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

In [2]:
```python
df=pd.read_csv(r"C:\Users\user\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	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]:
```python
df.head()
```

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

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]:
```
import seaborn as sns
```

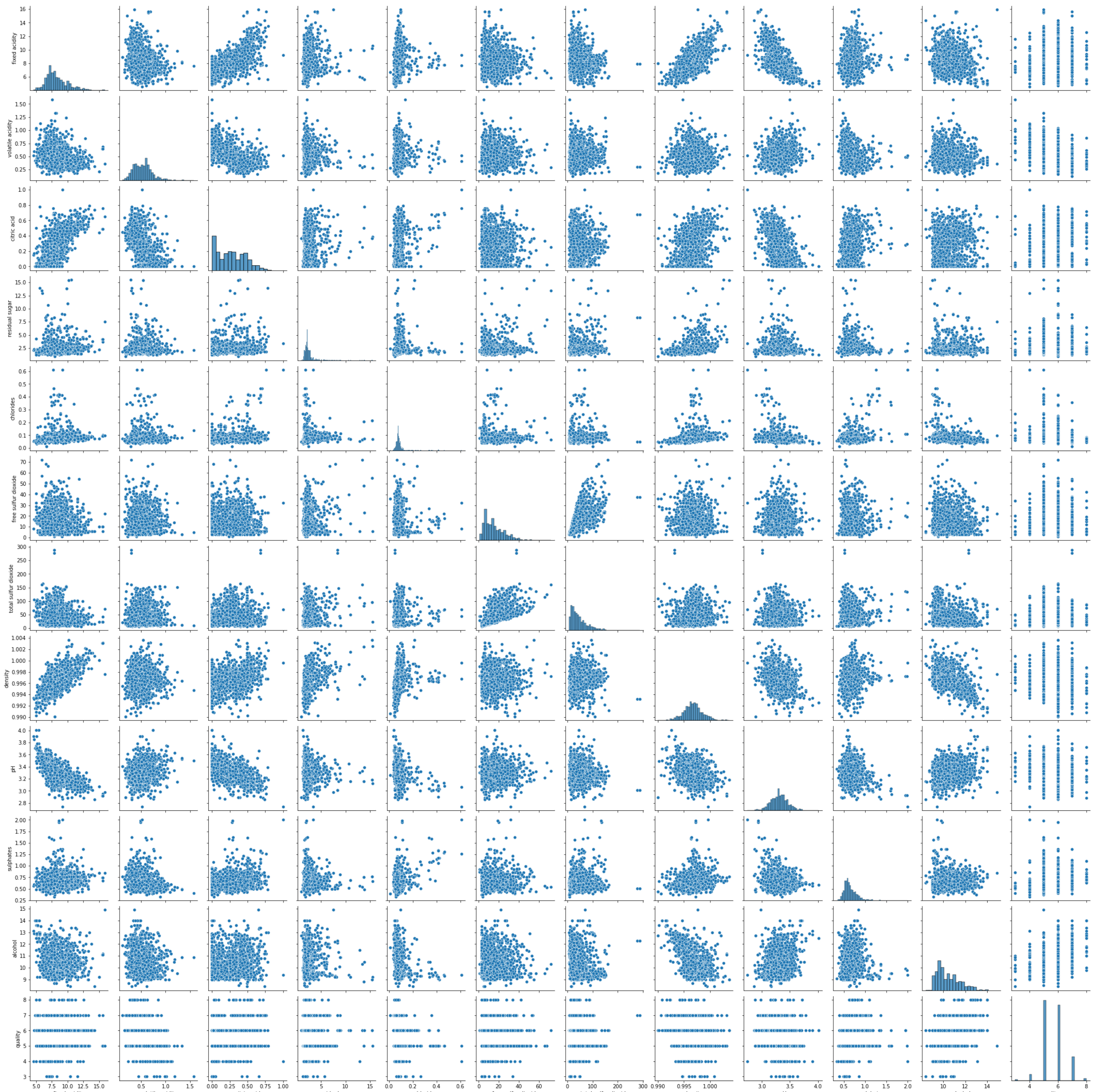
In [6]:
```
df.describe()
```

Out[6]:

	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	46.46779
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 [ ]:

In [7]:
```
sns.pairplot(df)
```
Out[7]: <seaborn.axisgrid.PairGrid at 0x1fdf66b4e80>

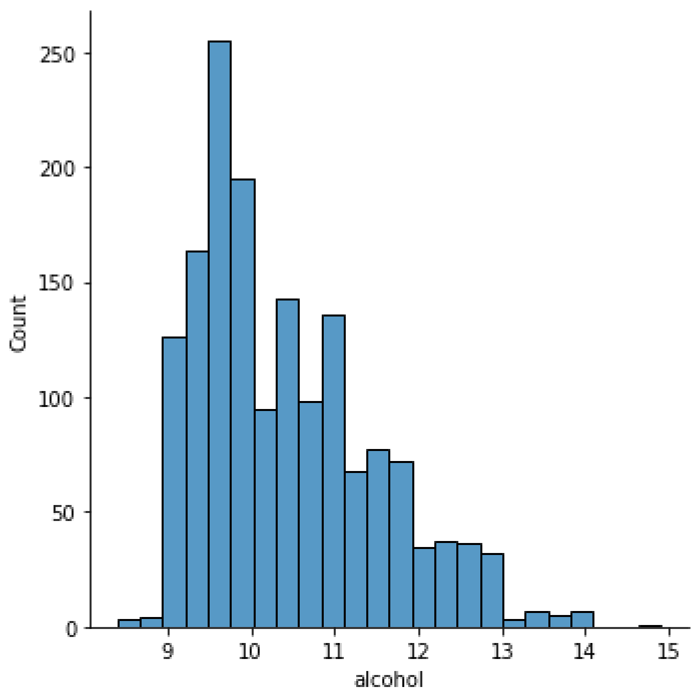


In [8]:
```
df1=df.drop(['citric acid'],axis=1)
df1
df1=df1.drop(df1.index[1537:])
df1.isna().sum()
```

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
```

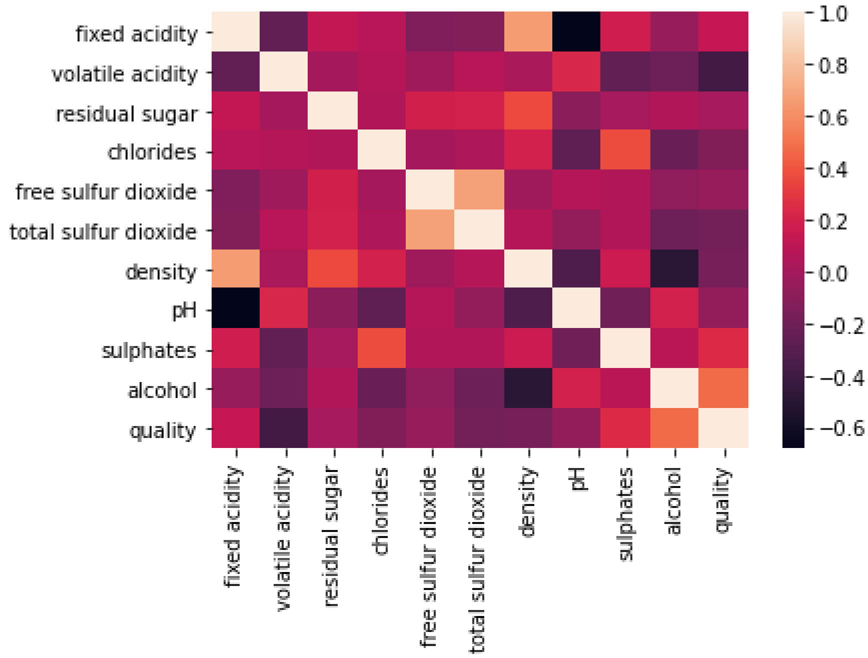
In [9]:
```
sns.displot(df['alcohol'])
```

Out[9]:
```
<seaborn.axisgrid.FacetGrid at 0x1fdfda65190>
```



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

Out[10]: <AxesSubplot:>



In [11]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [12]:
```
df1.isna().sum()
```

Out[12]:
```
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 [13]:
```
y=df1['fixed acidity']
x=df1.drop(['chlorides','residual sugar'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
print(x_train)
```

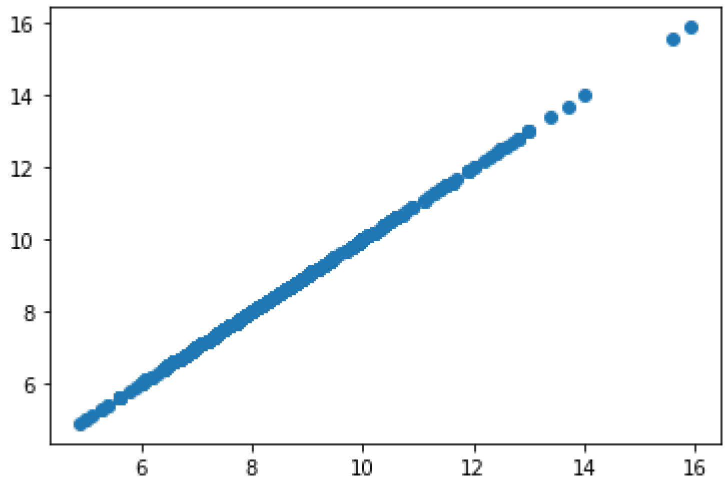
```
      fixed acidity  volatile acidity  free sulfur dioxide  \
1341            7.5              0.51                 13.0   
801             8.6              0.55                  8.0   
1459            7.9              0.20                  7.0   
756             6.3              0.98                 15.0   
1126            5.8              0.29                  3.0   
...             ...               ...                  ...   
583            12.0              0.28                 10.0   
1518            7.4              0.47                  7.0   
1490            7.1              0.22                  8.0   
388             7.8              0.46                 23.0   
118             8.8              0.55                 14.0   

      total sulfur dioxide  density    pH  sulphates  alcohol  quality  
1341                  31.0  0.99538  3.36       0.54     10.5        6  
801                   17.0  0.99735  3.23       0.44     10.0        5  
1459                  15.0  0.99458  3.32       0.80     11.9        7  
756                   33.0  0.99488  3.60       0.46     11.2        6  
1126                  11.0  0.99150  3.39       0.54     13.5        6  
...                    ...      ...   ...        ...      ...      ...  
583                   21.0  0.99760  2.98       0.66      9.9        7  
1518                  20.0  0.99647  3.32       0.63     10.5        5  
1490                  18.0  0.99344  3.39       0.56     12.4        6  
388                   53.0  0.99810  3.43       0.74      9.2        6  
118                   56.0  0.99620  3.21       0.60     10.9        6  

[1075 rows x 9 columns]
```

In [14]:
```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

Out[14]:
```
7.993605777301127e-14
```

In [15]:
```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[15]: <matplotlib.collections.PathCollection at 0x1fdfffd7610>



In [16]:
```
model.score(x_test,y_test)
```

Out[16]: 1.0

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]: 0.9999872504686617

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]: -0.00024395652394071377

In [22]:

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

print(en.coef_)


print(en.intercept_)


print(en.predict(x_test))


print(en.score(x_test,y_test))


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))


print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,pred
```

```
  7.67156836  7.04546752  9.849262   10.04525574  7.43595514  8.32032773
  7.95160255  8.05184922 13.49796054 12.15211542  8.15569941  8.29091502
  6.79844956  7.91693551  8.89210275  7.46386799  8.26615284  9.38628961
  8.50176607  7.5987817   9.98762828  6.78689388  8.10302439 10.63173711
  9.64336338  8.78975242  9.25617738  8.00382474  8.40947156  7.09334105
 10.72238081  8.33203436  7.64170281  9.4424172   5.98460902  8.92016654
  6.25308753  7.04546752  7.930142    7.49508247  5.98610889  8.23328756
  7.06197563  7.80813289  7.81158546  7.55451168 11.53937202  9.86907174
  8.63517992  8.15554846 11.6496745   8.08321465 10.49832326  9.7224514
  7.59248035  7.37157349  8.71426793  8.59075895  8.80326081  8.78840351
 10.71247594  8.33203436  7.29908872  6.9663795  11.48339537  9.05673107
  8.21993012  8.51497256  8.8854995   7.15096851  7.44090757 10.34675049
  8.78840351  8.66639439  7.17557974  9.93165163 11.05493908  8.58085408
 10.95619227  6.46408953  6.46739115  7.97621378  7.4077404  10.18692365
  7.22990558  5.92037831  7.86921292  7.7949264   9.20185155  7.70608446]
0.9155323766730585
Mean Absolute Error: 7.782134724991574e-15
Mean Squared Error: 9.409300658681618e-29
Root Mean Squared Error: 9.700154977463823e-15
```