In [2]:
```
# import packages
import pandas as pd
import  numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
# read the data
df=pd.read_csv(r'F:\FSDS\Data Files\winequality_red.csv')
```

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

Out[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	a
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 [5]:
```
# divide the data into two parts x and y
X=df.drop('quality',axis=1)
y=df['quality']
X.values
```

Out[5]:
```
array([[ 7.4  ,  0.7  ,  0.   , ...,  3.51 ,  0.56 ,  9.4  ],
       [ 7.8  ,  0.88 ,  0.   , ...,  3.2  ,  0.68 ,  9.8  ],
       [ 7.8  ,  0.76 ,  0.04 , ...,  3.26 ,  0.65 ,  9.8  ],
       ...,
       [ 6.3  ,  0.51 ,  0.13 , ...,  3.42 ,  0.75 , 11.   ],
       [ 5.9  ,  0.645,  0.12 , ...,  3.57 ,  0.71 , 10.2  ],
       [ 6.   ,  0.31 ,  0.47 , ...,  3.39 ,  0.66 , 11.   ]])
```

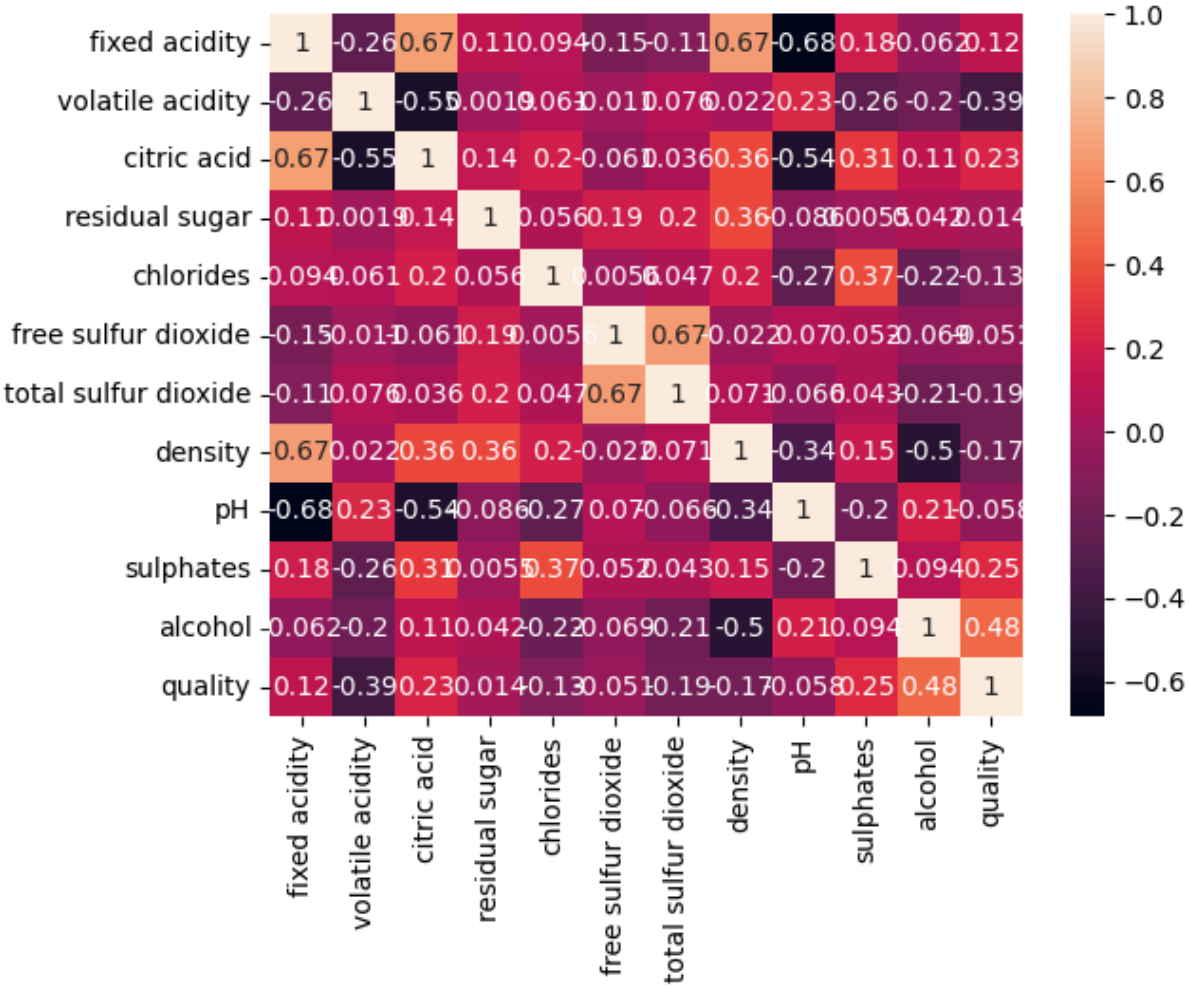
In [6]:
```
corr=df.corr()
corr
```

Out[6]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	den
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.66
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.02
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.36
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.35
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.20
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666	-0.02
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000	0.07
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269	1.00
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070377	-0.066495	-0.34
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051658	0.042947	0.14
alcohol	-0.061668	-0.202288	0.109903	0.042075	-0.221141	-0.069408	-0.205654	-0.49
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050656	-0.185100	-0.17

In [7]:
```
sns.heatmap(corr, annot=True)
```

Out[7]:
```
<Axes: >
```



In [8]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [9]:
```
# Apply VIF

vif=[variance_inflation_factor(X.values,i)  for i in range(len(X.columns))]

vif_data=pd.DataFrame(vif, index=X.columns, columns=['VIF'])

vif_data.sort_values(by= 'VIF', ascending=False )
```

Out[9]:

	VIF
density	1479.287209
pH	1070.967685
alcohol	124.394866
fixed acidity	74.452265
sulphates	21.590621
volatile acidity	17.060026
citric acid	9.183495
chlorides	6.554877
total sulfur dioxide	6.519699
free sulfur dioxide	6.442682
residual sugar	4.662992

In [10]:
```
## select VIF less than 15 columns
cols1=vif_data[vif_data['VIF']<15].index
cols1=cols1.to_list()
vif_data[vif_data['VIF']>15].index
cols2=['density','sulphates',  'alcohol']

final_cols=cols1+cols2
```

In [11]:
```
final_cols
```

Out[11]:
```
['citric acid',
 'residual sugar',
 'chlorides',
 'free sulfur dioxide',
 'total sulfur dioxide',
 'density',
 'sulphates',
 'alcohol']
```

In [12]:
```
import warnings
warnings.filterwarnings('ignore')

final_df=X[final_cols]
final_df['quality']=y
final_df
```

Out[12]:

	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	sulphates	alcohol	quality
0	0.00	1.9	0.076	11.0	34.0	0.99780	0.56	9.4	5
1	0.00	2.6	0.098	25.0	67.0	0.99680	0.68	9.8	5
2	0.04	2.3	0.092	15.0	54.0	0.99700	0.65	9.8	5
3	0.56	1.9	0.075	17.0	60.0	0.99800	0.58	9.8	6
4	0.00	1.9	0.076	11.0	34.0	0.99780	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...
1594	0.08	2.0	0.090	32.0	44.0	0.99490	0.58	10.5	5
1595	0.10	2.2	0.062	39.0	51.0	0.99512	0.76	11.2	6
1596	0.13	2.3	0.076	29.0	40.0	0.99574	0.75	11.0	6
1597	0.12	2.0	0.075	32.0	44.0	0.99547	0.71	10.2	5
1598	0.47	3.6	0.067	18.0	42.0	0.99549	0.66	11.0	6

1599 rows × 9 columns

Scaling

In [13]:
```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
X.scaled=ss.fit_transform(X)
X_scaled_df=pd.DataFrame(X.scaled, columns=X.columns)
X_scaled_df
```

Out[13]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density
0	-0.528360	0.961877	-1.391472	-0.453218	-0.243707	-0.466193	-0.379133	0.558274
1	-0.298547	1.967442	-1.391472	0.043416	0.223875	0.872638	0.624363	0.028261
2	-0.298547	1.297065	-1.186070	-0.169427	0.096353	-0.083669	0.229047	0.134264
3	1.654856	-1.384443	1.484154	-0.453218	-0.264960	0.107592	0.411500	0.664277
4	-0.528360	0.961877	-1.391472	-0.453218	-0.243707	-0.466193	-0.379133	0.558274
...	...	...	...	...	...	...	...	...
1594	-1.217796	0.403229	-0.980669	-0.382271	0.053845	1.542054	-0.075043	-0.978765
1595	-1.390155	0.123905	-0.877968	-0.240375	-0.541259	2.211469	0.137820	-0.862162
1596	-1.160343	-0.099554	-0.723916	-0.169427	-0.243707	1.255161	-0.196679	-0.533554
1597	-1.390155	0.654620	-0.775267	-0.382271	-0.264960	1.542054	-0.075043	-0.676657
1598	-1.332702	-1.216849	1.021999	0.752894	-0.434990	0.203223	-0.135861	-0.666057

1599 rows × 11 columns

Train-Test-Split

In [14]:
```
from sklearn.model_selection import train_test_split
```

In [15]:
```
X_train, X_test, y_train, y_test= train_test_split(X_scaled_df, y, test_size=0.2, r
```

In [16]:
```
X_scaled_df.shape, X_train.shape, X_test.shape
```

Out[16]:
```
((1599, 11), (1279, 11), (320, 11))
```

In [17]:
```
y_train.shape, y_test.shape
```

Out[17]:
```
((1279,), (320,))
```

In [18]:
```
X_train.index
```

Out[18]:
```
Index([ 441,  227, 1386, 1245,   60,  463, 1017, 1226, 1213,   33,
       ...
       1257,  279,  689,  664, 1396, 1228, 1077, 1318,  723,  815],
      dtype='int64', length=1279)
```

In [19]:
```
y_train.index
```

Out[19]:
```
Index([ 441,  227, 1386, 1245,   60,  463, 1017, 1226, 1213,   33,
       ...
        1257,  279,  689,  664, 1396, 1228, 1077, 1318,  723,  815],
      dtype='int64', length=1279)
```

In [20]:
```
X_test.index
```

Out[20]:
```
Index([ 688,  961,  726,  537, 1544, 1251, 1096, 1547, 1446,  142,
       ...
        1583,  414, 1250,  232,  853,  351,  415,  564, 1124,  147],
      dtype='int64', length=320)
```

In [21]:
```
y_test.index
```

Out[21]:
```
Index([ 688,  961,  726,  537, 1544, 1251, 1096, 1547, 1446,  142,
       ...
        1583,  414, 1250,  232,  853,  351,  415,  564, 1124,  147],
      dtype='int64', length=320)
```

Develop the model

In [22]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train, y_train)
```

Out[22]:
```
▾ LinearRegression
LinearRegression()
```

prediction

In [23]:
```
y_pred=lr.predict(X_test)
y_pred
```

Out[23]:
```
array([5.18402693, 5.35222632, 5.59491103, 5.52091961, 6.34175415,
       5.39173054, 5.4386121 , 5.95987316, 5.47004148, 6.91005691,
       4.98509136, 5.23277066, 5.69063919, 5.52790309, 5.51816101,
       5.25774342, 4.97138685, 5.12531479, 6.16831487, 5.17480772,
       5.11570282, 5.84314362, 5.18881658, 5.47252159, 5.7234016 ,
       6.30684795, 5.29979062, 5.56538456, 6.00395044, 5.51753047,
       4.75589372, 6.22649885, 5.24955921, 5.61323762, 5.54866083,
       5.4907831 , 6.35361446, 5.23982145, 5.561088  , 5.81104267,
       5.21412309, 6.08767487, 5.71655198, 5.34443848, 5.65810399,
       5.02362366, 4.94708499, 5.80532814, 6.37676041, 5.54270371,
       4.92835023, 5.73605681, 6.61798021, 6.15855747, 5.3704761 ,
       5.11932029, 6.22074911, 5.2262822 , 6.44385318, 5.26198348,
       5.28301604, 5.12118822, 5.12443471, 5.59218111, 5.02408579,
       6.1394565 , 5.71787898, 6.31942795, 5.27013911, 6.0846879 ,
       5.72948584, 5.53327202, 5.25038298, 6.17460327, 5.56538456,
       5.32083864, 6.0150401 , 6.43530813, 4.88814887, 5.04913954,
       5.68729866, 5.57896455, 5.7290186 , 5.97612524, 5.36898168,
       5.02880372, 6.12712446, 5.31278196, 5.98479919, 6.13194433,
       6.51189266, 5.41283432, 6.05002097, 5.0137043 , 5.46517764,
       5.90374559, 5.30178062, 6.6341725 , 6.55082539, 6.0150401 ,
       5.39782793, 5.49065123, 5.00327726, 5.55240601, 5.56966691,
       5.31776897, 5.8785315 , 5.16223316, 5.42433364, 5.50048598,
       6.36084316, 5.56366246, 6.20650152, 5.33506802, 5.55268727,
       5.65470064, 5.38775256, 5.88472934, 6.79479485, 5.22782945,
       5.95305554, 6.00736039, 5.60945827, 6.32139675, 6.04958854,
       5.08971314, 5.24655894, 6.46316326, 6.26310018, 6.20577219,
       5.30871293, 5.06889943, 5.87371277, 5.04742047, 5.29421699,
       6.33394821, 5.86535105, 5.44319332, 5.41377041, 5.55953192,
       5.92200919, 4.90131241, 5.42674997, 5.17617378, 5.75249678,
       5.73817797, 5.20373637, 6.21836803, 6.14855266, 5.38058809,
       5.47800612, 5.76410437, 6.55219996, 6.67759536, 6.01314068,
       5.82095648, 5.7008064 , 5.6372558 , 6.53315447, 5.58878886,
       6.39210152, 5.79143598, 6.00258743, 5.36003016, 5.78468889,
       6.11196626, 5.35037059, 4.26057299, 5.76791222, 5.93493452,
       4.91915794, 5.72793247, 4.98863426, 5.73442024, 6.54892053,
       5.99519265, 5.32064824, 5.5417675 , 5.75760708, 4.99679769,
       6.00322766, 5.22622448, 5.81451724, 6.55850657, 5.26574422,
       5.14661707, 6.47572385, 6.66312736, 5.82315529, 6.20626549,
       6.01180722, 5.5084863 , 4.76066998, 6.41669333, 5.33756863,
       4.80308078, 5.37155263, 6.9354863 , 5.31029439, 5.0480916 ,
       5.85891607, 5.89886295, 5.74584066, 5.01770049, 6.60247899,
       5.57766701, 5.8545049 , 5.11620488, 5.05950153, 4.8547598 ,
       5.67796929, 5.02413353, 5.61492682, 5.08946876, 5.90855691,
       6.66104943, 5.34975935, 5.36444569, 5.34261961, 5.61719826,
       5.58848586, 5.89060018, 4.91678293, 5.39826875, 5.78675148,
       4.98881432, 5.50889257, 6.13549554, 5.58819852, 5.99411749,
       6.31666165, 5.50318331, 5.71787898, 5.0510233 , 5.44834178,
       5.30067944, 6.40801641, 5.76406936, 5.79597235, 6.33739864,
       5.26793641, 5.09437997, 6.34545057, 5.26134981, 5.23889138,
       4.96522989, 4.94576152, 5.76101187, 6.44246558, 6.2376922 ,
       5.64193325, 5.72848848, 6.19479981, 5.02413353, 5.84825118,
       5.48096597, 5.32537219, 5.19429238, 5.80581404, 5.40012258,
       5.40404058, 5.04857237, 6.12756557, 6.3227982 , 5.90458709,
       6.06247903, 6.05103932, 5.90143817, 5.35740672, 5.3704761 ,
       5.08845724, 5.28767367, 5.42170404, 5.37003359, 5.6390435 ,
       5.7296821 , 6.07227189, 4.96325159, 5.56339633, 5.33250366,
```

```
       6.12791952, 5.76158439, 5.86698002, 5.39748052, 5.28465633,
       5.75837267, 5.8149593 , 5.04968   , 5.32844916, 5.75393581,
       6.40127499, 5.9373478 , 4.96300949, 5.84909362, 5.70593835,
       5.61992295, 5.18050416, 5.35265055, 4.98881432, 5.3791698 ,
       5.94624924, 6.53523473, 5.91201571, 5.12493362, 6.52934475,
       6.3308775 , 5.73186785, 4.97265088, 5.64601463, 5.15505816,
       5.39108249, 5.11050691, 5.73657613, 5.55826824, 6.03342611,
       5.22024243, 5.14645374, 6.27657951, 5.77733102, 5.12131274])
```

In [24]:
```
len(y_pred)
```

Out[24]:
```
320
```

Coefficient

In [25]:
```
lr.coef_
```

Out[25]:
```
array([ 0.0226771 , -0.19455291, -0.02078537,  0.0220569 , -0.07958515,
        0.05044688, -0.12002793, -0.03457533, -0.06818618,  0.14404931,
        0.29540577])
```

In [26]:
```
lr.intercept_
```

Out[26]:
```
5.637208139901266
```

In [27]:
```
lr.intercept_+np.sum(lr.coef_*X_test.values[0])
```

Out[27]:
```
5.184026927195503
```

In [28]:
```
X_test
```

Out[28]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density
688	-0.356000	0.738418	-1.186070	-0.666062	-1.030094	-1.135608	-1.139357	-0.289747
961	-0.700719	0.179770	-0.672566	-0.666062	-0.201199	-0.848716	-0.865676	-0.438151
726	-0.126188	1.073606	-0.929318	0.185312	-0.073677	0.203223	0.077002	1.406296
537	-0.126188	1.660186	-0.159061	-0.311323	-0.073677	-1.039977	-1.017721	0.240266
1544	0.046171	-0.881661	0.816598	-0.169427	-0.520005	-0.370562	-0.835267	-0.660757
...	...	...	...	...	...	...	...	...
351	0.448342	1.492592	-1.391472	0.043416	0.181368	-0.466193	-0.622404	1.406296
415	0.161077	1.101539	-0.159061	2.881328	0.627696	1.446423	2.661764	2.466324
564	2.689011	-0.323013	1.124700	1.249529	-0.052423	-0.944346	0.016184	2.837333
1124	-1.045437	0.291499	-1.391472	-0.240375	0.181368	-1.231239	-1.017721	-0.623656
147	-0.413454	-0.211283	-0.056360	-0.666062	3.156889	-0.561823	1.262952	0.028261

320 rows × 11 columns

In [29]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_pred, y_test)
```

Out[29]:
```
0.37988847343818904
```

In [30]:
```
from sklearn.metrics import mean_squared_error, r2_score
mse=mean_squared_error(y_test, y_pred)
R_Square=r2_score(y_test, y_pred)
rmse=np.sqrt(mse)
print('Mse:', mse)
print('rmse:', rmse)
print('r2_score:', R_Square)
```

```
Mse: 0.37988847343818904
rmse: 0.6163509336718725
r2_score: 0.36758336427516125
```

save the model

In [31]:
```
lr
```

Out[31]:
```
LinearRegression
LinearRegression()
```

In [36]:
```
import pickle
path=open('F:\FSDS\Data Files\linear_regression_wine.pkl','wb')
pickle.dump(lr,path)
```

load the model

In [37]:
```
import warnings
warnings.filterwarnings('ignore')

file_path='F:\FSDS\Data Files\linear_regression_wine.pkl'
with open(file_path, 'rb') as file:
    data=pickle.load(file)

data
```

Out[37]:
```
▾ LinearRegression
LinearRegression()
```

In [41]:
```
data.predict([[1,2,3,4,5,6,7,8,9,10,11]])
```

Out[41]:
```
array([8.16088926])
```

In [ ]:

In [ ]:

In [ ]: