36643
516e-16,
       -1.63064007e-16, -4.69906630e-14,  1.41813644e-16,  6.84944
722e-17,
        7.96888597e-18,  1.00000000e+00])
```

In [15]:

```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[15]:

```
<matplotlib.collections.PathCollection at 0x7f8eca903880>
```



In [16]:

```
model.score(x_test,y_test)
```

Out[16]:

```
1.0
```

In [ ]: