Name :C.Rushitha

Reg.No : 21BCE5460

Gmail : chennareddygari.rushitha2021@vitstudent.ac.in

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
df=pd.read_csv("winequality-red.csv")df.head()
```

```
   fixed acidity volatile acidity citric acid residual sugar
chlorides \
0            7.4              0.70         0.00             1.9
0.076
1            7.8              0.88         0.00             2.6
0.098
2            7.8              0.76         0.04             2.3
0.092
3           11.2              0.28         0.56             1.9
0.075
4            7.4              0.70         0.00             1.9
0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates
\
0                 11.0                  34.0   0.9978  3.51       0.56

1                 25.0                  67.0   0.9968  3.20       0.68

2                 15.0                  54.0   0.9970  3.26       0.65

3                 17.0                  60.0   0.9980  3.16       0.58

4                 11.0                  34.0   0.9978  3.51       0.56


   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
```

```
df.shape
```

```
(1599, 12)
```

```
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
```

```
df.isnull().any()
```

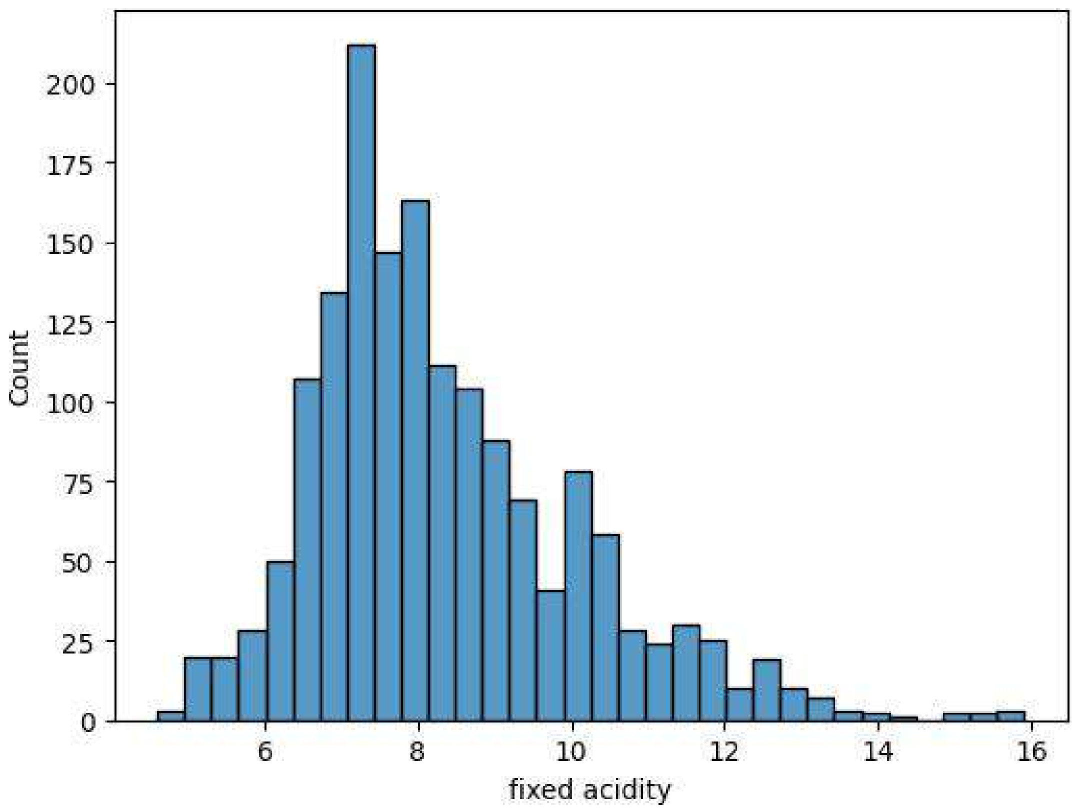
```
fixed acidity           False
volatile acidity        False
citric acid             False
residual sugar          False
chlorides               False
free sulfur dioxide     False
total sulfur dioxide    False
density                 False
pH                      False
sulphates               False
alcohol                 False
quality                 False
dtype: bool
```

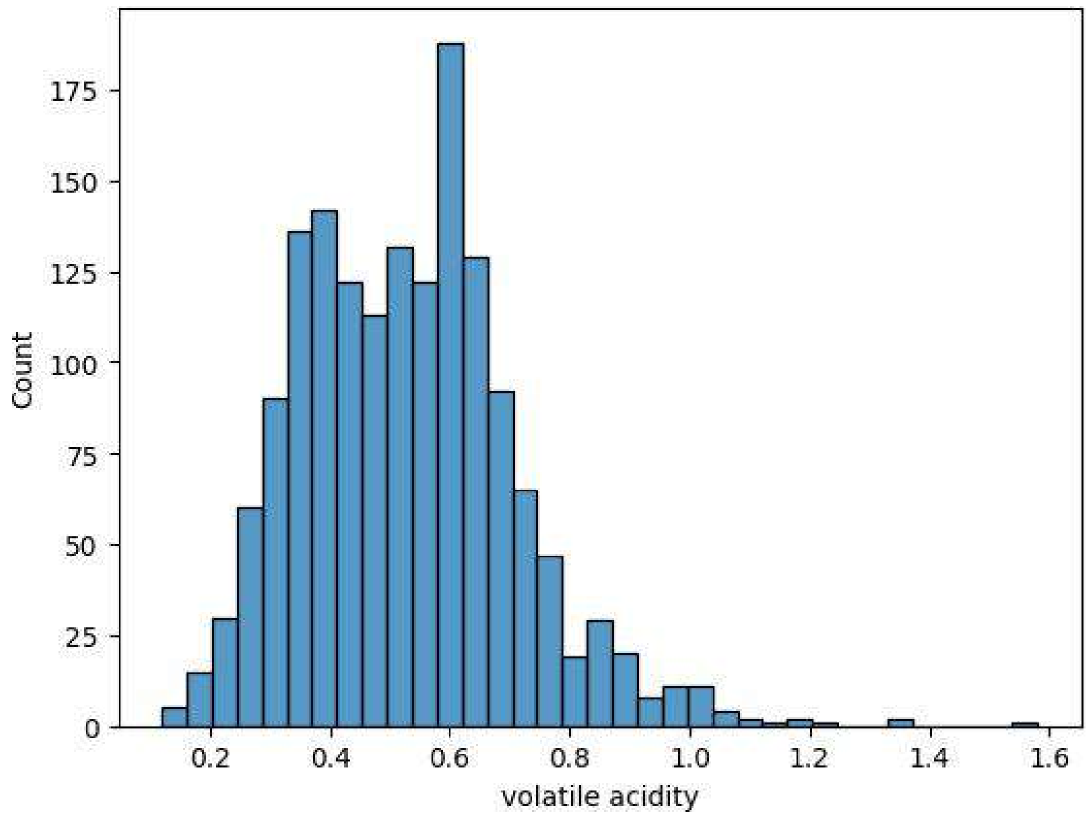
#Data Preprocessing and Visualisation

- ## Univariate Analysis

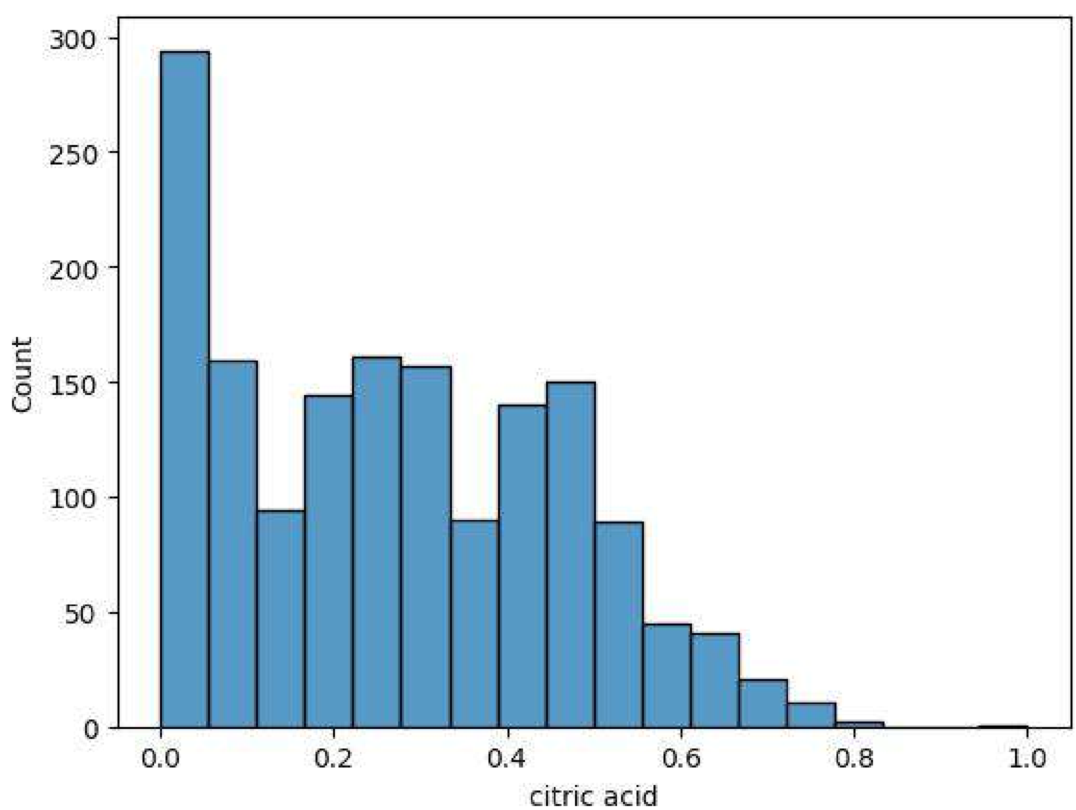
1) Histogram

```
for i in df.columns[:-1]:
  sns.histplot(df[i])
  plt.show()
```

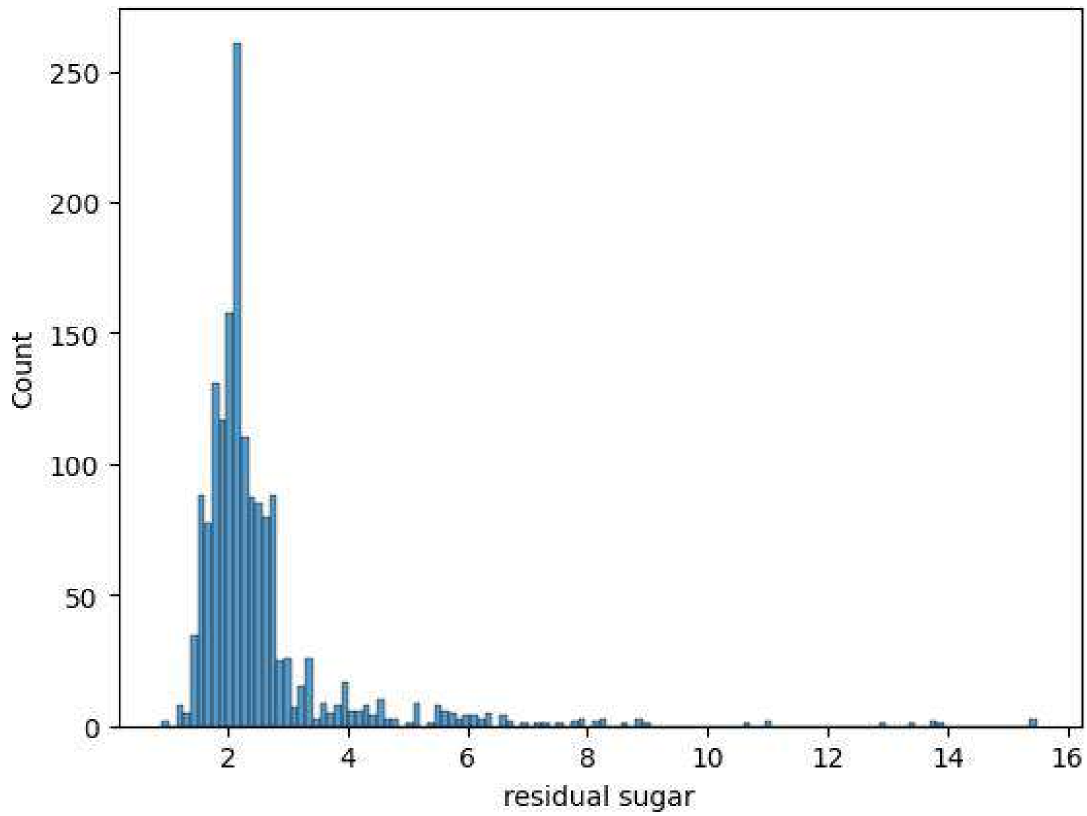




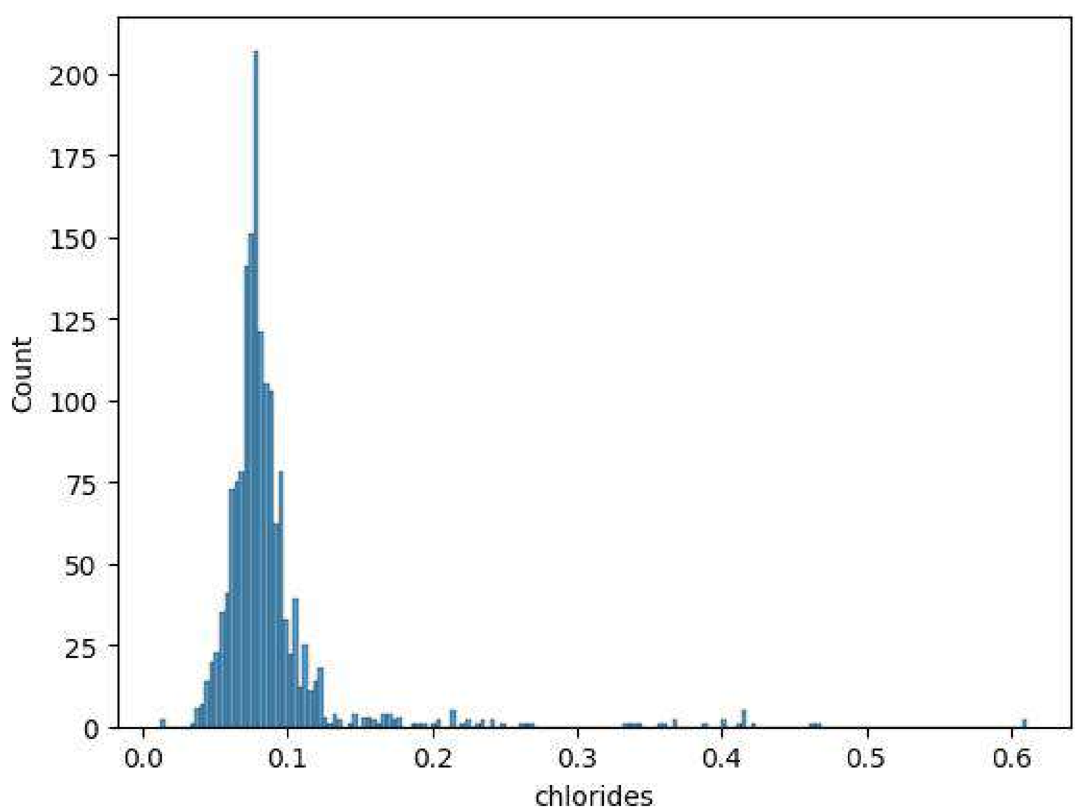
Count
0
25
50
75
100
125
150
175
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
volatile acidity



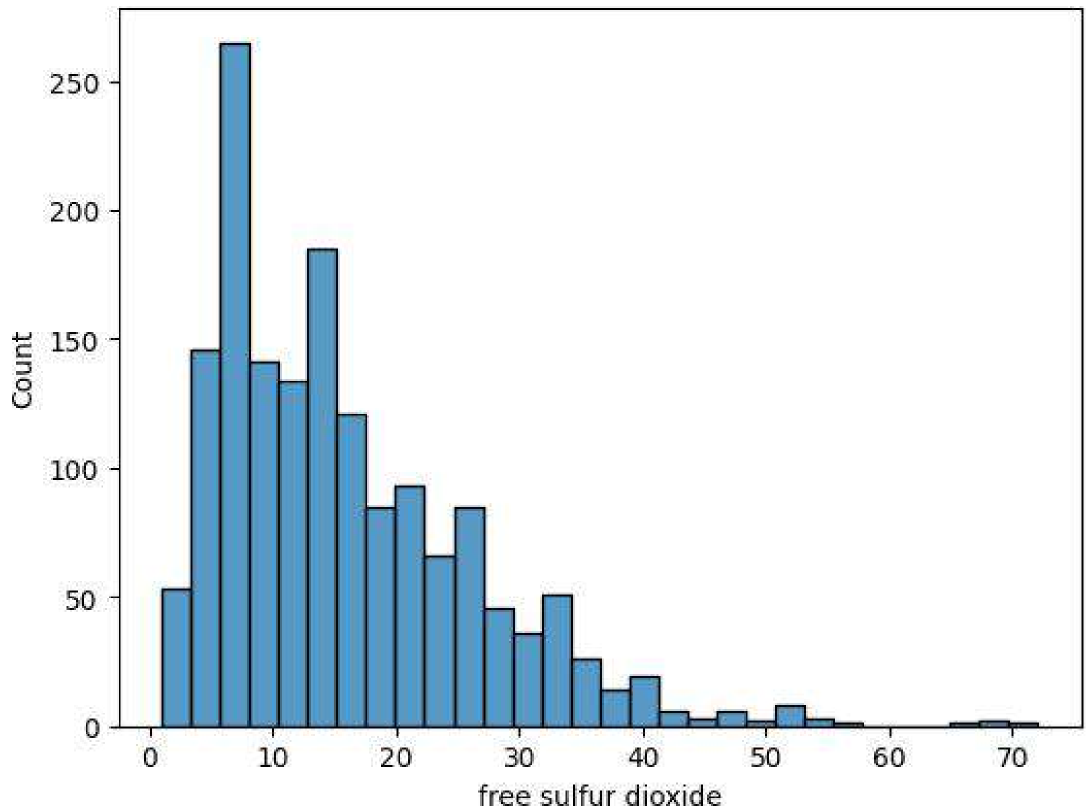
Count
0
50
100
150
200
250
300
0.0
0.2
0.4
0.6
0.8
1.0
citric acid

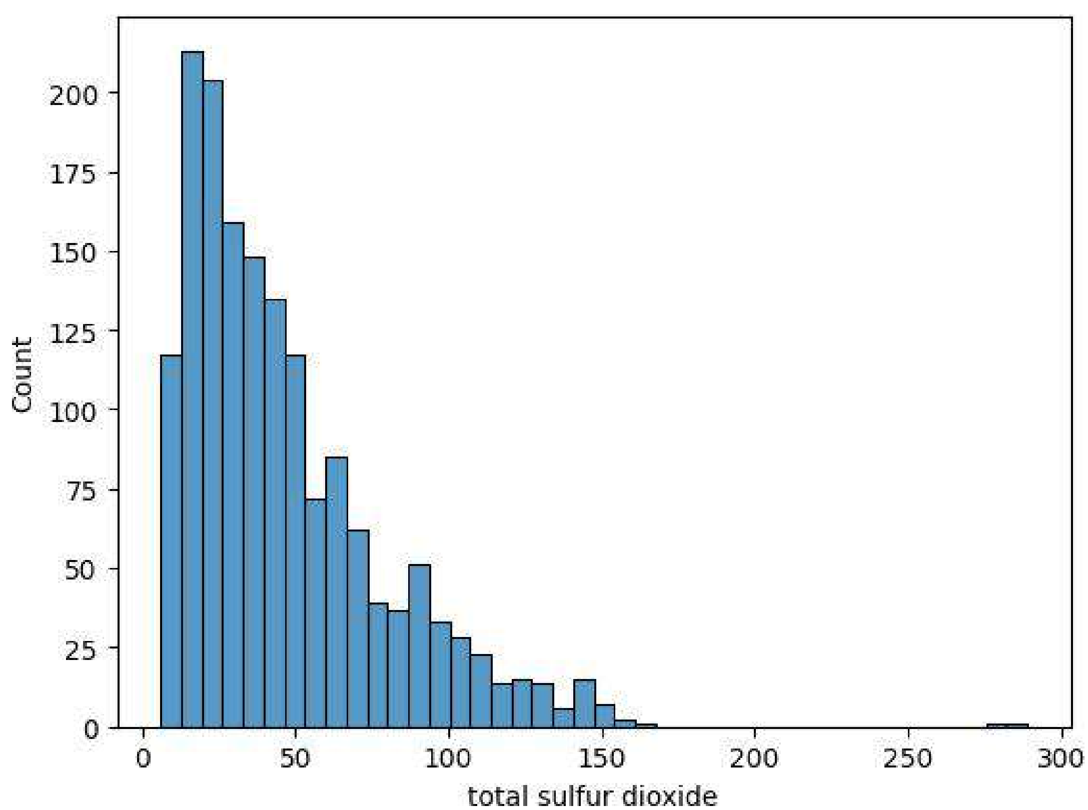


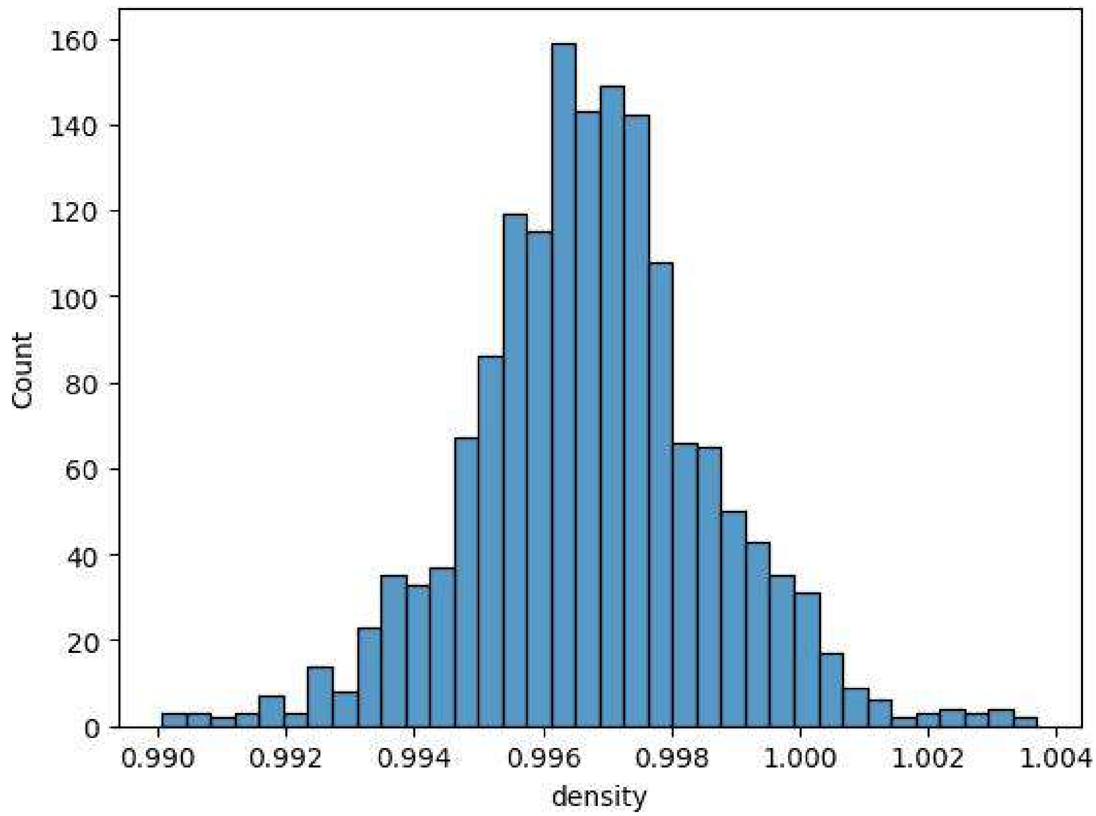
Count
residual sugar
0
50
100
150
200
250
2
4
6
8
10
12
14
16



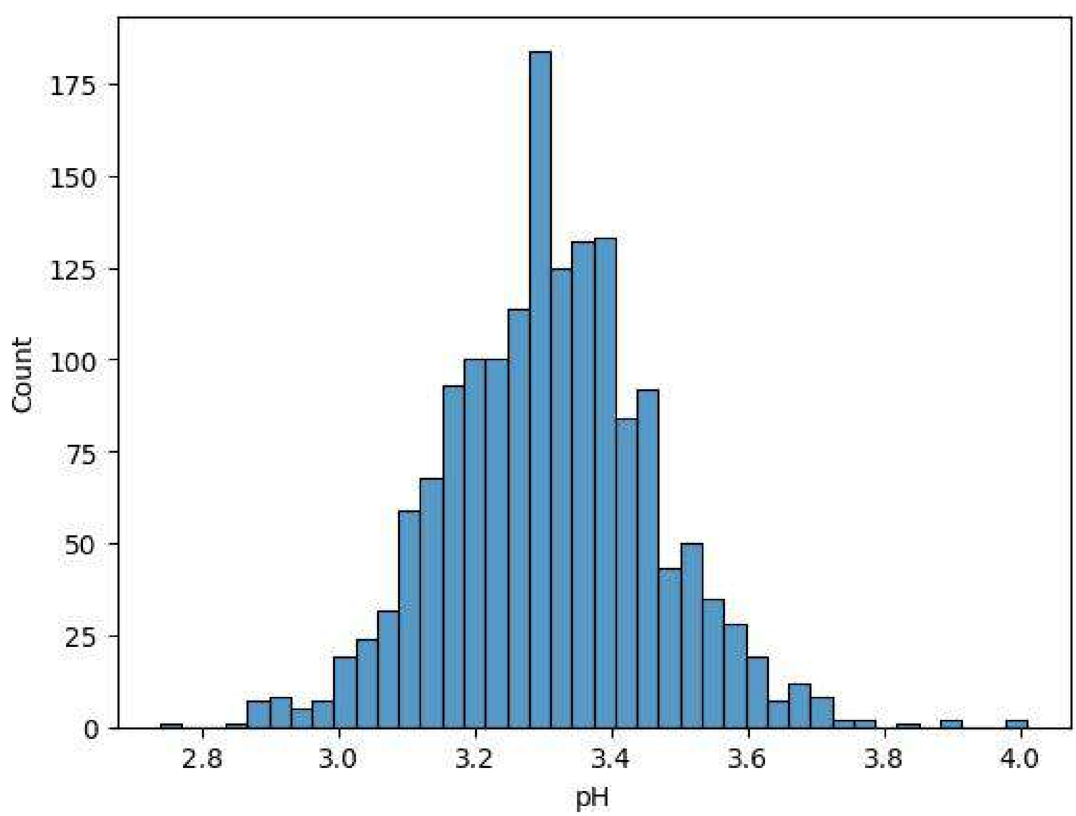
Count
chlorides
0
25
50
75
100
125
150
175
200
0.0
0.1
0.2
0.3
0.4
0.5
0.6


Count
free sulfur dioxide
0
50
100
150
200
250
0
10
20
30
40
50
60
70

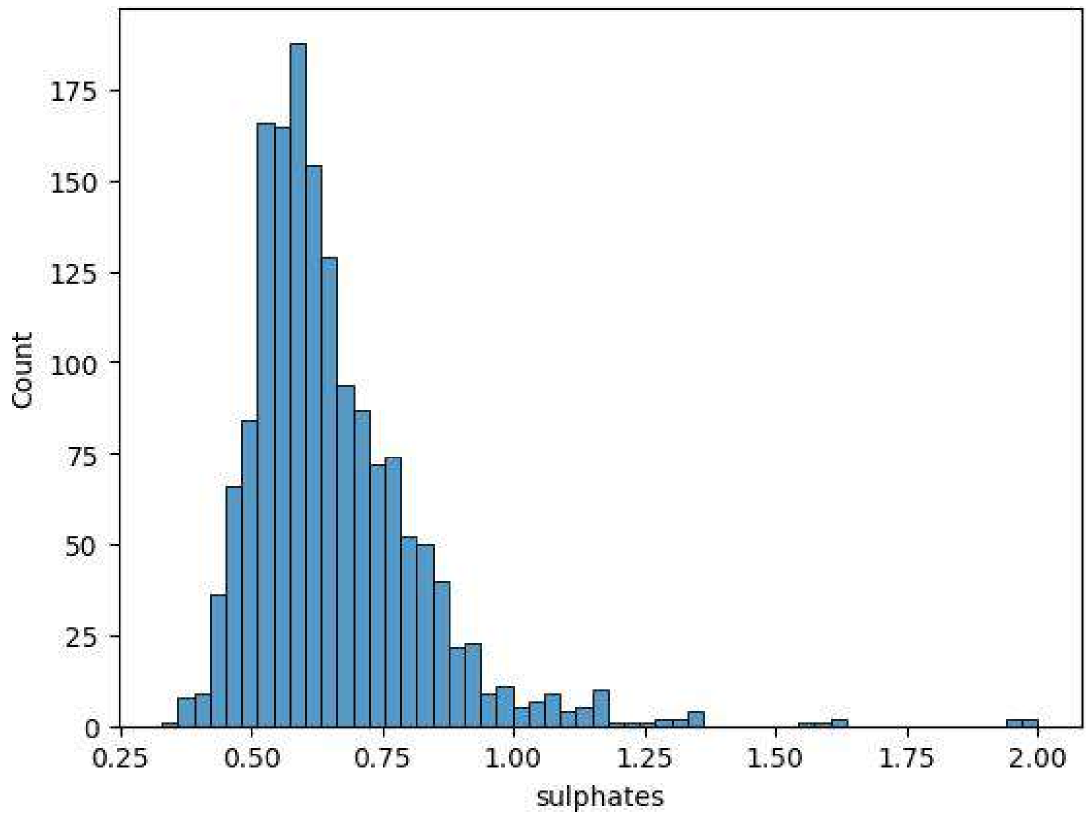

Count
total sulfur dioxide
0
25
50
75
100
125
150
175
200
0
50
100
150
200
250
300

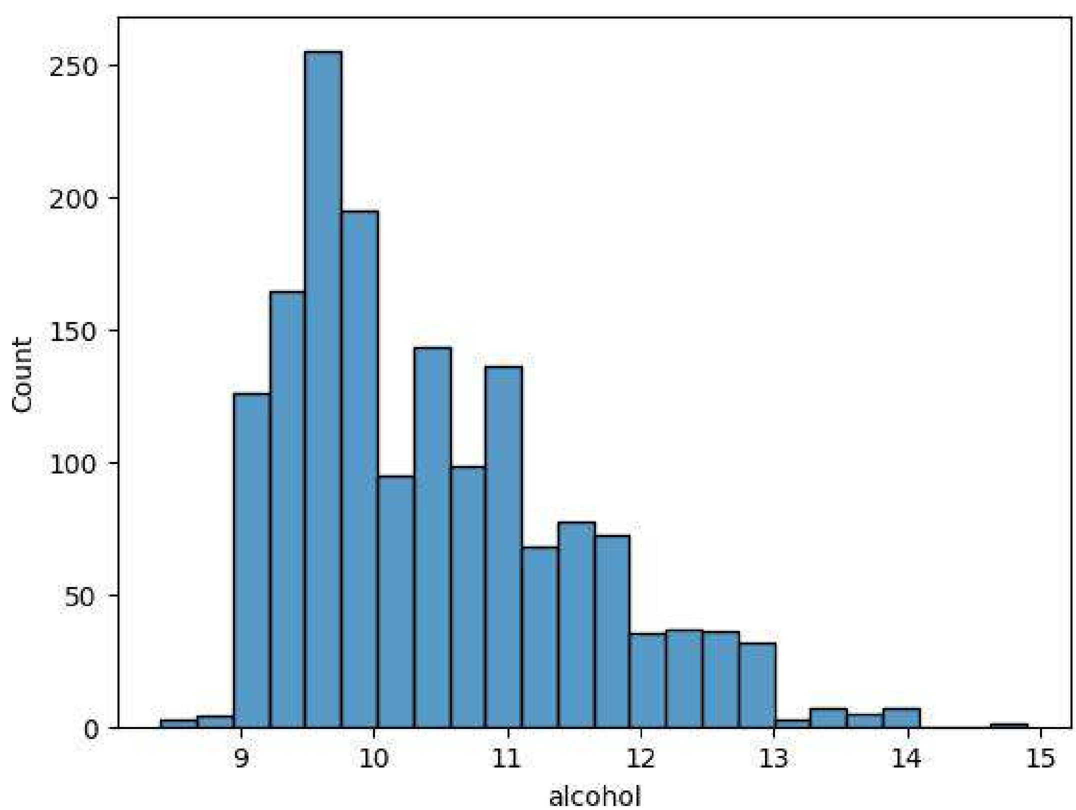


160
140
120
100
80
60
40
20
0
Count
0.990
0.992
0.994
0.996
0.998
1.000
1.002
1.004
density



175
150
125
100
75
50
25
0
Count
2.8
3.0
3.2
3.4
3.6
3.8
4.0
pH


Count
175
150
125
100
75
50
25
0
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
sulphates

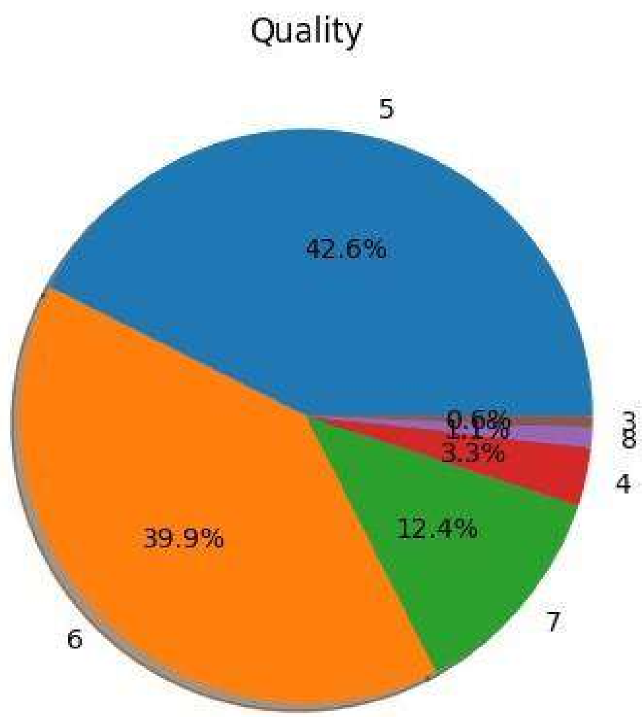

Count
250
200
150
100
50
0
9
10
11
12
13
14
15
alcohol

2) piechart

```
df['quality'].value_counts()
```

```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

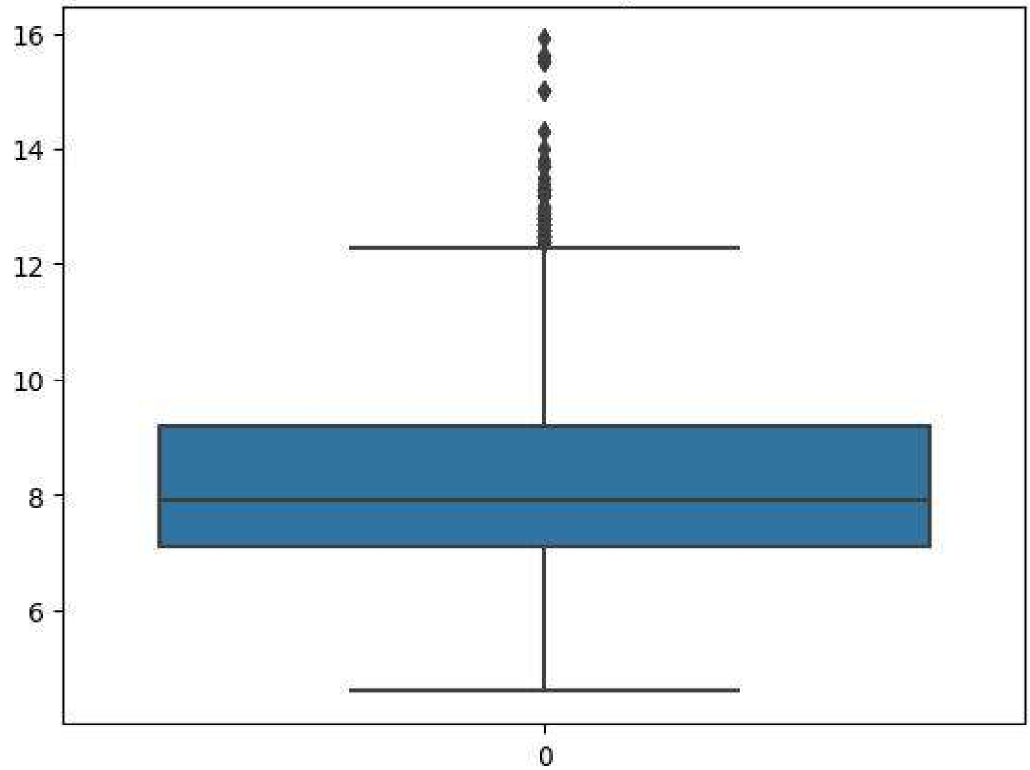
```
plt.pie(df['quality'].value_counts(),autopct ='%1.1f%%',shadow =
True,labels=df['quality'].unique())
plt.title('Quality')
plt.show()
```



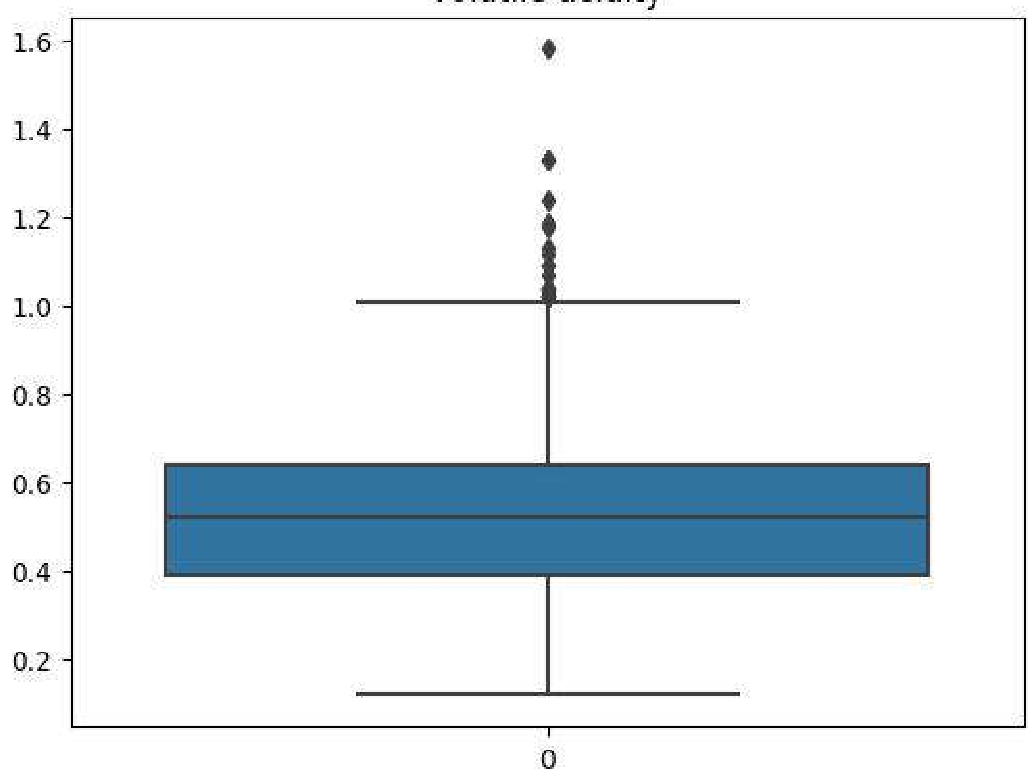
3) Boxplot

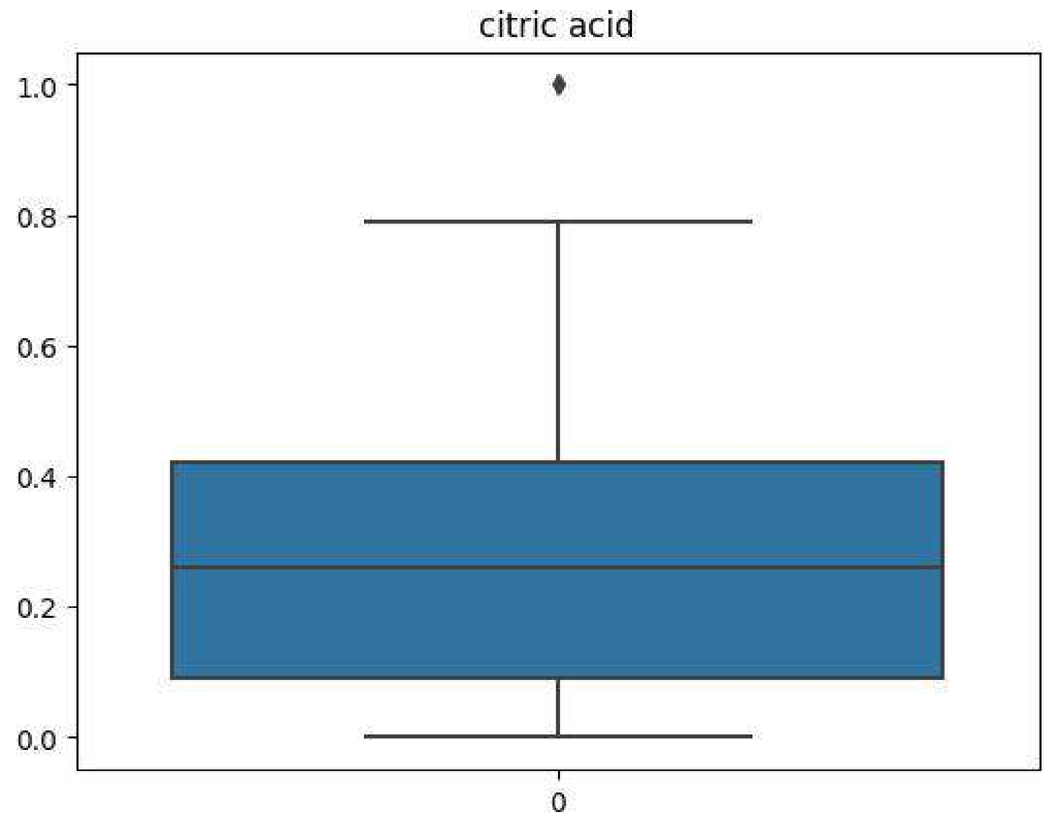
```
for i in df.columns[:-1]:
  sns.boxplot(df[i],)
  plt.title(i)
  plt.show()
```

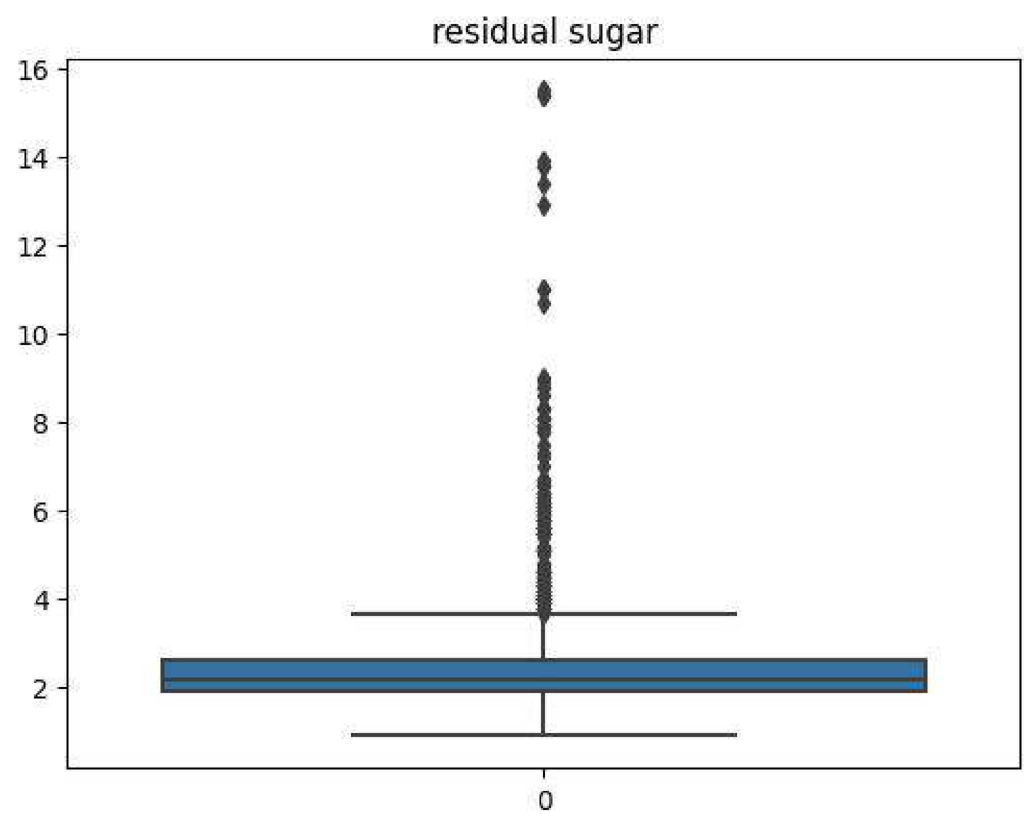
fixed acidity

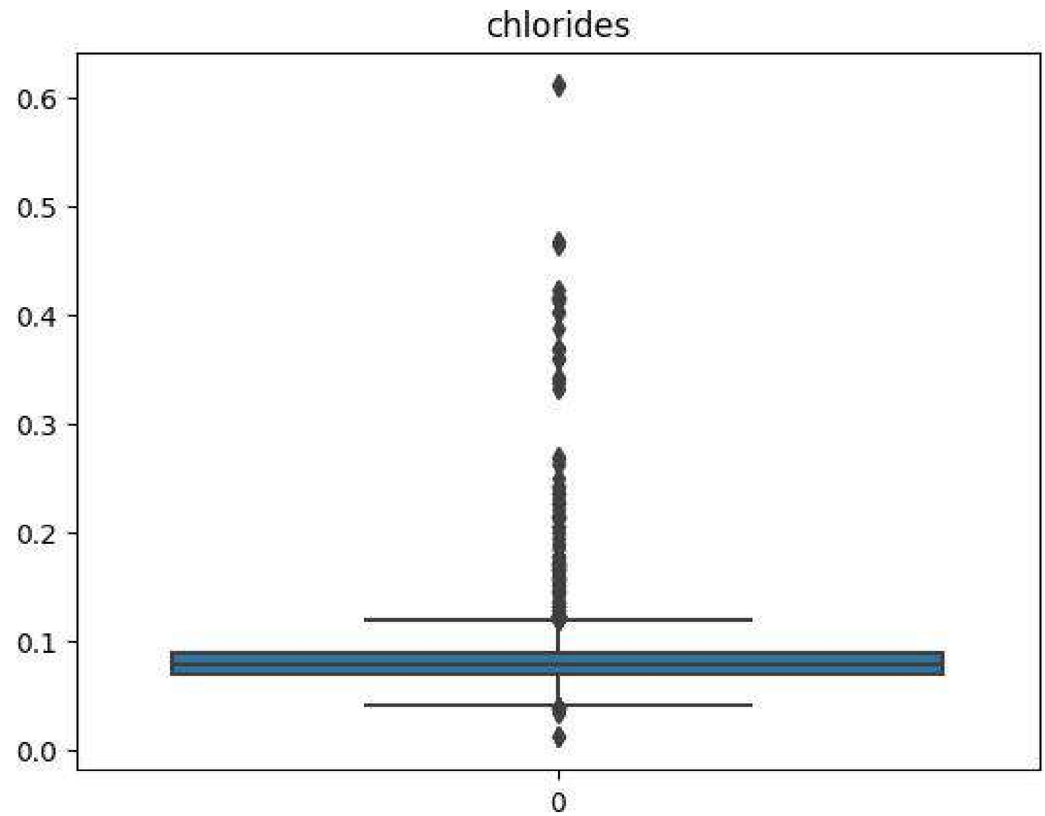


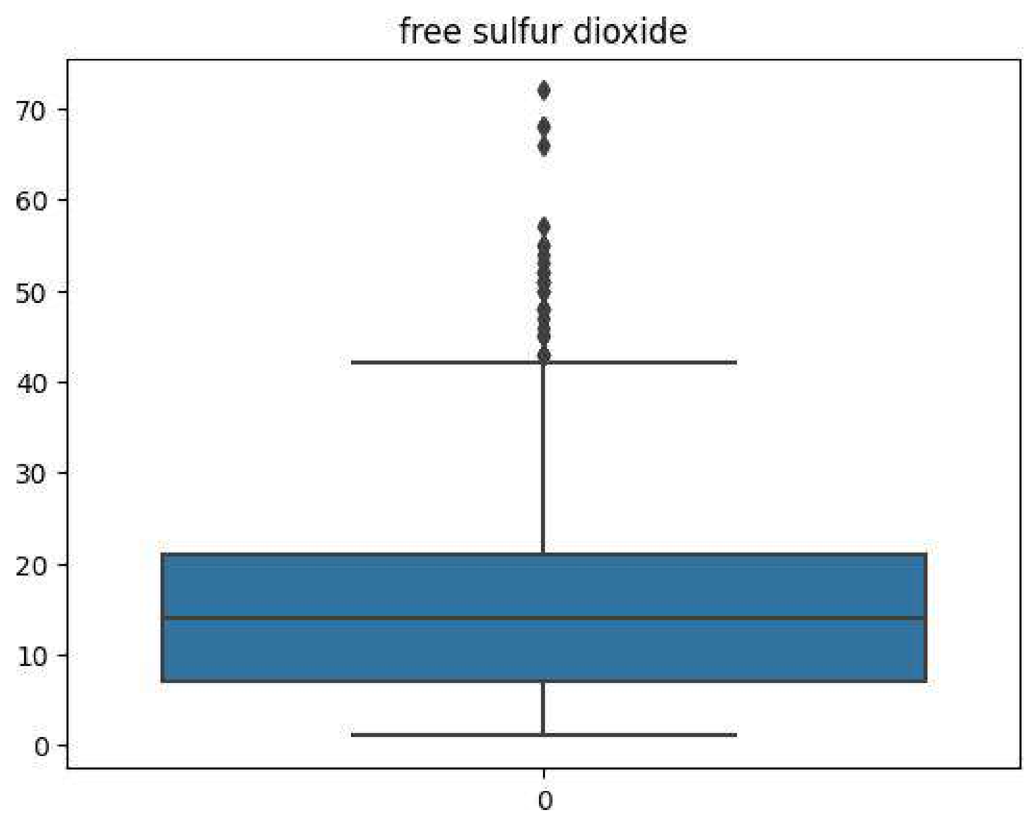
volatile acidity

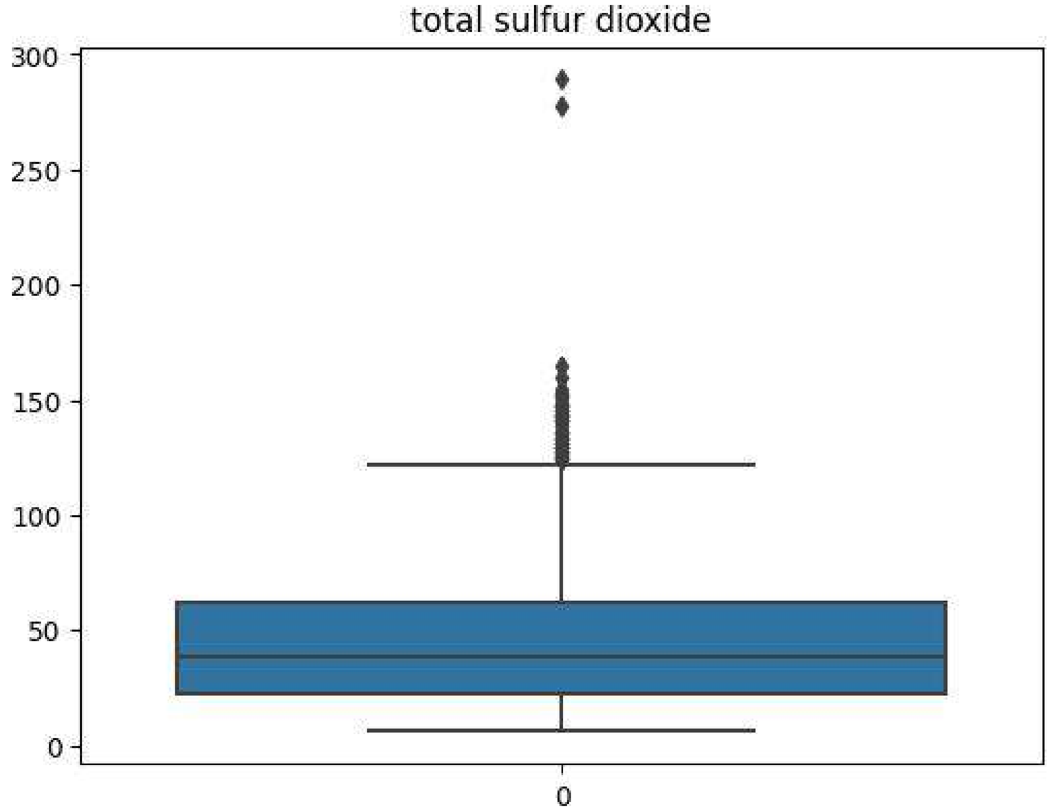


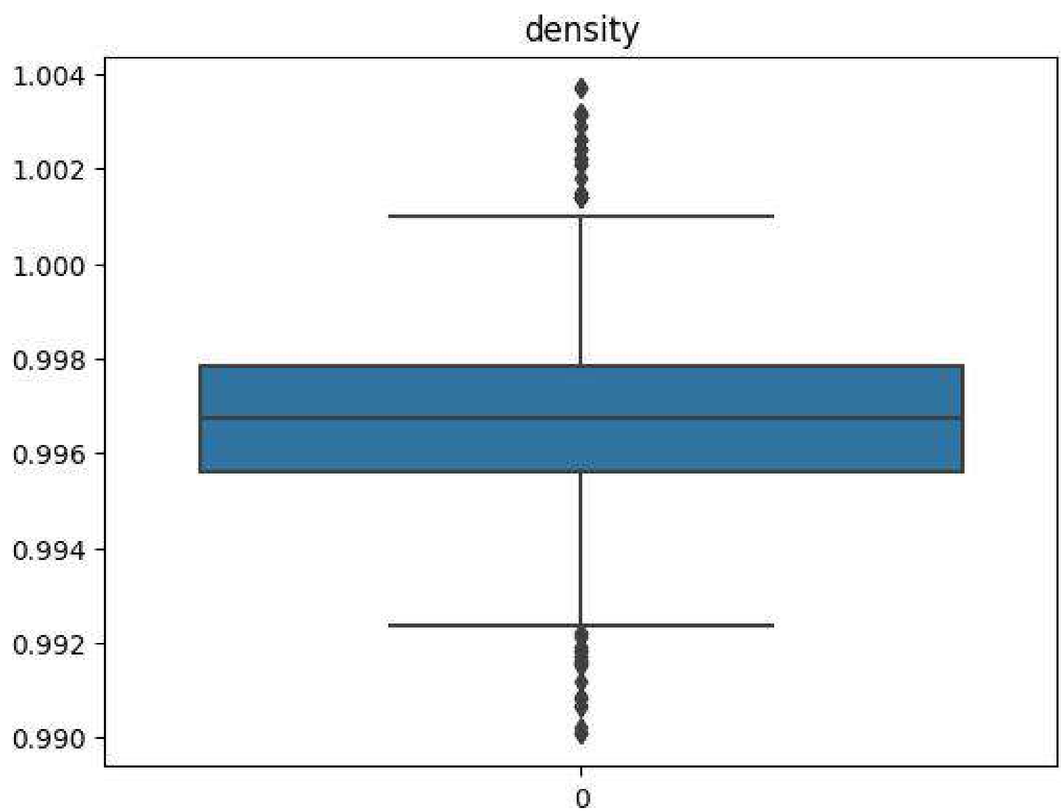

citric acid
1.0
0.8
0.6
0.4
0.2
0.0
0

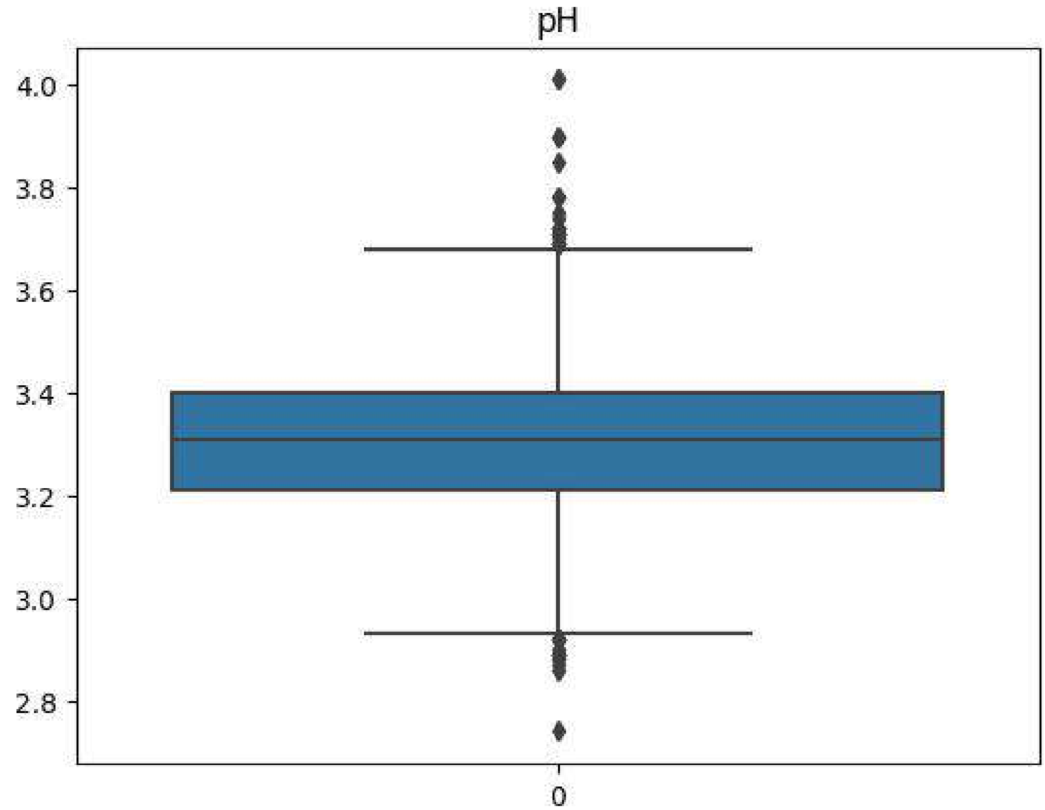

residual sugar
16
14
12
10
8
6
4
2
0

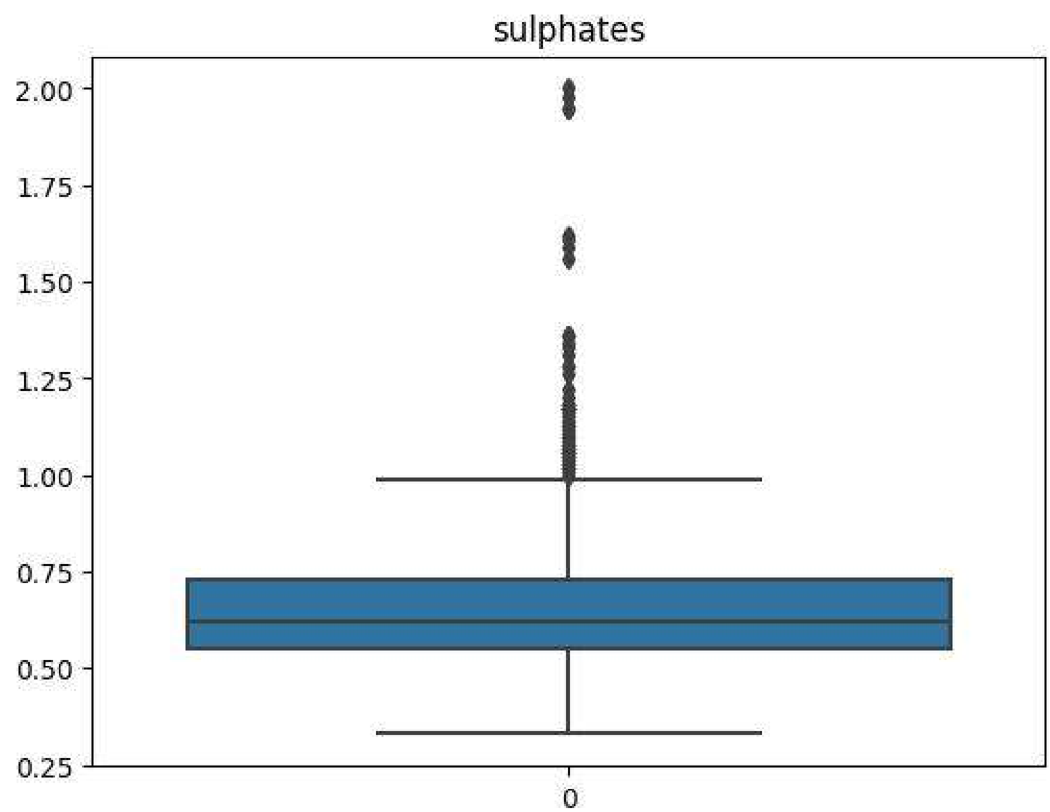

chlorides
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0

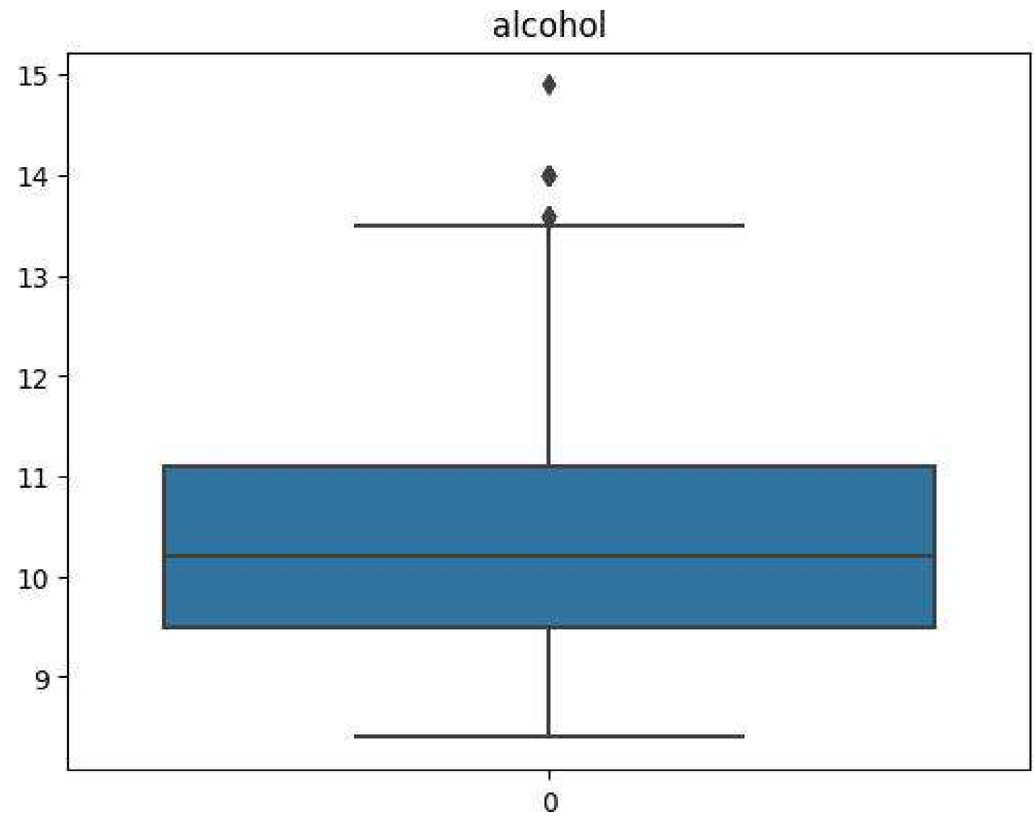

free sulfur dioxide
70
60
50
40
30
20
10
0
0


total sulfur dioxide
300
250
200
150
100
50
0
0


density
1.004
1.002
1.000
0.998
0.996
0.994
0.992
0.990
0


pH
4.0
3.8
3.6
3.4
3.2
3.0
2.8
0

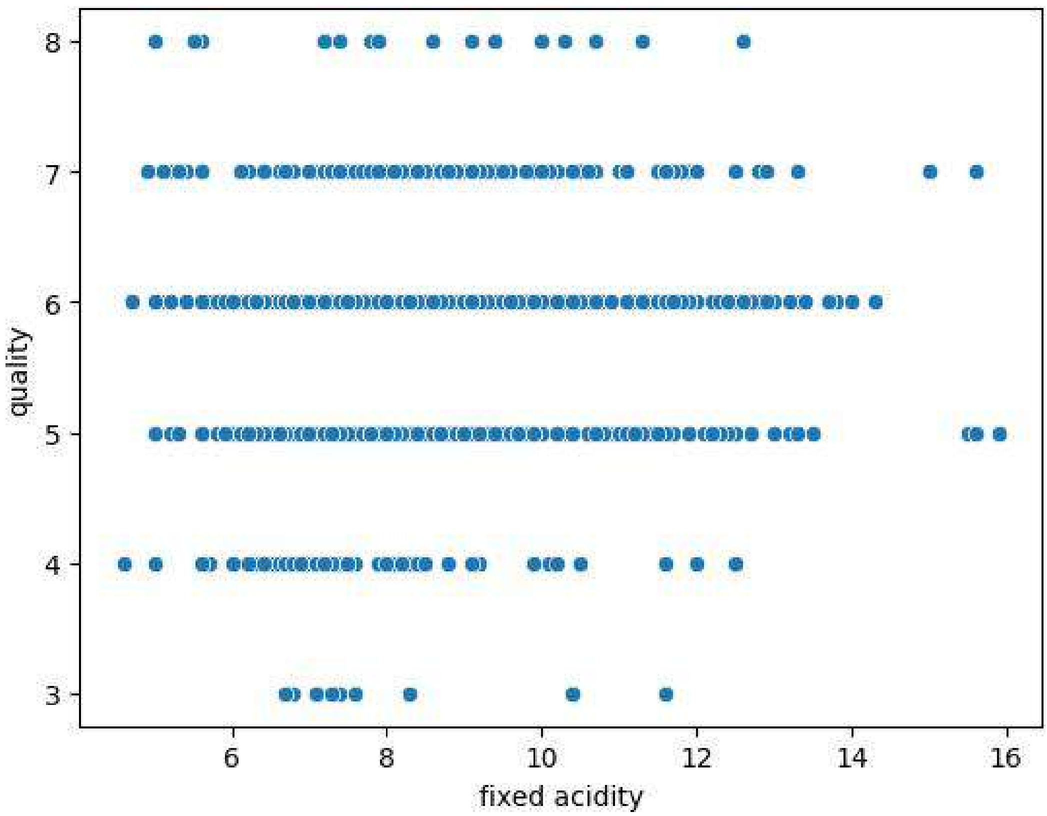

sulphates
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0



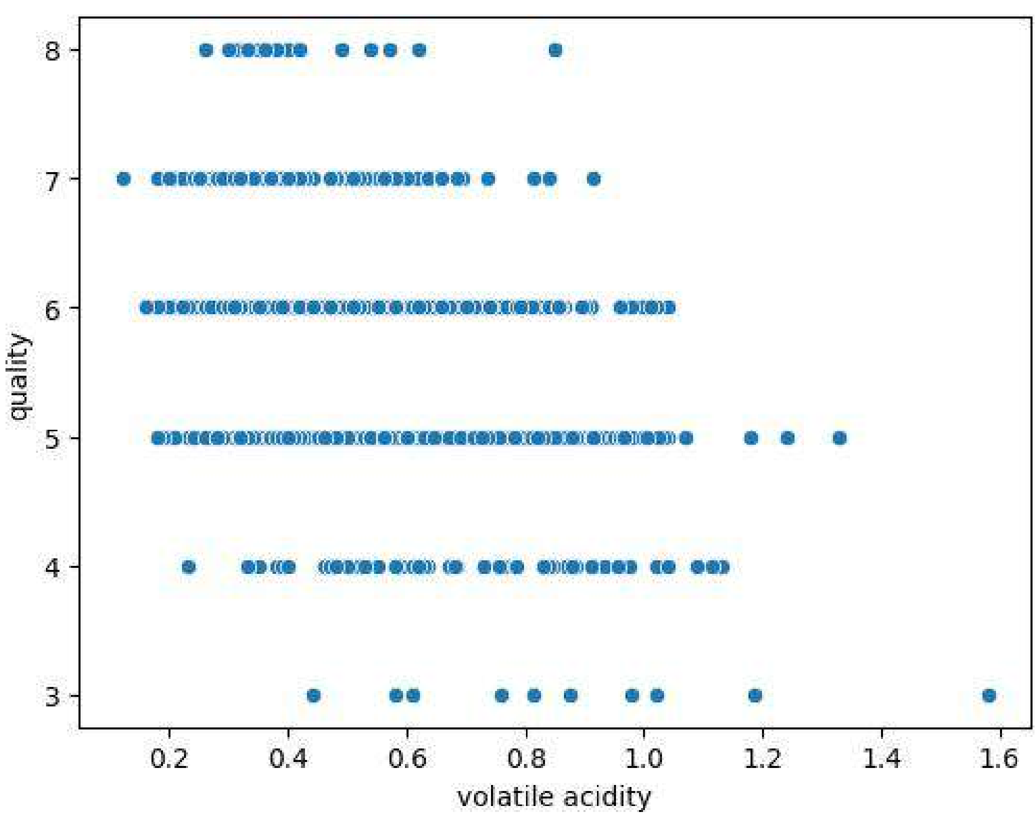
- Bivariate Analysis

1) ScatterPlot

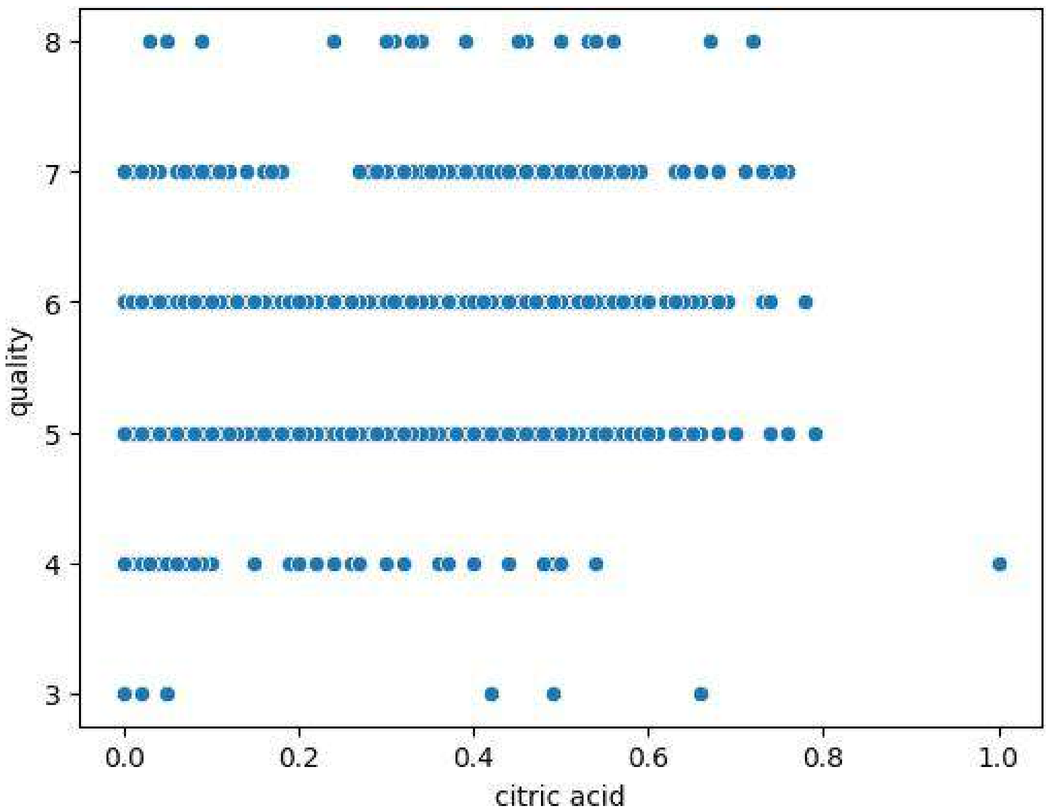
```
for i in df.columns[:-1]:
  sns.scatterplot(x=df[i],y=df['quality'])
  plt.show()
```



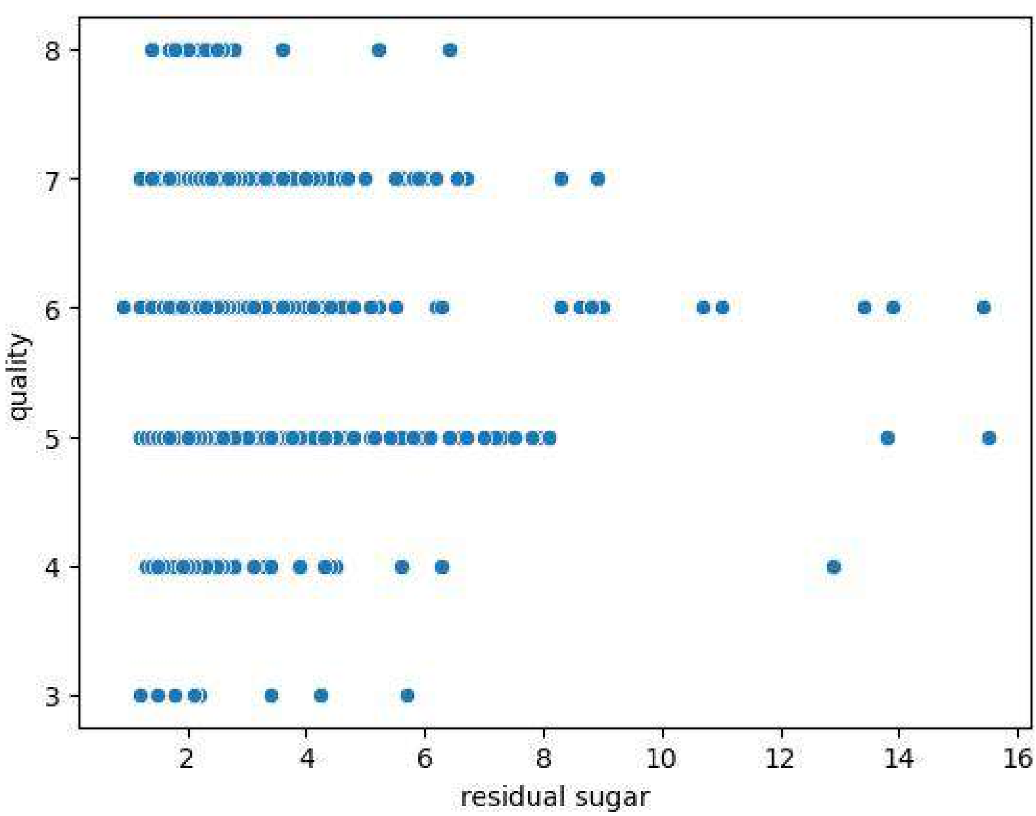
quality
8
7
6
5
4
3
6
8
10
12
14
16
fixed acidity



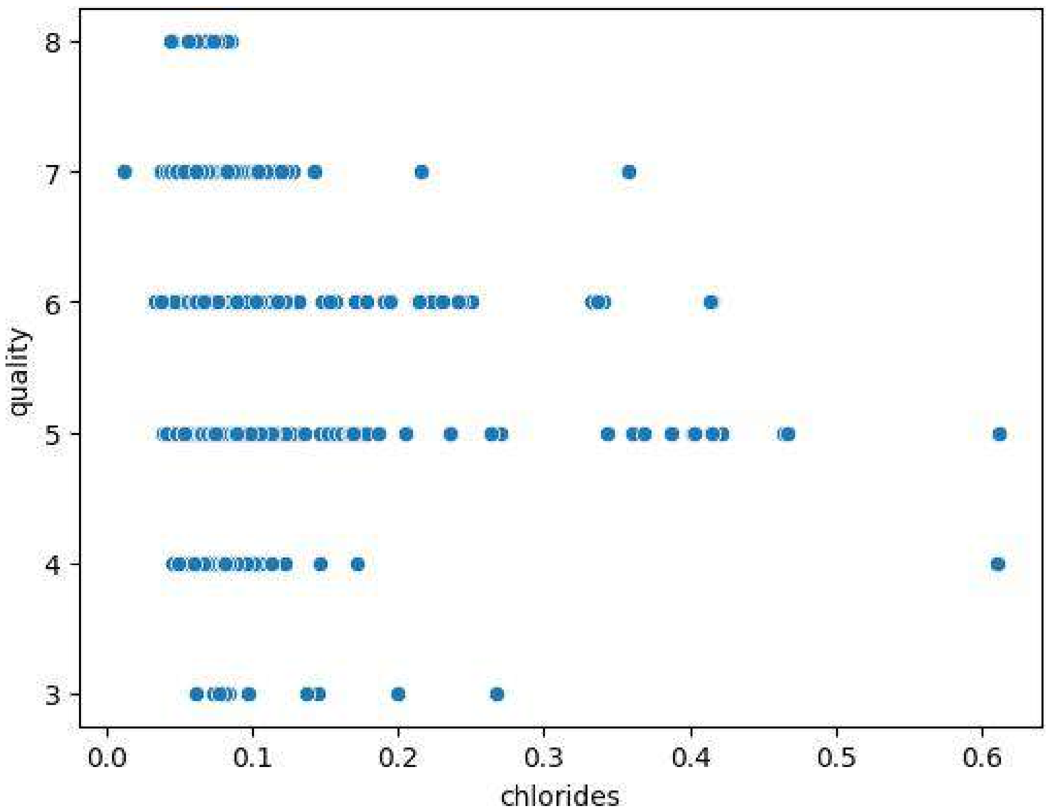
quality
8
7
6
5
4
3
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
volatile acidity

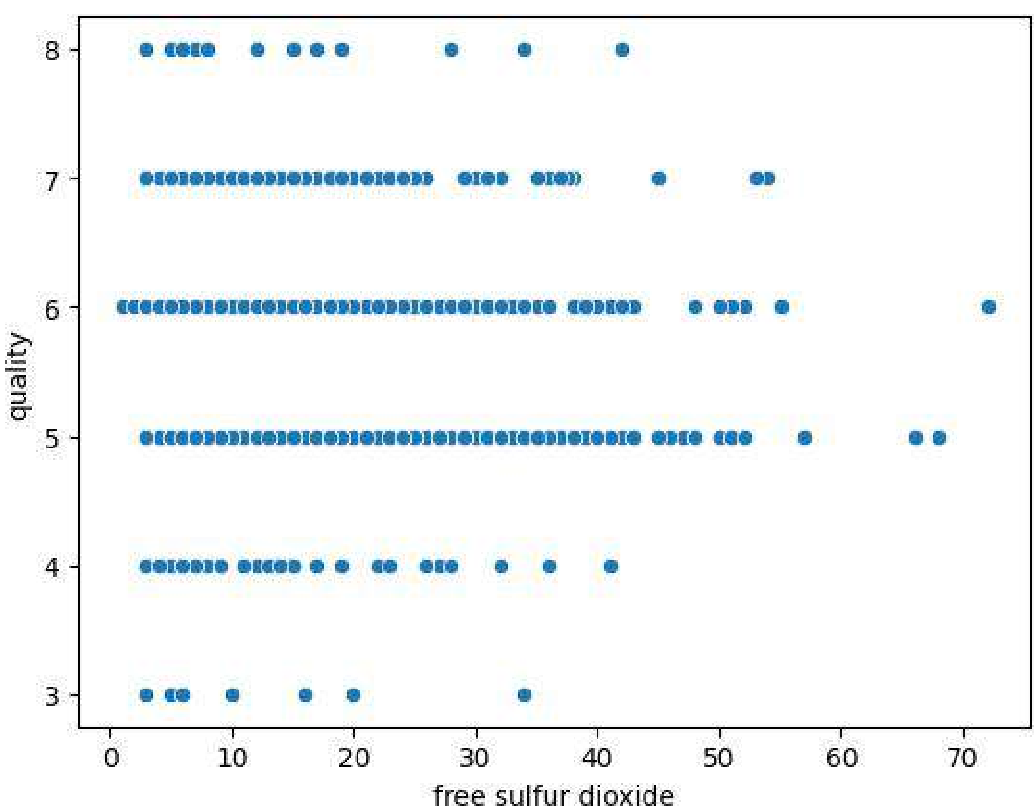


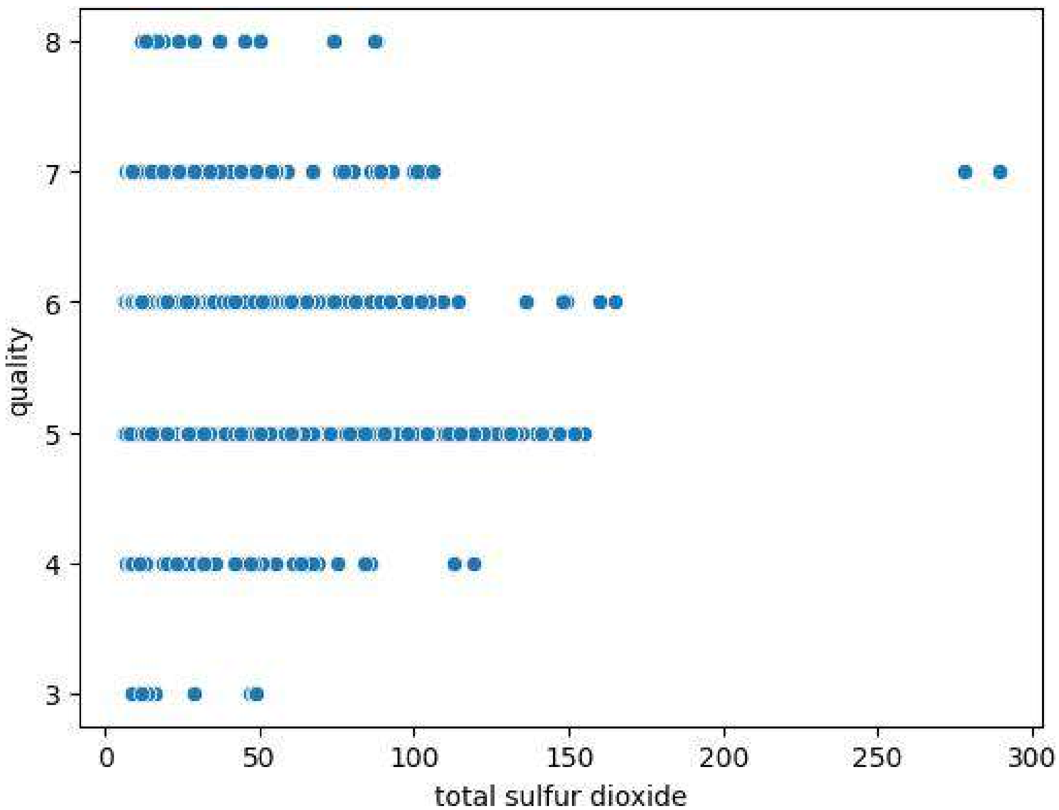
quality
citric acid
8
7
6
5
4
3
0.0
0.2
0.4
0.6
0.8
1.0

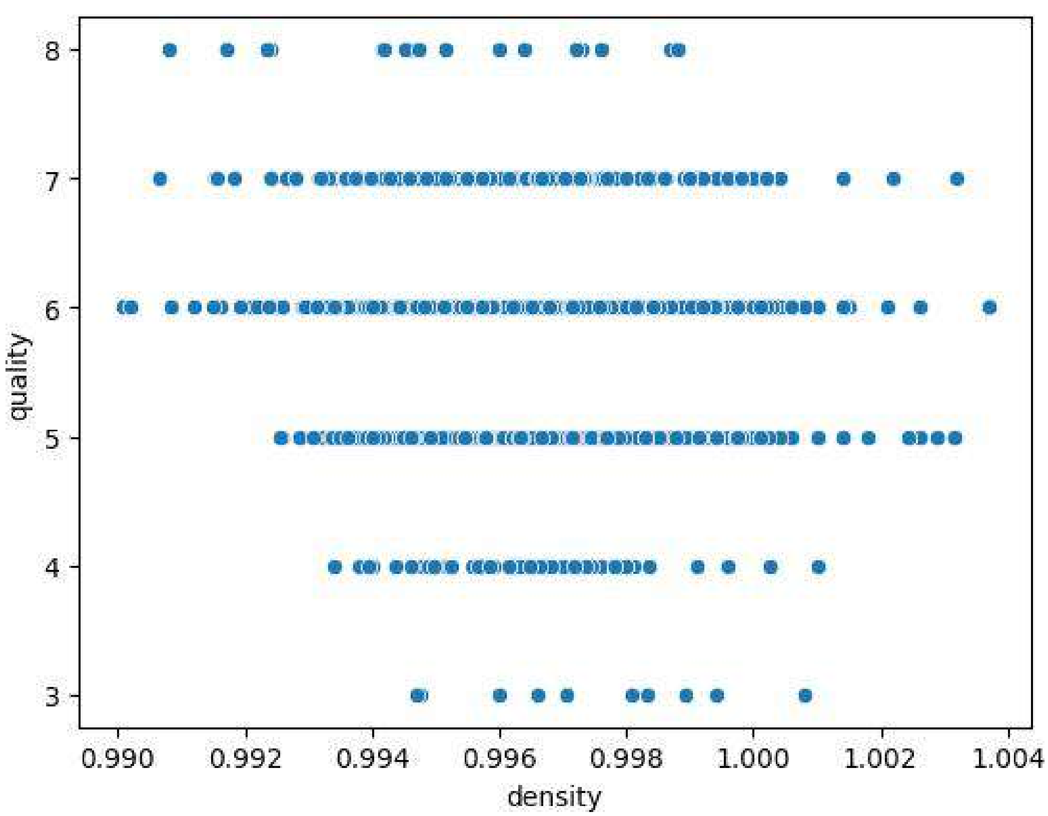


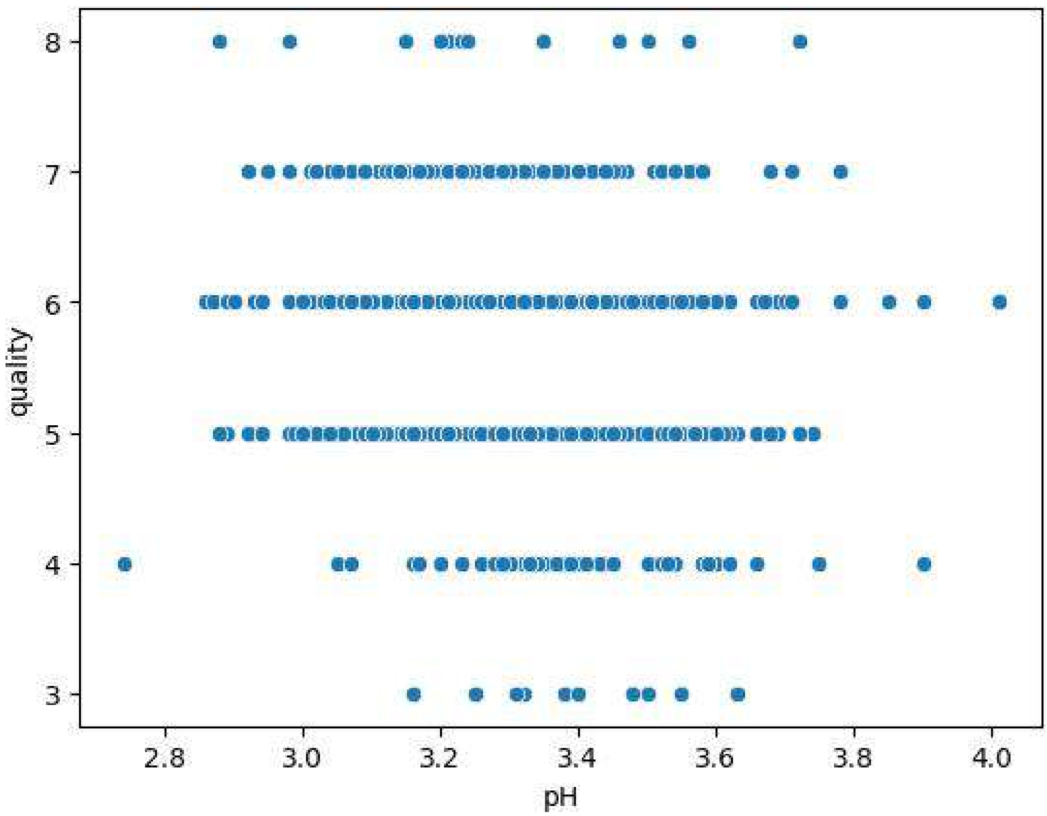
quality
residual sugar
8
7
6
5
4
3
2
4
6
8
10
12
14
16

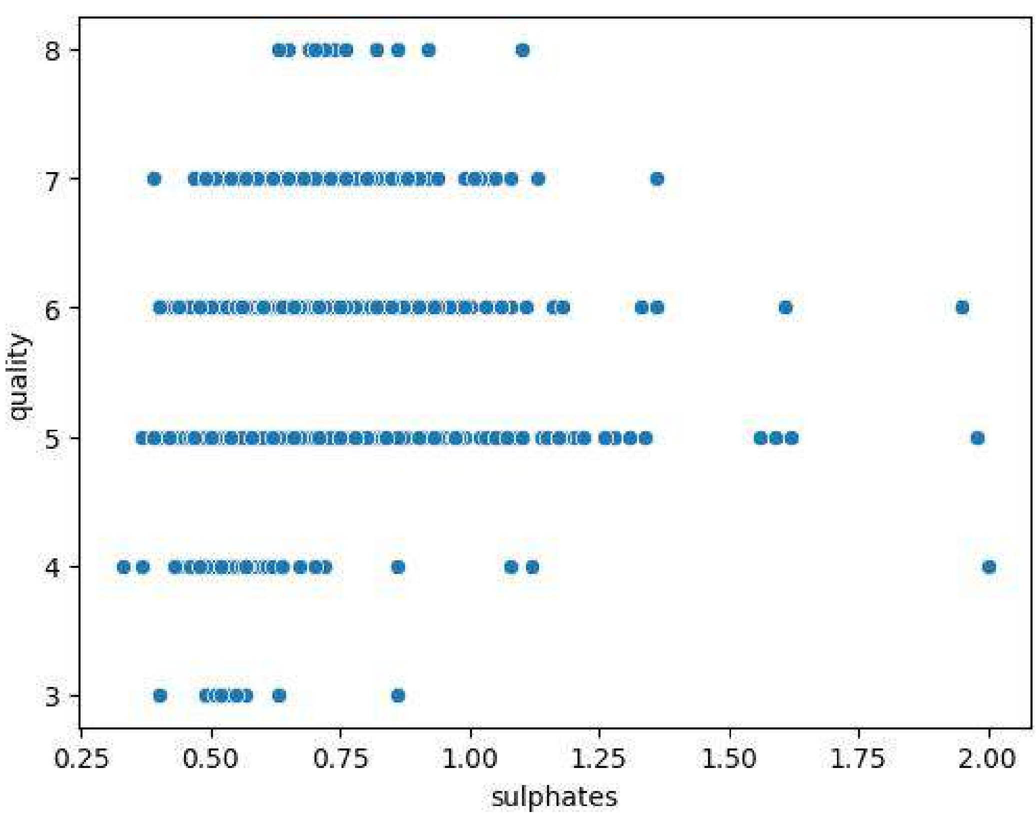

quality
8
7
6
5
4
3
0.0
0.1
0.2
0.3
0.4
0.5
0.6
chlorides

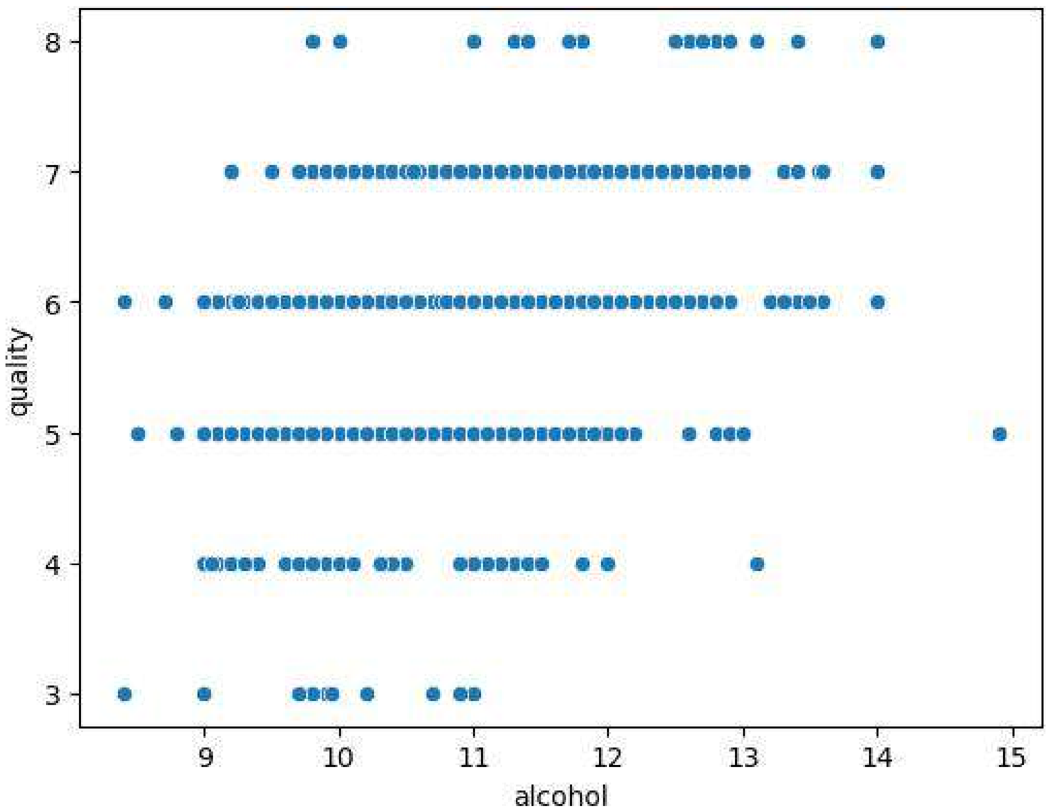

quality
8
7
6
5
4
3
0
10
20
30
40
50
60
70
free sulfur dioxide


quality
8
7
6
5
4
3
0
50
100
150
200
250
300
total sulfur dioxide


quality
8
7
6
5
4
3
0.990
0.992
0.994
0.996
0.998
1.000
1.002
1.004
density

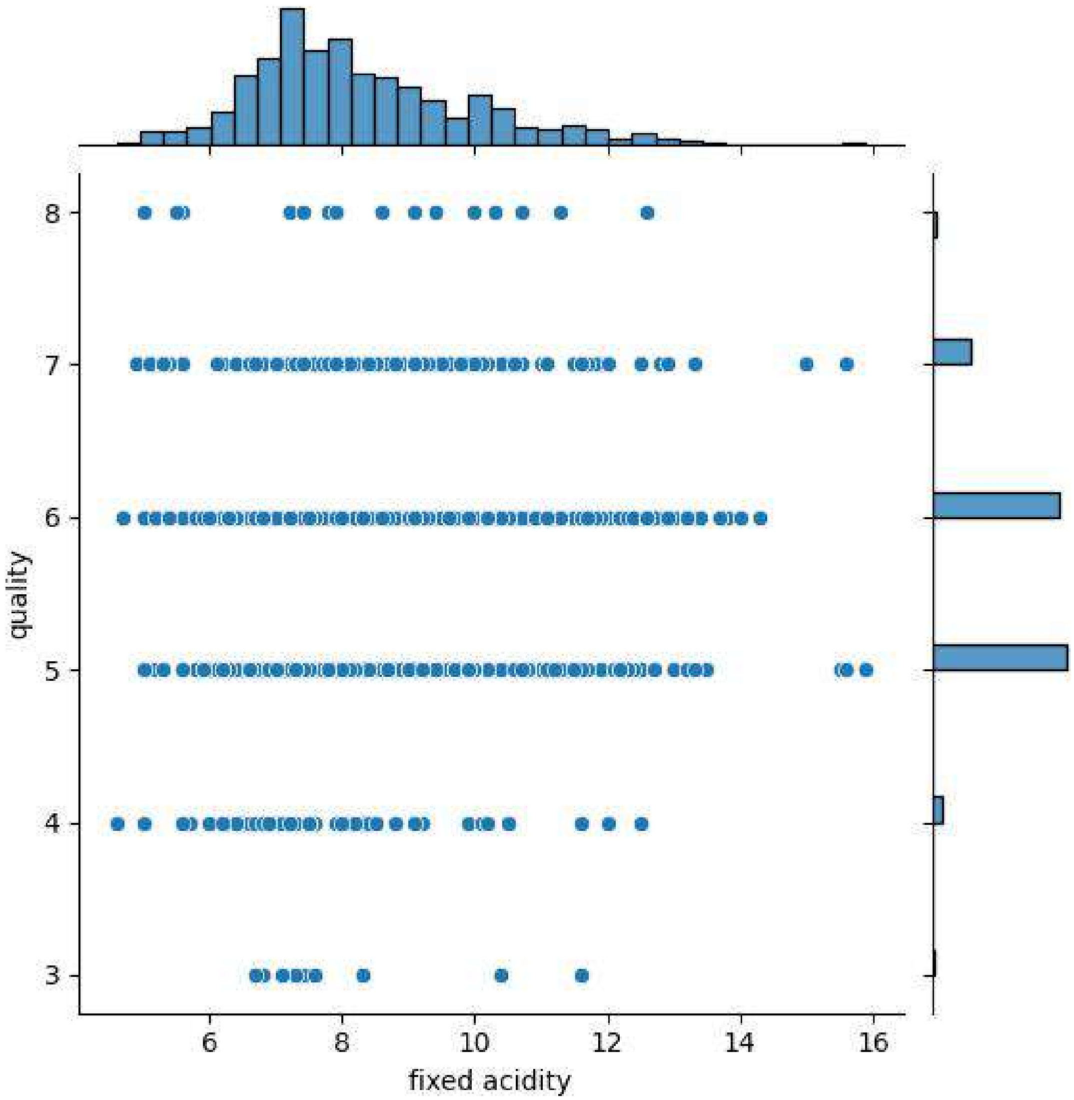

quality
pH
8
7
6
5
4
3
2.8
3.0
3.2
3.4
3.6
3.8
4.0


quality
sulphates
8
7
6
5
4
3
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

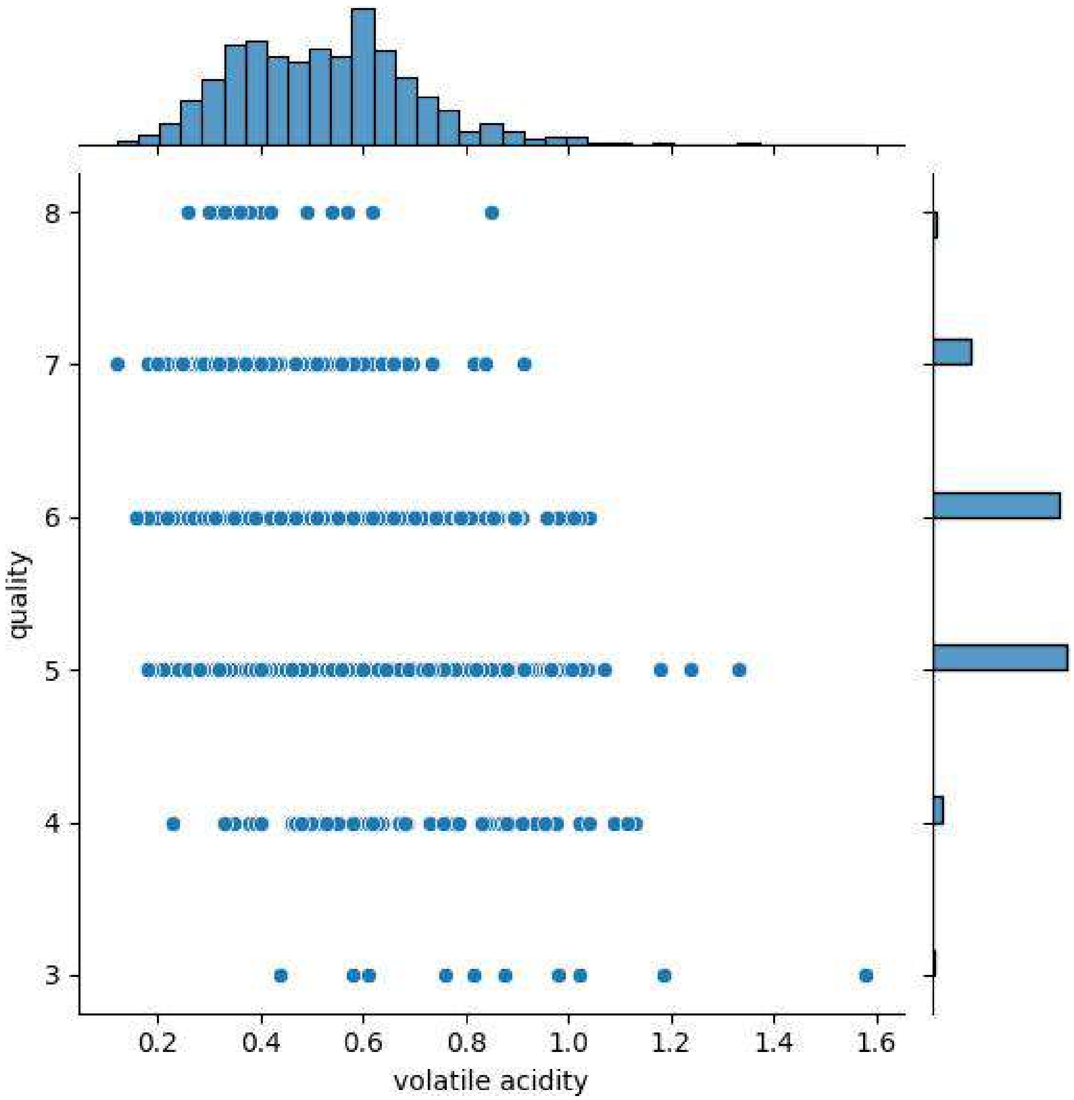


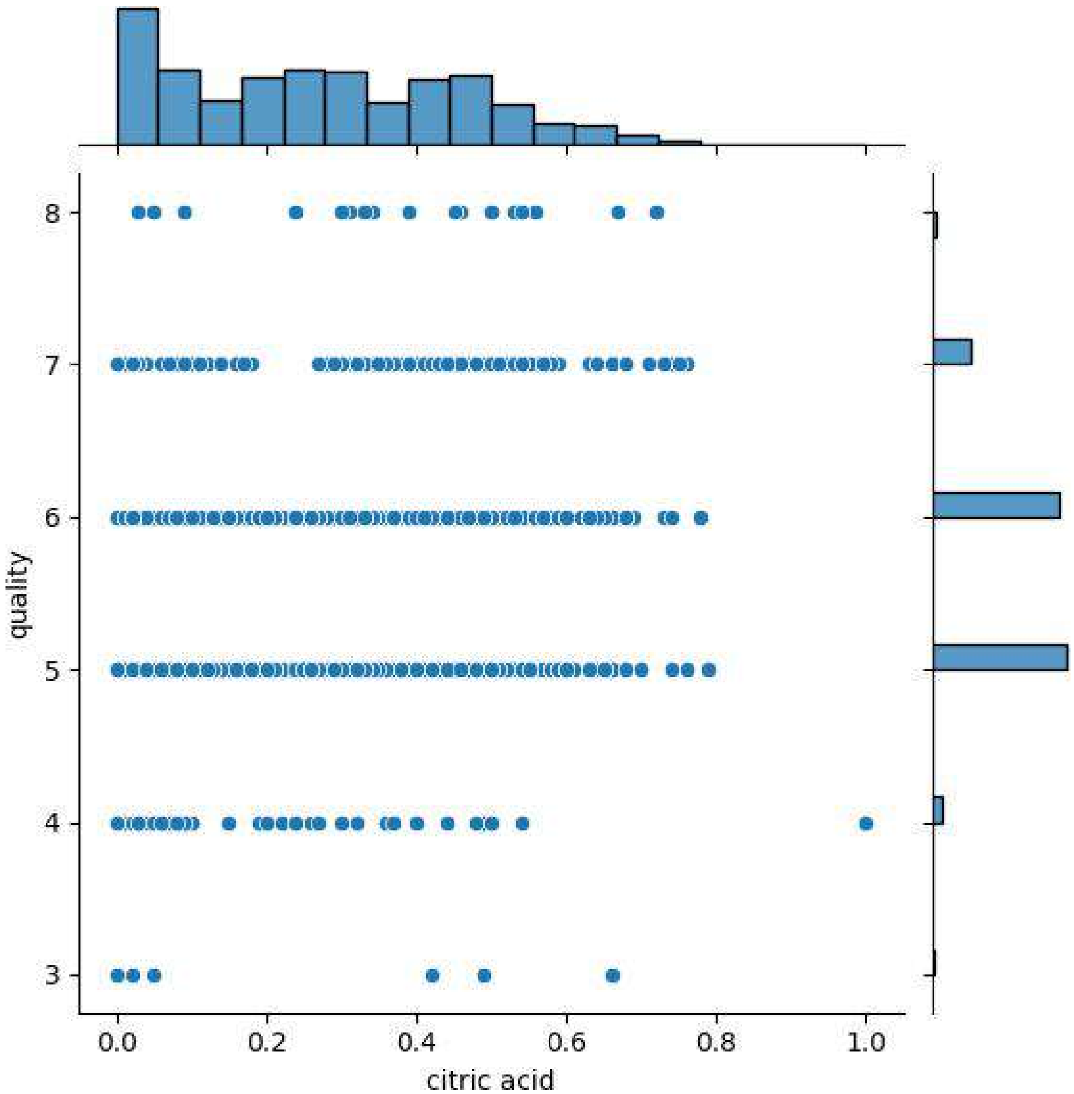
2) JointPlot

```
for i in df.columns[:-1]:
  sns.jointplot(x=df[i],y=df['quality'])
  plt.show()
```

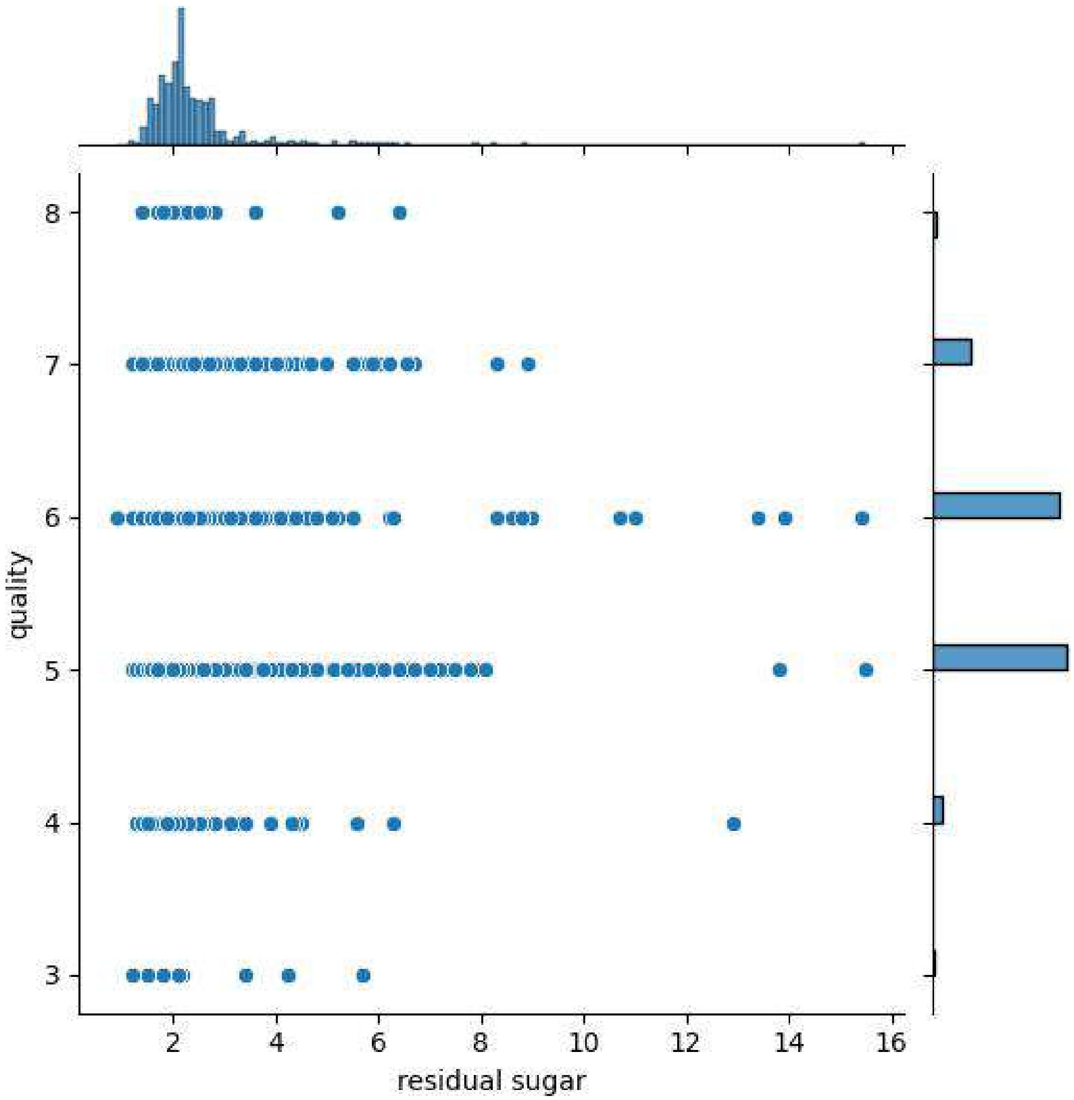


8
7
6
5
4
3
quality
6
8
10
12
14
16
fixed acidity

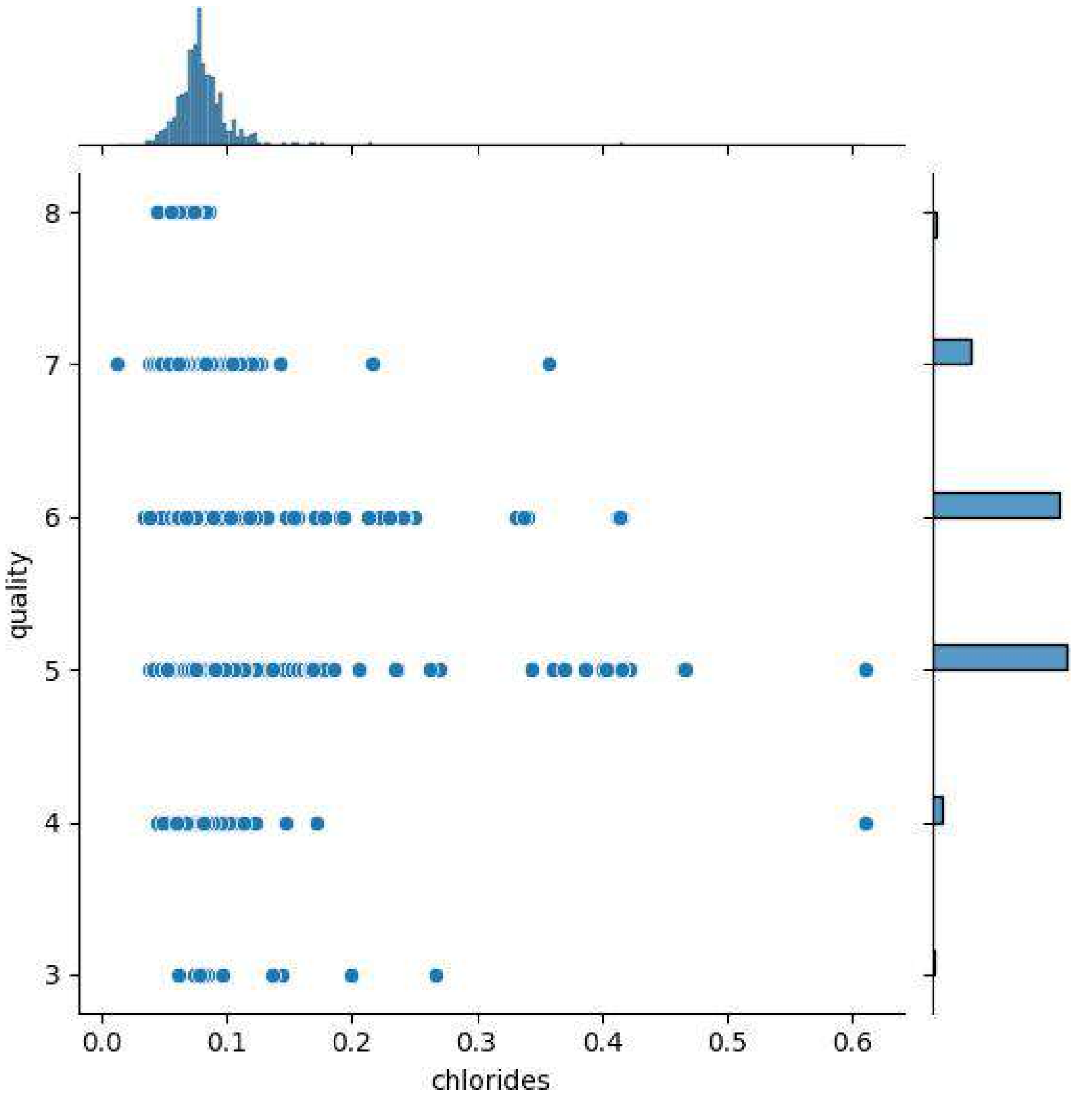

quality
volatile acidity
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
3
4
5
6
7
8



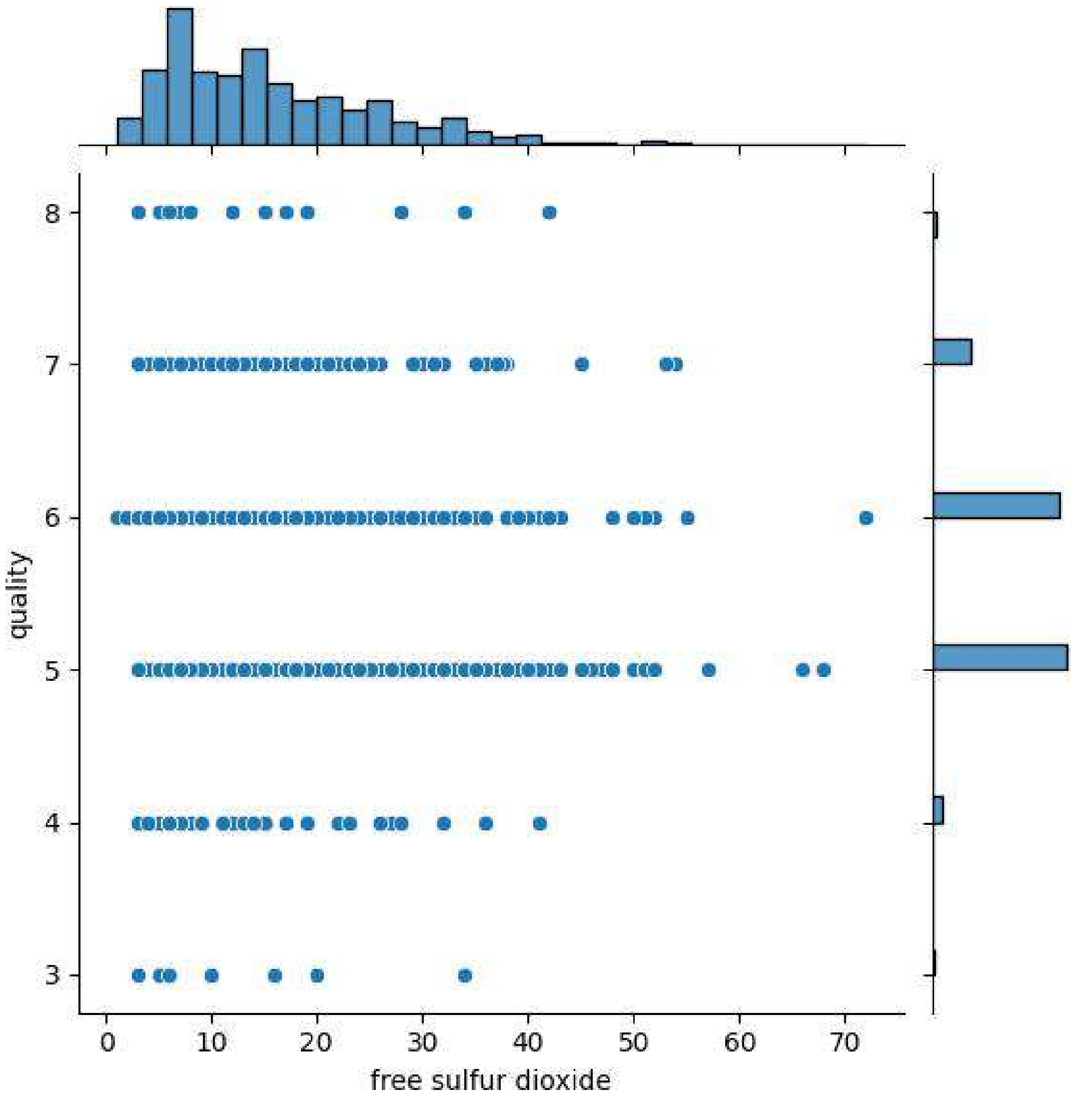
quality
citric acid
0.0
0.2
0.4
0.6
0.8
1.0
3
4
5
6
7
8



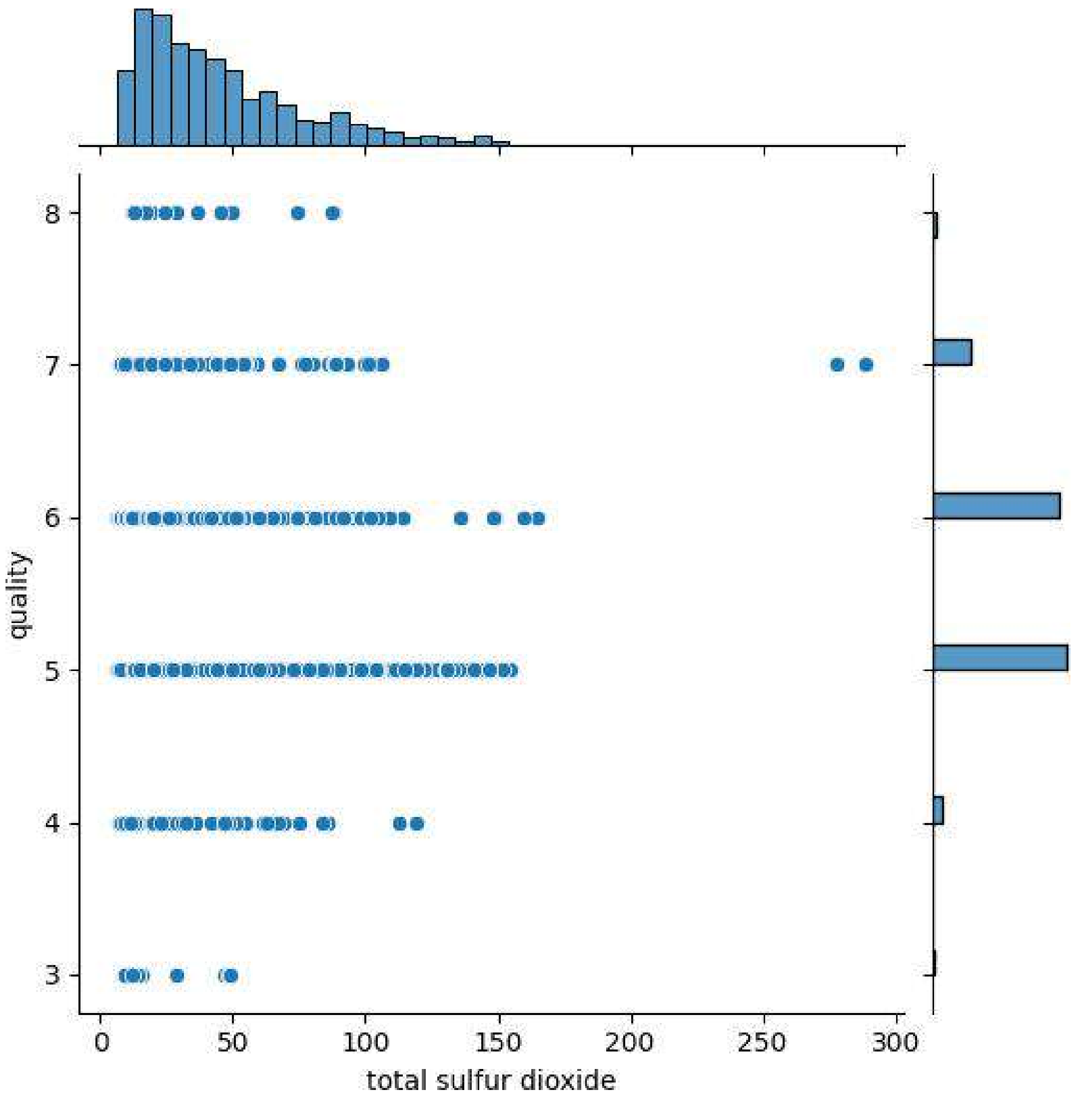
8
7
6
5
4
3
quality
2
4
6
8
10
12
14
16
residual sugar



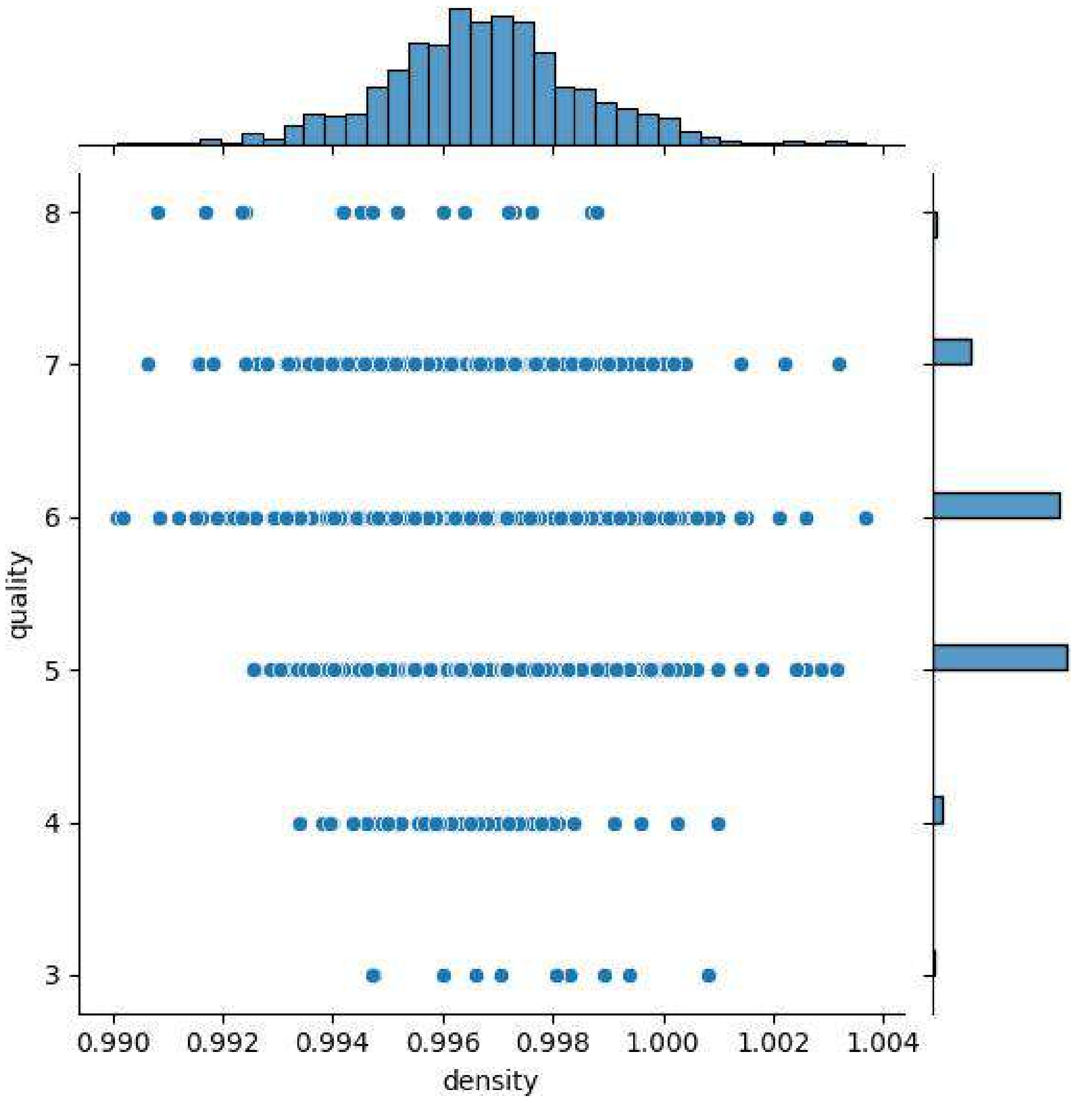
8
7
6
5
4
3
quality
0.0
0.1
0.2
0.3
0.4
0.5
0.6
chlorides



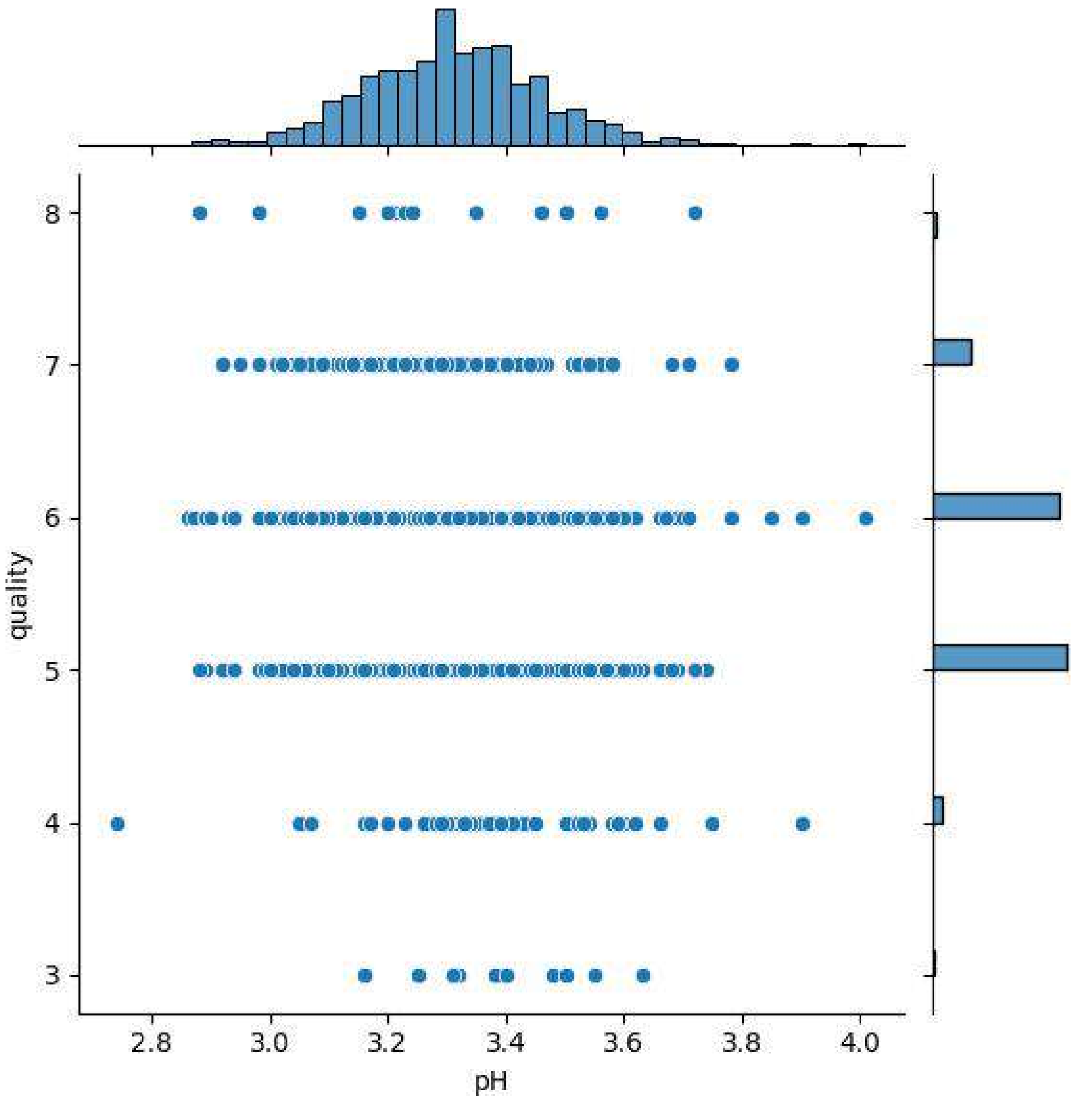
8
7
6
5
4
3
quality
0
10
20
30
40
50
60
70
free sulfur dioxide



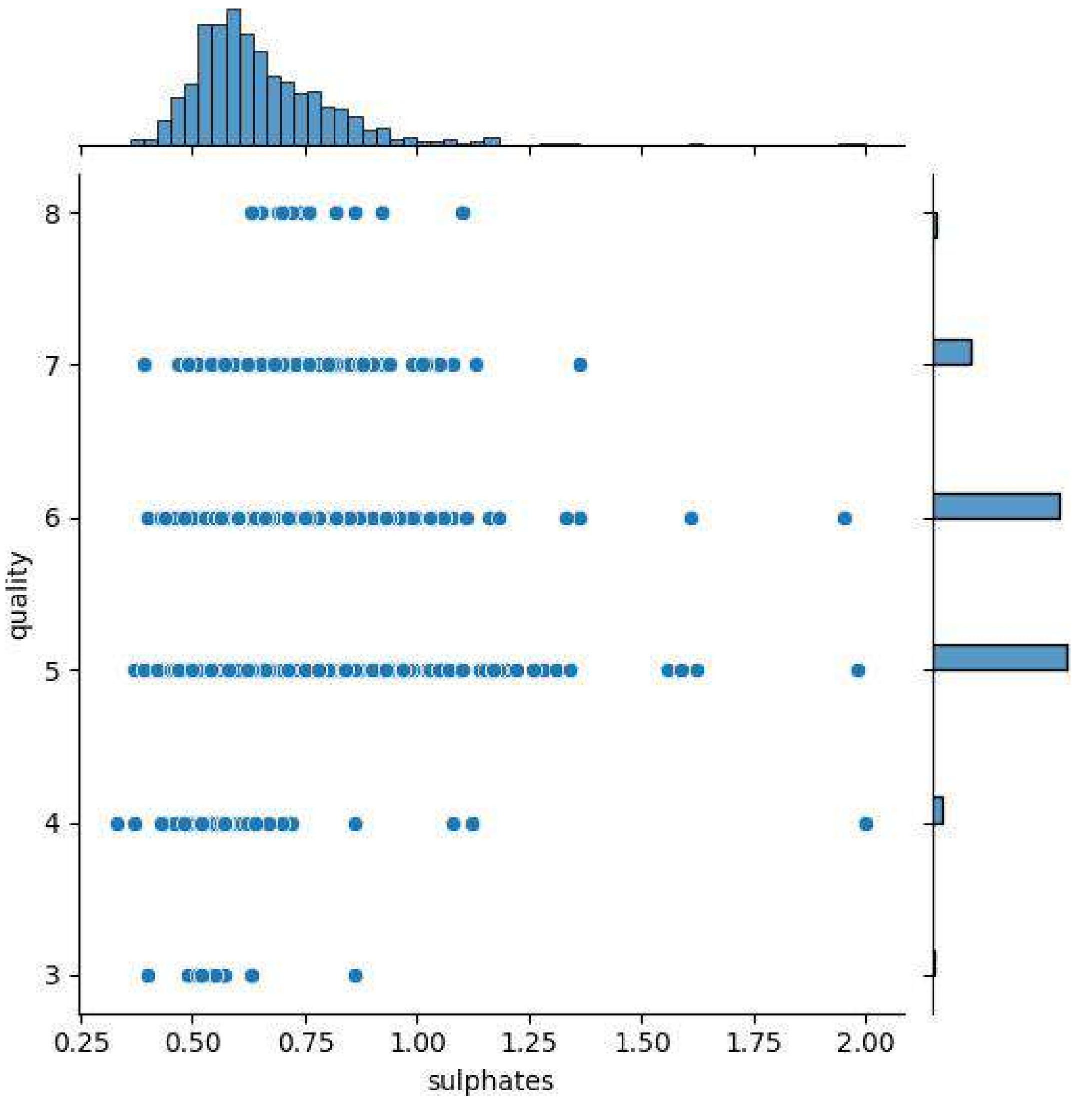
8
7
6
5
4
3
quality
0
50
100
150
200
250
300
total sulfur dioxide

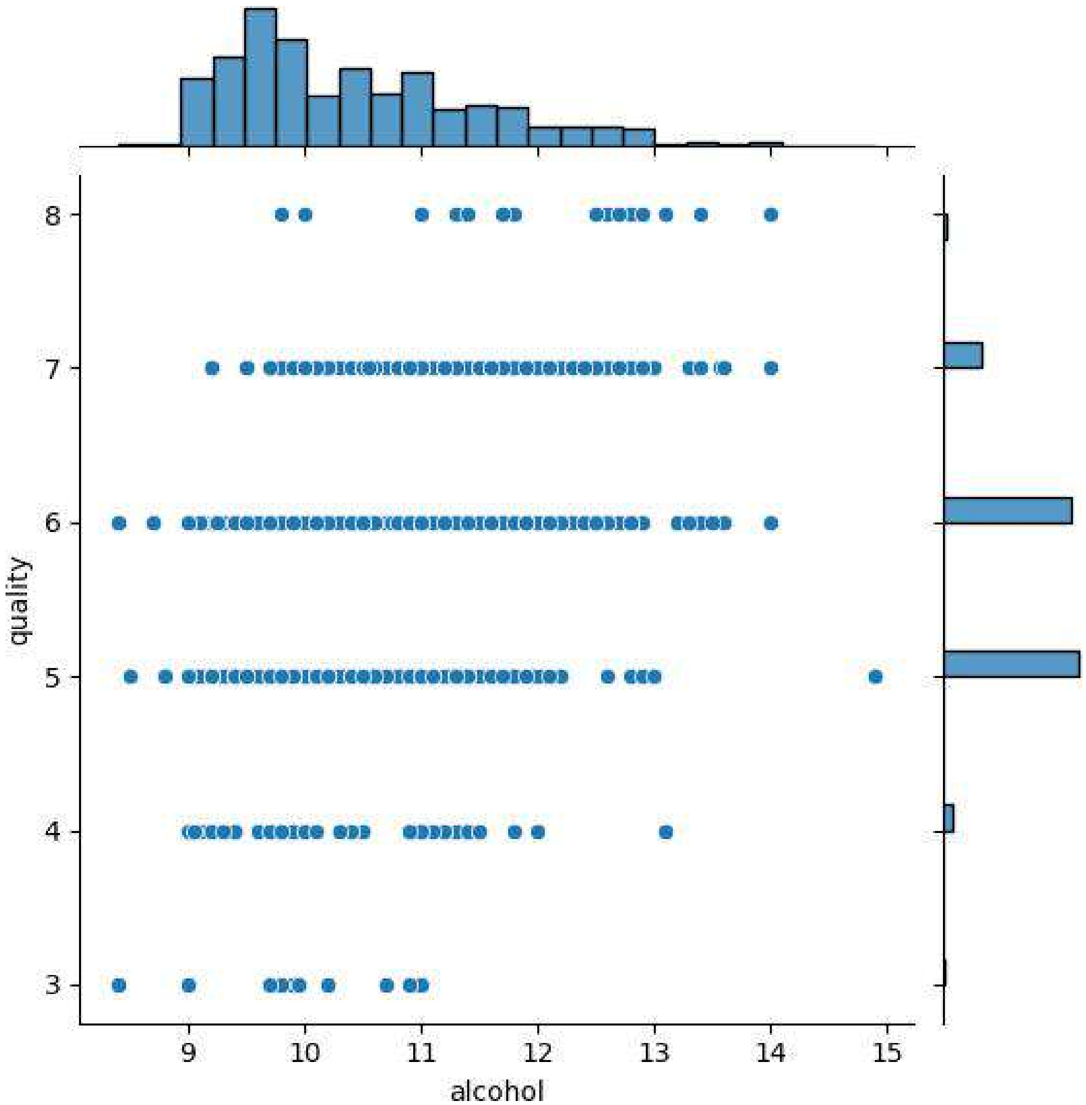


8
7
6
5
4
3
quality
0.990
0.992
0.994
0.996
0.998
1.000
1.002
1.004
density



8
7
6
5
4
3
quality
2.8
3.0
3.2
3.4
3.6
3.8
4.0
pH

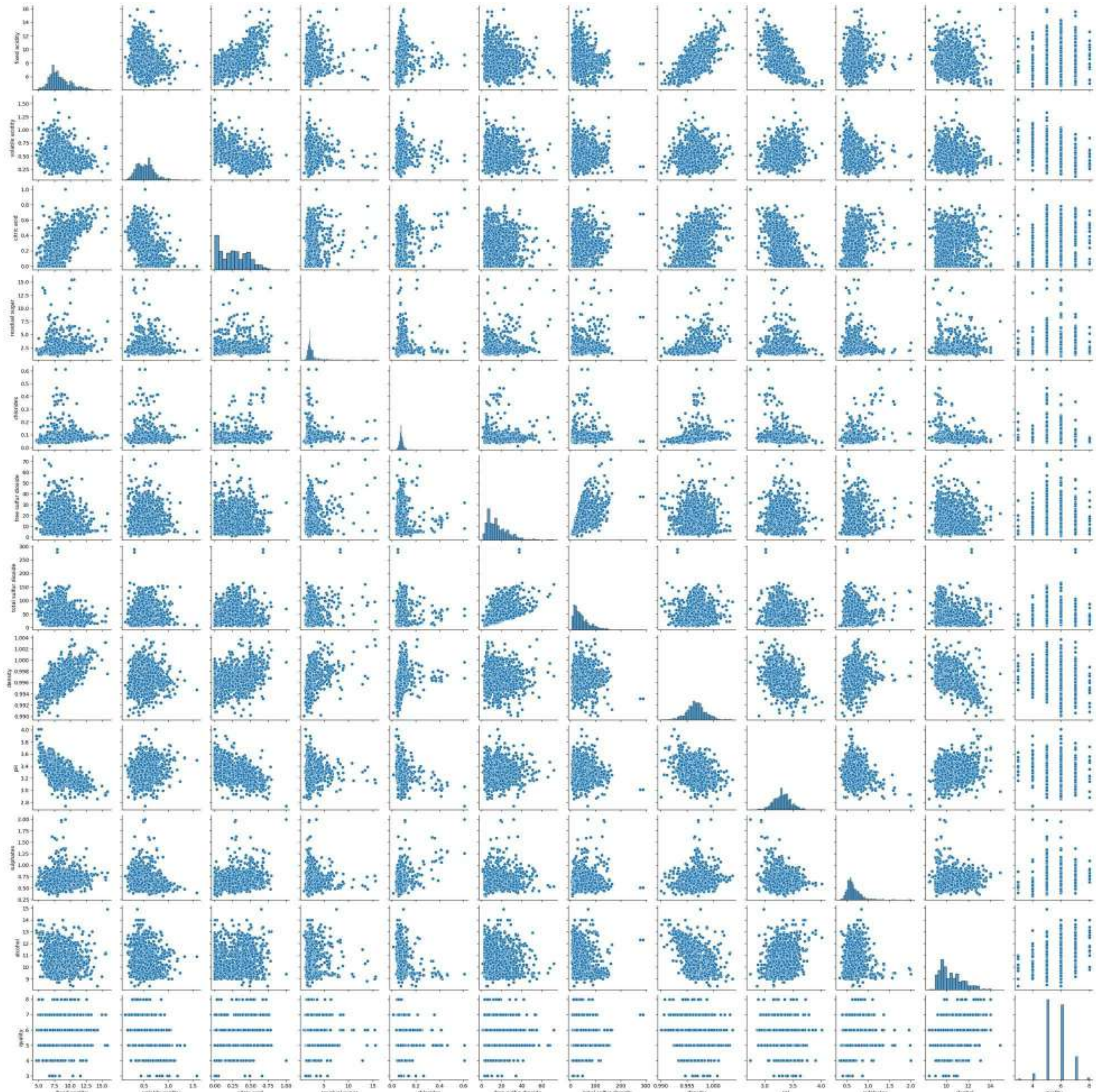

8
7
6
5
4
3
quality
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
sulphates



- ## Multivariate Analysis

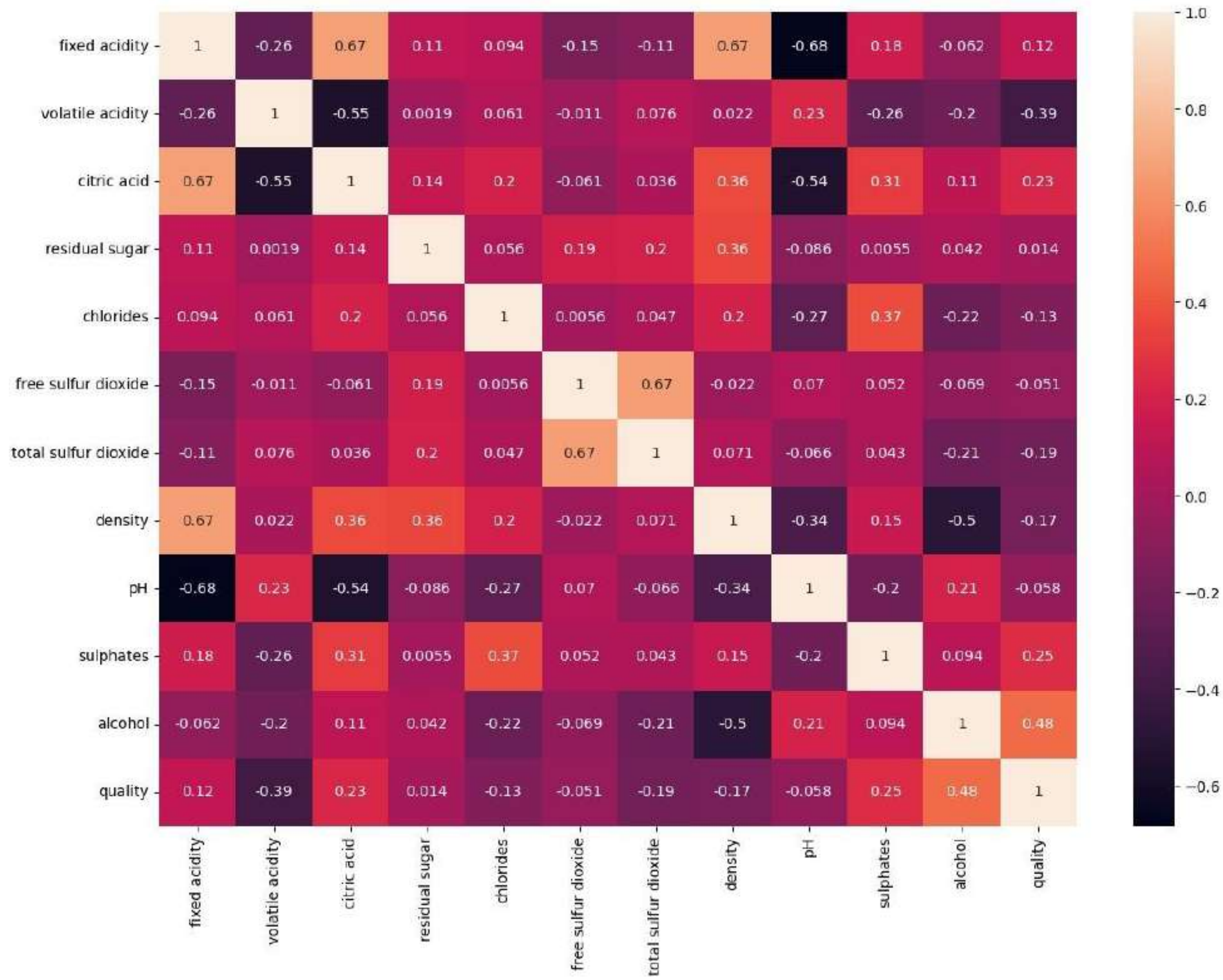
1) pairplot

```
sns.pairplot(df)
plt.show()
```



2) Heatmap

```
fig, ax = plt.subplots(figsize=(14, 10))# increase the size of the
heatmap
sns.heatmap(df.corr(),annot=True)
plt.show()
```




```
df['quality'].value_counts()
```

```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

```
# Define the threshold for categorizing wine quality
threshold = 6.5
# here we assume that the wine with quality >6.5 is good and others
are ordinary
#as its a binary classifiction we take good as 1  and ordinary as 0

df['quality'] = df['quality'].apply(lambda x: 1 if x > threshold else
0)
```

Model Building

```
x=df.iloc[:,:-1]
x.head()
```

```
   fixed acidity volatile acidity citric acid residual sugar
chlorides \
0            7.4             0.70        0.00            1.9
0.076
1            7.8             0.88        0.00            2.6
0.098
2            7.8             0.76        0.04            2.3
0.092
3           11.2             0.28        0.56            1.9
0.075
4            7.4             0.70        0.00            1.9
0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates
\
0                 11.0                  34.0   0.9978  3.51       0.56

1                 25.0                  67.0   0.9968  3.20       0.68

2                 15.0                  54.0   0.9970  3.26       0.65

3                 17.0                  60.0   0.9980  3.16       0.58

4                 11.0                  34.0   0.9978  3.51       0.56


   alcohol
0      9.4
1      9.8
2      9.8
3      9.8
4      9.4
```

```
y=df.iloc[:,-1]
y.head()
```

```
0    0
1    0
2    0
3    0
4    0
Name: quality, dtype: int64
```

Scaling the data

```
from sklearn.preprocessing import MinMaxScaler
scale = MinMaxScaler()

x_scaled=pd.DataFrame(scale.fit_transform(x),columns=x.columns)
x_scaled.head()
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
0       0.247788          0.397260         0.00        0.068493
0.106845
1       0.283186          0.520548         0.00        0.116438
0.143573
2       0.283186          0.438356         0.04        0.095890
0.133556
3       0.584071          0.109589         0.56        0.068493
0.105175
4       0.247788          0.397260         0.00        0.068493
0.106845

   free sulfur dioxide  total sulfur dioxide   density        pH
sulphates \
0             0.140845              0.098940  0.567548  0.606299
0.137725
1             0.338028              0.215548  0.494126  0.362205
0.209581
2             0.197183              0.169611  0.508811  0.409449
0.191617
3             0.225352              0.190813  0.582232  0.330709
0.149701
4             0.140845              0.098940  0.567548  0.606299
0.137725

    alcohol
0  0.153846
1  0.215385
2  0.215385
3  0.215385
4  0.153846
```

train test Split

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test
=train_test_split(x_scaled,y,test_size=0.3,random_state=365)

x_train.shape
```

```
(1119, 11)
```

```
x_test.shape
```

```
(480, 11)
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()

#fitting the data
model.fit(x_train,y_train)
```

```
LogisticRegression()
```

#Evaluate the model

```
y_pred = model.predict(x_test)
y_pred
```

```
array([0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
```

```
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
import sklearn.metrics as metrics

print("Classification Report")
print(metrics.classification_report(y_test, y_pred))
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.90      0.99      0.94       421
           1       0.65      0.19      0.29        59

    accuracy                           0.89       480
   macro avg       0.77      0.59      0.61       480
weighted avg       0.87      0.89      0.86       480
```

```
print("Accuracy score:",metrics.accuracy_score(y_test, y_pred))
print("Precision score:",metrics.precision_score(y_test, y_pred,
average = 'macro'))
print("Recall score:",metrics.recall_score(y_test, y_pred, average =
'macro'))
```

```
Accuracy score: 0.8875
Precision score: 0.7716935586329564
Recall score: 0.5860944482467088
```

Test with random Observations

```
x_scaled.head(50)
```

```
    fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
0        0.247788          0.397260         0.00        0.068493
0.106845
1        0.283186          0.520548         0.00        0.116438
0.143573
2        0.283186          0.438356         0.04        0.095890
0.133556
```

3	0.584071	0.109589	0.56	0.068493	0.105175
4	0.247788	0.397260	0.00	0.068493	0.106845
5	0.247788	0.369863	0.00	0.061644	0.105175
6	0.292035	0.328767	0.06	0.047945	0.095159
7	0.238938	0.363014	0.00	0.020548	0.088481
8	0.283186	0.315068	0.02	0.075342	0.101836
9	0.256637	0.260274	0.36	0.356164	0.098497
10	0.185841	0.315068	0.08	0.061644	0.141903
11	0.256637	0.260274	0.36	0.356164	0.098497
12	0.088496	0.339041	0.00	0.047945	0.128548
13	0.283186	0.335616	0.29	0.047945	0.170284
14	0.380531	0.342466	0.18	0.198630	0.273790
15	0.380531	0.342466	0.19	0.205479	0.263773
16	0.345133	0.109589	0.56	0.061644	0.133556
17	0.309735	0.301370	0.28	0.054795	0.594324
18	0.247788	0.321918	0.08	0.239726	0.123539
19	0.292035	0.136986	0.51	0.061644	0.549249
20	0.380531	0.068493	0.48	0.061644	0.108514
21	0.265487	0.184932	0.31	0.095890	0.116861
22	0.292035	0.212329	0.21	0.047945	0.156928
23	0.345133	0.253425	0.11	0.095890	0.120200
24	0.203540	0.191781	0.14	0.102740	0.121870
25	0.150442	0.184932	0.16	0.034247	0.113523
26	0.265487	0.198630	0.24	0.061644	0.113523
27	0.292035	0.212329	0.21	0.047945	

```
0.156928
28       0.221239           0.404110         0.00         0.068493
0.113523
29       0.283186           0.359589         0.00         0.075342
0.116861
30       0.185841           0.380137         0.07         0.102740
0.128548
31       0.203540           0.386986         0.00         0.109589
0.155259
32       0.327434           0.366438         0.12         0.095890
0.118531
33       0.203540           0.332192         0.12         0.671233
0.101836
34       0.053097           0.136986         0.25         0.061644
0.151920
35       0.283186           0.359589         0.00         0.315068
0.123539
36       0.283186           0.328767         0.14         0.102740
0.123539
37       0.309735           0.178082         0.28         0.082192
0.090150
38       0.097345           0.691781         0.09         0.041096
0.267112
39       0.238938           0.226027         0.36         0.342466
0.103506
40       0.238938           0.226027         0.36         0.342466
0.103506
41       0.371681           0.335616         0.30         0.130137
0.126878
42       0.256637           0.253425         0.20         0.116438
0.534224
43       0.309735           0.369863         0.22         0.089041
0.095159
44       0.194690           0.376712         0.02         0.061644
0.063439
45       0.000000           0.273973         0.15         0.082192
0.070117
46       0.274336           0.558219         0.43         0.089041
0.170284
47       0.362832           0.116438         0.52         0.047945
0.168614
48       0.159292           0.191781         0.23         0.047945
0.090150
49       0.088496           0.130137         0.37         0.034247
0.103506

    free sulfur dioxide  total sulfur dioxide   density        pH
sulphates  \
0               0.140845              0.098940  0.567548  0.606299
```

```
0.137725
1           0.338028               0.215548  0.494126  0.362205
0.209581
2           0.197183               0.169611  0.508811  0.409449
0.191617
3           0.225352               0.190813  0.582232  0.330709
0.149701
4           0.140845               0.098940  0.567548  0.606299
0.137725
5           0.169014               0.120141  0.567548  0.606299
0.137725
6           0.197183               0.187279  0.464758  0.440945
0.077844
7           0.197183               0.053004  0.332599  0.511811
0.083832
8           0.112676               0.042403  0.494126  0.488189
0.143713
9           0.225352               0.339223  0.567548  0.480315
0.281437
10          0.197183               0.208481  0.428047  0.425197
0.125749
11          0.225352               0.339223  0.567548  0.480315
0.281437
12          0.211268               0.187279  0.310573  0.661417
0.113772
13          0.112676               0.081272  0.538179  0.409449
0.736527
14          0.718310               0.491166  0.626285  0.330709
0.329341
15          0.704225               0.501767  0.626285  0.338583
0.359281
16          0.478873               0.342756  0.501468  0.440945
0.251497
17          0.211268               0.176678  0.494126  0.291339
0.568862
18          0.070423               0.081272  0.538179  0.503937
0.101796
19          0.225352               0.176678  0.501468  0.236220
0.449102
20          0.394366               0.190813  0.494126  0.511811
0.119760
21          0.309859               0.229682  0.596916  0.614173
0.191617
22          0.126761               0.109541  0.479442  0.338583
0.347305
23          0.112676               0.215548  0.494126  0.338583
0.119760
24          0.281690               0.120141  0.494126  0.543307
0.179641
```

25	0.140845	0.060071	0.398678	0.472441	0.137725
26	0.042254	0.017668	0.450073	0.425197	0.155689
27	0.126761	0.109541	0.479442	0.338583	0.347305
28	0.183099	0.102473	0.523495	0.574803	0.131737
29	0.098592	0.035336	0.464758	0.503937	0.155689
30	0.225352	0.268551	0.420705	0.480315	0.125749
31	0.295775	0.109541	0.479442	0.566929	0.143713
32	0.197183	0.378092	0.479442	0.338583	0.197605
33	0.549296	0.272085	0.677680	0.559055	0.113772
34	0.169014	0.155477	0.413363	0.503937	0.131737
35	0.056338	0.042403	0.626285	0.519685	0.131737
36	0.028169	0.031802	0.545521	0.535433	0.161677
37	0.169014	0.084806	0.494126	0.385827	0.239521
38	0.084507	0.045936	0.288546	0.598425	0.089820
39	0.154930	0.286219	0.567548	0.464567	0.299401
40	0.154930	0.286219	0.567548	0.464567	0.299401
41	0.225352	0.141343	0.552863	0.409449	0.107784
42	0.098592	0.028269	0.494126	0.370079	0.341317
43	0.112676	0.060071	0.494126	0.440945	0.520958
44	0.056338	0.017668	0.450073	0.582677	0.113772
45	0.098592	0.208481	0.244493	0.913386	0.137725
46	0.295775	0.381625	0.508811	0.401575	0.239521
47	0.154930	0.109541	0.501468	0.401575	0.149701
48	0.056338	0.021201	0.420705	0.472441	0.137725
49	0.154930	0.318021	0.391336	0.456693	

```
0.149701
```

	alcohol
0	0.153846
1	0.215385
2	0.215385
3	0.215385
4	0.153846
5	0.153846
6	0.153846
7	0.246154
8	0.169231
9	0.323077
10	0.123077
11	0.323077
12	0.230769
13	0.107692
14	0.123077
15	0.123077
16	0.323077
17	0.138462
18	0.092308
19	0.123077
20	0.153846
21	0.200000
22	0.169231
23	0.153846
24	0.200000
25	0.138462
26	0.169231
27	0.169231
28	0.153846
29	0.215385
30	0.261538
31	0.338462
32	0.215385
33	0.153846
34	0.123077
35	0.184615
36	0.369231
37	0.200000
38	0.215385
39	0.323077
40	0.323077
41	0.138462
42	0.323077
43	0.292308
44	0.169231
45	0.723077
46	0.123077

```
47  0.169231
48  0.123077
49  0.123077
```

```
model.predict([[0.283186,0.315068,0.02,0.075342,0.101836,0.112676,0.042403,0.494126,0.488189,0.143713,0.169231]])
```

```
array([0])
```

```
model.predict([[0.292035,0.054795     ,0.35,     0.054795,  0.070117
     ,0.084507  ,0.031802, 0.331131,
                0.456693   ,0.281437, 0.538462]])
```

```
array([1])
```