In [296]:

```
# IMPORT LIBRARIES
import numpy  as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [297]:

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

Out[297]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	a
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 [298]:

```
a=a.head(10)
a
```

Out[298]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alco
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	
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	1
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	
9	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	1

In [299]:

```
# to find
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         10 non-null     float64
 1   volatile acidity      10 non-null     float64
 2   citric acid           10 non-null     float64
 3   residual sugar        10 non-null     float64
 4   chlorides             10 non-null     float64
 5   free sulfur dioxide   10 non-null     float64
 6   total sulfur dioxide  10 non-null     float64
 7   density               10 non-null     float64
 8   pH                    10 non-null     float64
 9   sulphates             10 non-null     float64
 10  alcohol               10 non-null     float64
 11  quality               10 non-null     int64  
dtypes: float64(11), int64(1)
memory usage: 1.1 KB
```

In [300]:

```
# to display summary of statastic
a.describe()
```

Out[300]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density
count	10.000000	10.000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000
mean	7.950000	0.631	0.104000	2.330000	0.077000	14.800000	48.900000	0.997080
std	1.162612	0.161	0.194548	1.376025	0.010198	4.467164	25.066356	0.001038
min	7.300000	0.280	0.000000	1.200000	0.065000	9.000000	18.000000	0.994600
25%	7.400000	0.585	0.000000	1.825000	0.071500	11.500000	34.000000	0.996800
50%	7.650000	0.655	0.010000	1.900000	0.075000	15.000000	47.000000	0.997400
75%	7.800000	0.700	0.055000	2.225000	0.076000	16.500000	59.750000	0.997800
max	11.200000	0.880	0.560000	6.100000	0.098000	25.000000	102.000000	0.998000

In [301]:

```
# to display colum heading
a.columns
```

Out[301]:

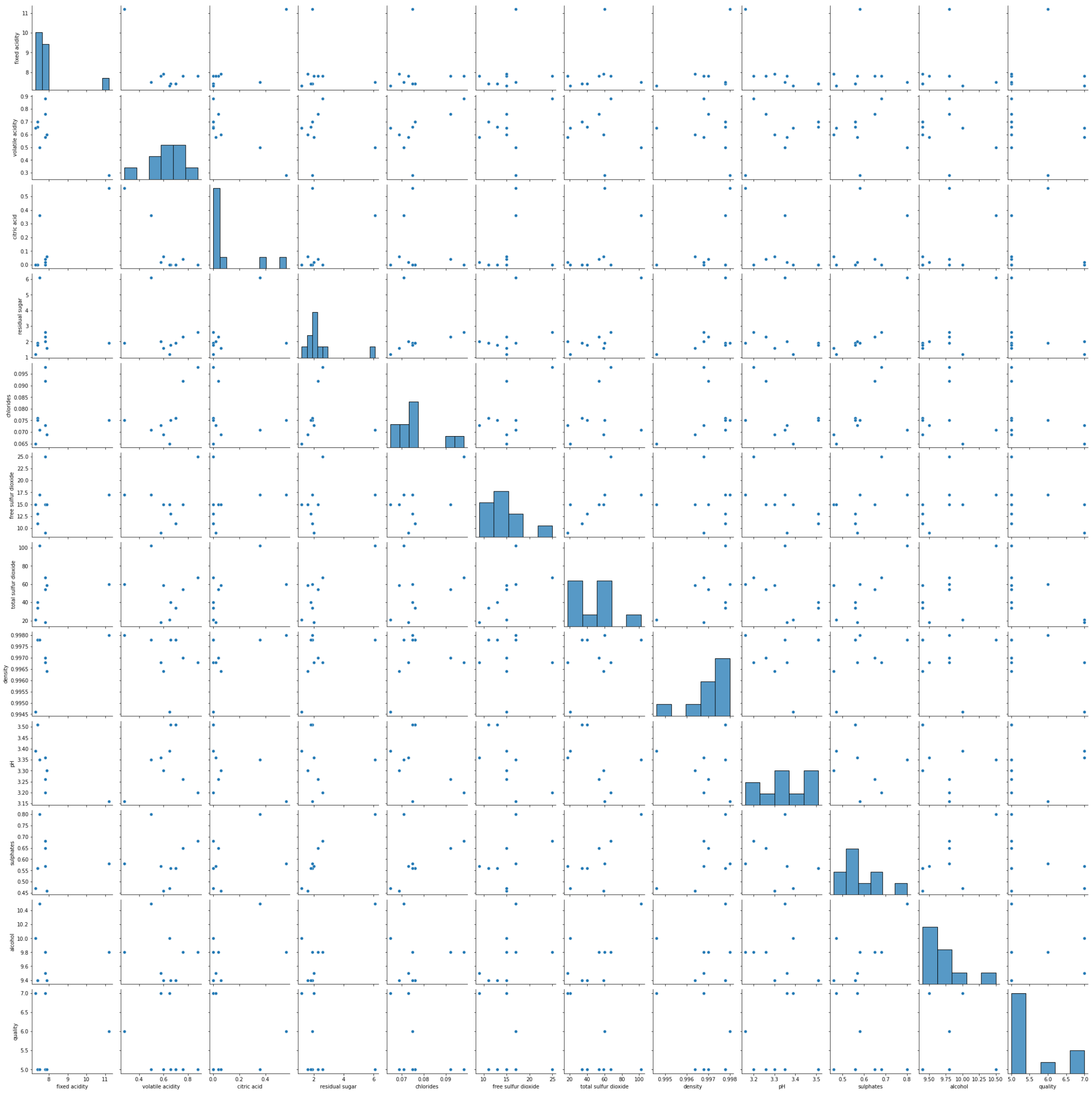
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual suga
r',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'densit
y',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

In [302]:

```
sns.pairplot(a)
```

Out[302]:

```
<seaborn.axisgrid.PairGrid at 0x198d5b2ff40>
```

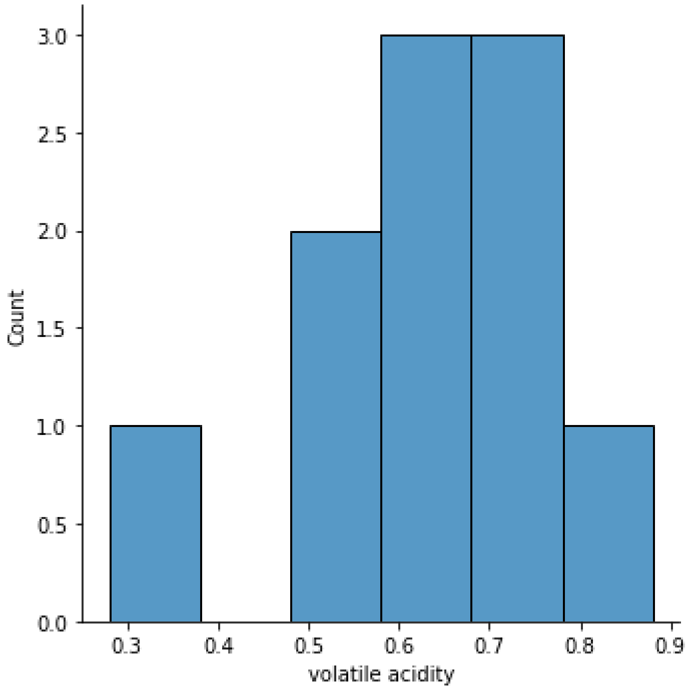


In [304]:

```
sns.displot(a["volatile acidity"])
```

Out[304]:

```
<seaborn.axisgrid.FacetGrid at 0x198db9abd60>
```



In [305]:

```
b=a[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide']]
b
```

Out[305]:

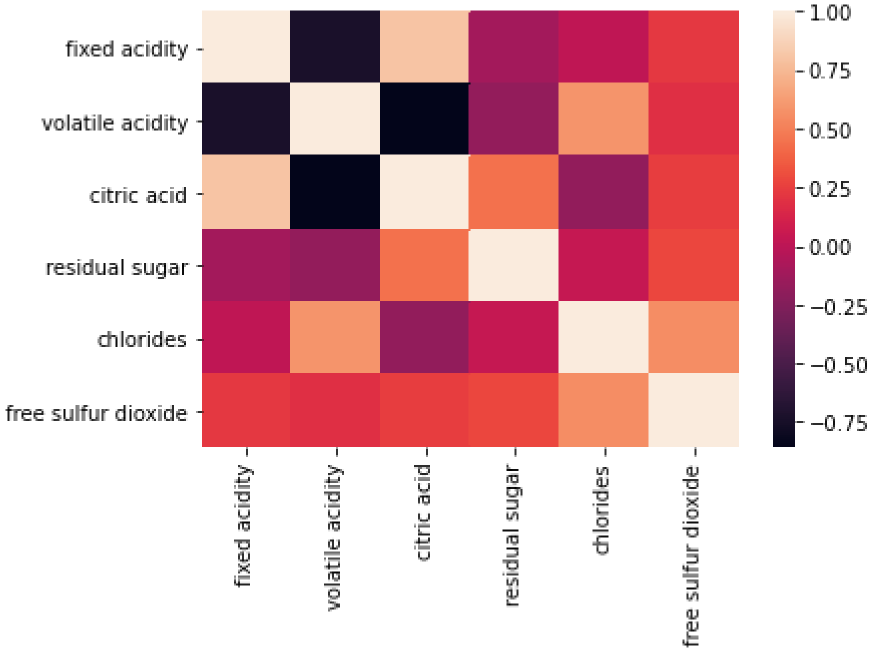
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide
0	7.4	0.70	0.00	1.9	0.076	11.0
1	7.8	0.88	0.00	2.6	0.098	25.0
2	7.8	0.76	0.04	2.3	0.092	15.0
3	11.2	0.28	0.56	1.9	0.075	17.0
4	7.4	0.70	0.00	1.9	0.076	11.0
5	7.4	0.66	0.00	1.8	0.075	13.0
6	7.9	0.60	0.06	1.6	0.069	15.0
7	7.3	0.65	0.00	1.2	0.065	15.0
8	7.8	0.58	0.02	2.0	0.073	9.0
9	7.5	0.50	0.36	6.1	0.071	17.0

In [306]:

```
sns.heatmap(b.corr())
```

Out[306]:

```
<AxesSubplot:>
```



In [308]:

```
x=a[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide']]
y=a['free sulfur dioxide']
```

In [309]:

```
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 [310]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[310]:

```
LinearRegression()
```

In [311]:

```
lr.intercept_
```

Out[311]:

```
-9.791006556270077e-05
```

In [312]:

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

Out[312]:

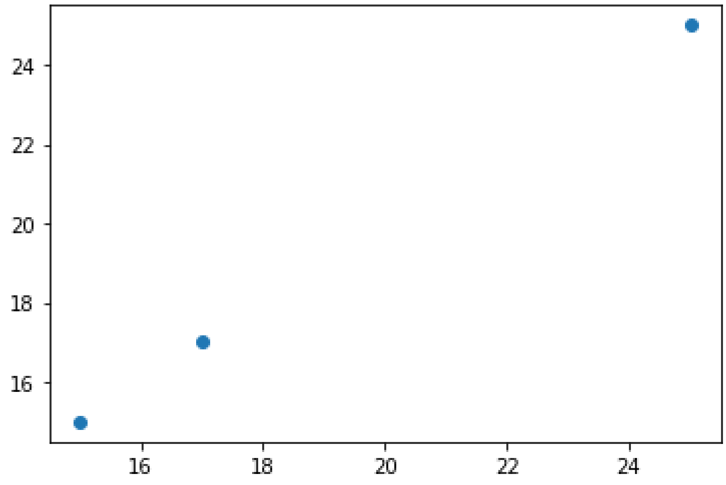
	Co-efficient
fixed acidity	0.000013
volatile acidity	-0.000006
citric acid	-0.000078
residual sugar	-0.000038
chlorides	0.001192
free sulfur dioxide	0.999999

In [313]:

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

Out[313]:

```
<matplotlib.collections.PathCollection at 0x198dcddb910>
```



In [314]:

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

Out[314]:

```
0.9999999992775543
```

In [315]:

```
lr.score(x_train,y_train)
```

Out[315]:

```
1.0
```

In [316]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [317]:

```
rr=Ridge(alpha=10)
rr.fit(x_test,y_test)
```

Out[317]:

```
Ridge(alpha=10)
```

In [318]:

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

Out[318]:

```
0.9771563092244006
```

In [319]:

```
la=Lasso(alpha=10)
la.fit(x_test,y_test)
```

Out[319]:

```
Lasso(alpha=10)
```

In [320]:

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

Out[320]:

```
0.7130102040816326
```

In [321]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[321]:

```
ElasticNet()
```

In [322]:

```
en.coef_
```

Out[322]:

```
array([ 0.        , -0.        ,  0.        , -0.        ,  0.        ,
        0.86407767])
```

In [323]:

```
en.intercept_
```

Out[323]:

```
1.766990291262136
```

In [324]:

```
prediction=en.predict(x_test)
prediction
```

Out[324]:

```
array([14.72815534, 23.36893204, 16.45631068])
```

In [325]:

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

Out[325]:

```
0.9458949948157223
```

EVALUATION METRICS

In [326]:

```
from sklearn import metrics
```

In [327]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.8155339805825262
```

In [328]:

```
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error 1.0099600967731843
```

In [329]:

```
print("Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error 1.0049677093186549
```

In [ ]:

```

```