assignment-4-kavin-21bps1630

September 21, 2023

[55]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
 ↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
 ↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
 ↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
 ↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
 ↪outside of the current session
```

```
/kaggle/input/red-wine-quaility-dataset/winequality-red.csv
```

[56]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[57]:
```
data=pd.read_csv('/kaggle/input/red-wine-quaility-dataset/winequality-red.csv')
```

[58]:
```
data.head(10)
```

[58]:

	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
5            7.4              0.66         0.00             1.8      0.075
6            7.9              0.60         0.06             1.6      0.069
7            7.3              0.65         0.00             1.2      0.065
8            7.8              0.58         0.02             2.0      0.073
9            7.5              0.50         0.36             6.1      0.071

   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
5                 13.0                  40.0   0.9978  3.51       0.56
6                 15.0                  59.0   0.9964  3.30       0.46
7                 15.0                  21.0   0.9946  3.39       0.47
8                  9.0                  18.0   0.9968  3.36       0.57
9                 17.0                 102.0   0.9978  3.35       0.80

   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
5      9.4        5
6      9.4        5
7     10.0        7
8      9.5        7
9     10.5        5
```

[59]:
```
data.shape
```

[59]:
```
(1599, 12)
```

[60]:
```
data.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
```

[61]:
```
data.describe()
```

[61]:

	fixed acidity	volatile acidity	citric acid	residual sugar	\
count	1599.000000	1599.000000	1599.000000	1599.000000	
mean	8.319637	0.527821	0.270976	2.538806	
std	1.741096	0.179060	0.194801	1.409928	
min	4.600000	0.120000	0.000000	0.900000	
25%	7.100000	0.390000	0.090000	1.900000	
50%	7.900000	0.520000	0.260000	2.200000	
75%	9.200000	0.640000	0.420000	2.600000	
max	15.900000	1.580000	1.000000	15.500000	

	chlorides	free sulfur dioxide	total sulfur dioxide	density	\
count	1599.000000	1599.000000	1599.000000	1599.000000	
mean	0.087467	15.874922	46.467792	0.996747	
std	0.047065	10.460157	32.895324	0.001887	
min	0.012000	1.000000	6.000000	0.990070	
25%	0.070000	7.000000	22.000000	0.995600	
50%	0.079000	14.000000	38.000000	0.996750	
75%	0.090000	21.000000	62.000000	0.997835	
max	0.611000	72.000000	289.000000	1.003690	

	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000
mean	3.311113	0.658149	10.422983	5.636023
std	0.154386	0.169507	1.065668	0.807569
min	2.740000	0.330000	8.400000	3.000000
25%	3.210000	0.550000	9.500000	5.000000
50%	3.310000	0.620000	10.200000	6.000000
75%	3.400000	0.730000	11.100000	6.000000
max	4.010000	2.000000	14.900000	8.000000

[62]:
```
data.isnull().sum()
```

[62]:
```
fixed acidity           0
volatile acidity        0
citric acid             0
```

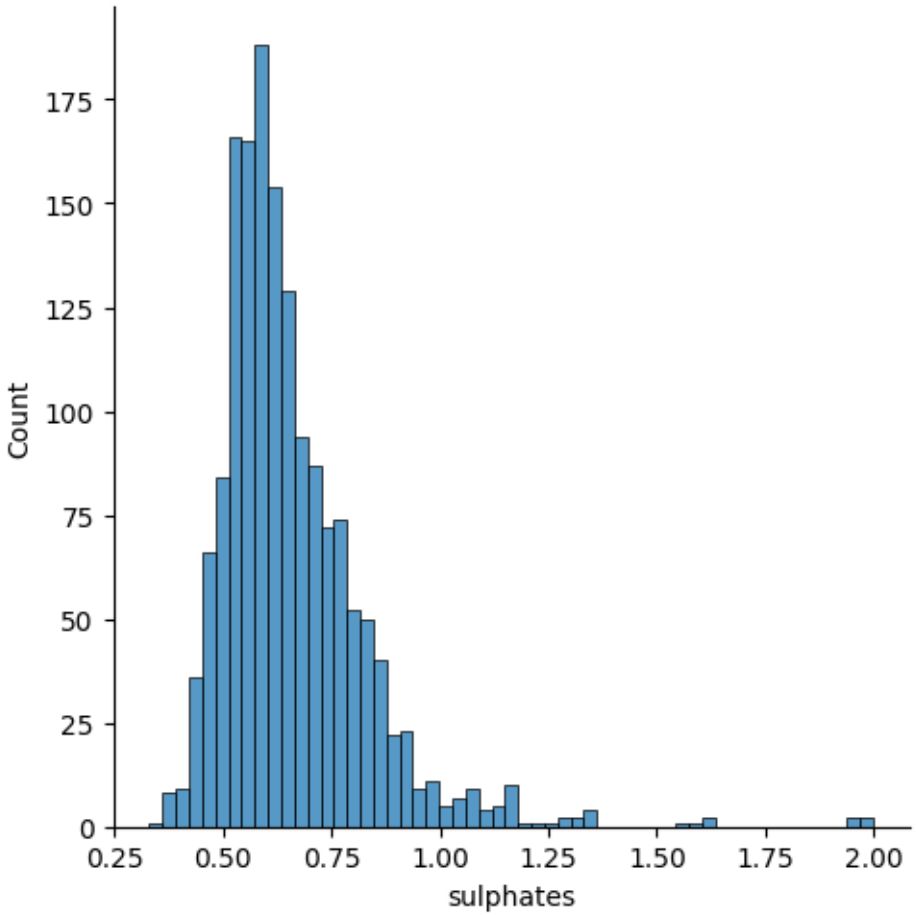
```
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

[63]:
```
data.corr().quality.sort_values(ascending=False)
```

[63]:
```
quality                 1.000000
alcohol                 0.476166
sulphates               0.251397
citric acid             0.226373
fixed acidity           0.124052
residual sugar          0.013732
free sulfur dioxide    -0.050656
pH                     -0.057731
chlorides              -0.128907
density                -0.174919
total sulfur dioxide   -0.185100
volatile acidity       -0.390558
Name: quality, dtype: float64
```

[64]:
```
sns.displot(data['sulphates'])
plt.show()
```

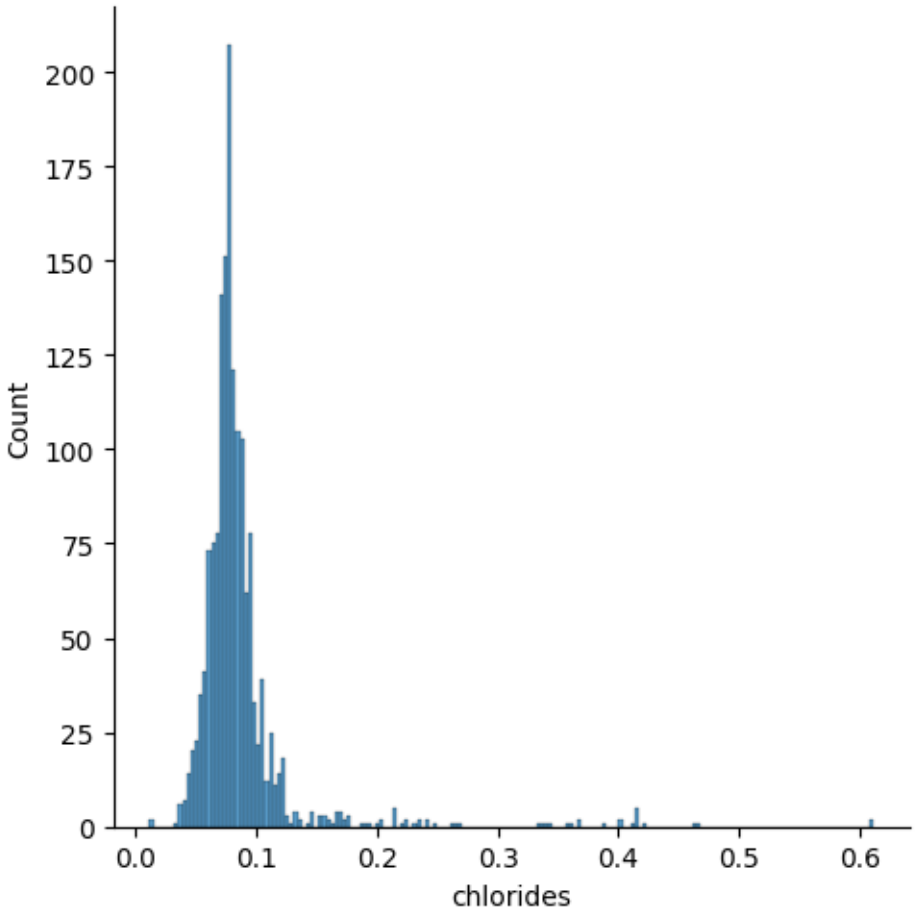
```
/opt/conda/lib/python3.10/site-packages/seaborn/axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



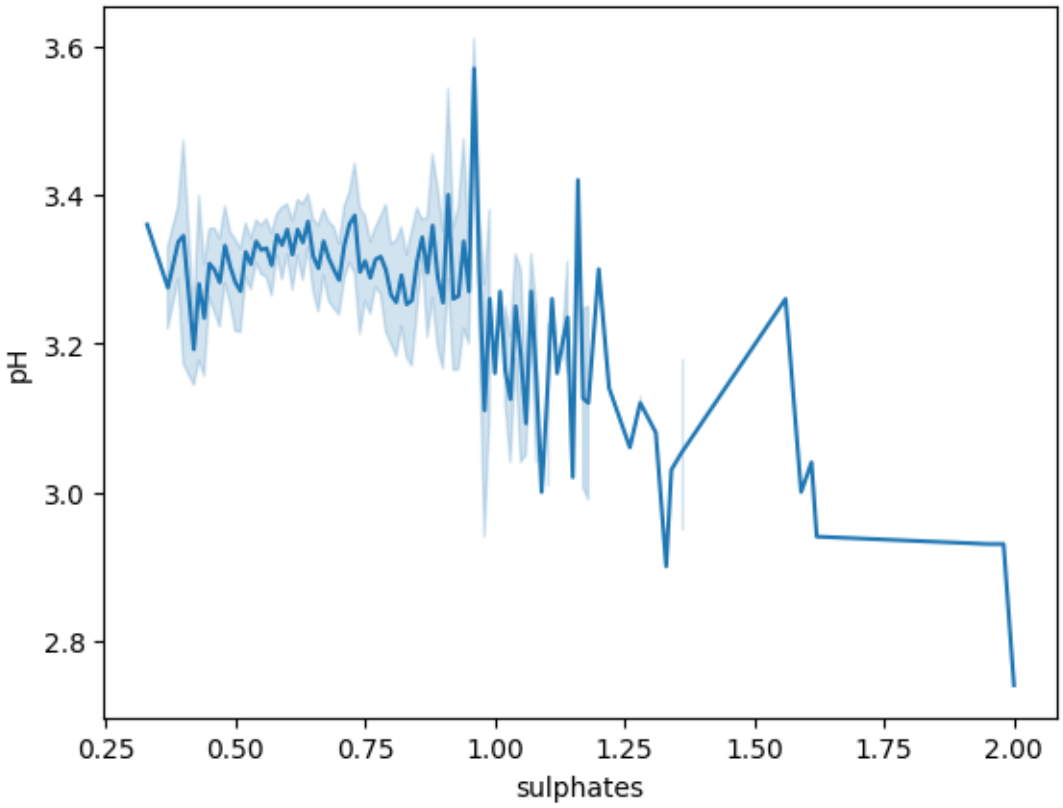
[65]:
```
sns.displot(data['chlorides'])
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

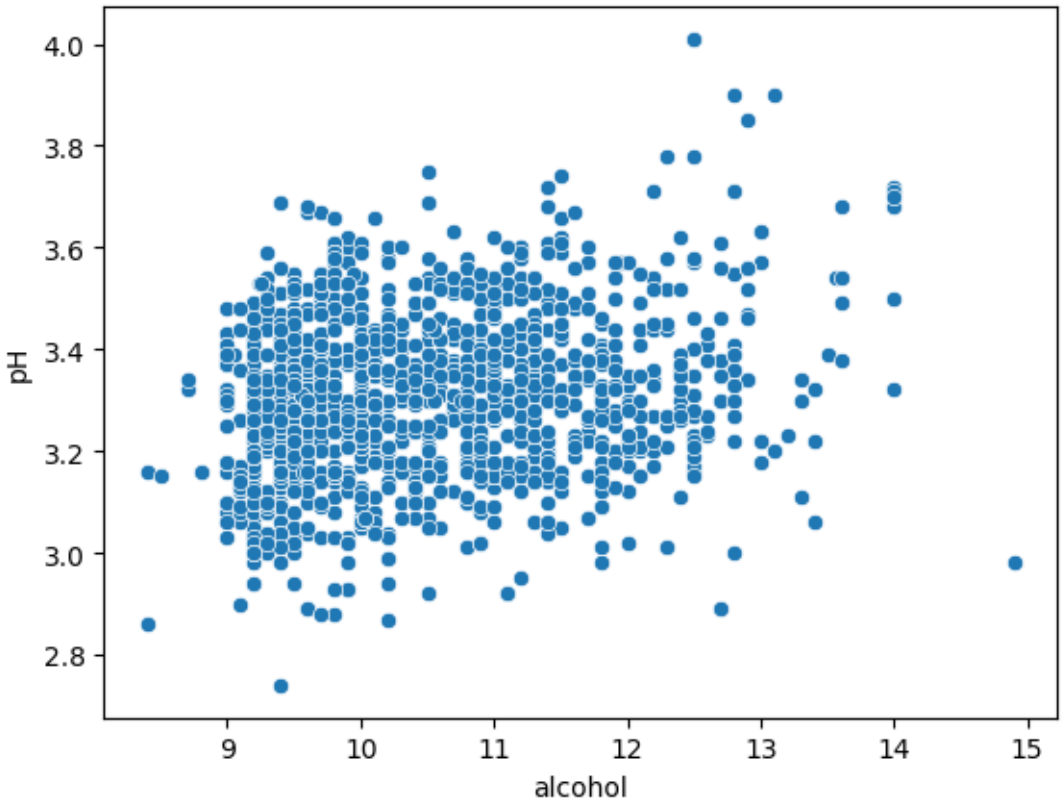
[65]:
```
<seaborn.axisgrid.FacetGrid at 0x7f1fbd410160>
```



[66]:
```
sns.lineplot(x=data['sulphates'],y=data['pH'])
plt.show()
```

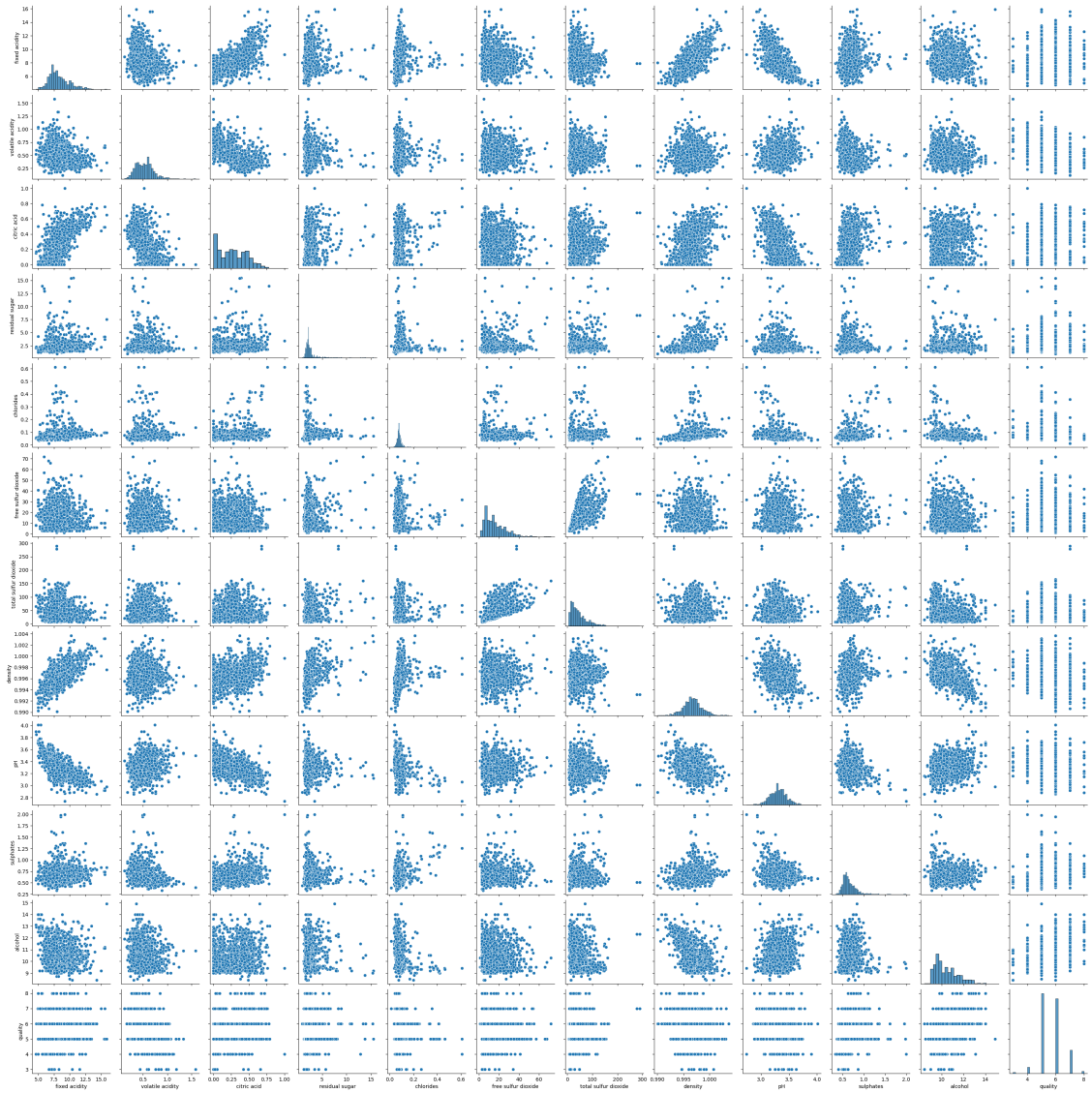


[67]:
```
sns.scatterplot(x=data['alcohol'],y=data['pH'])
plt.show()
```



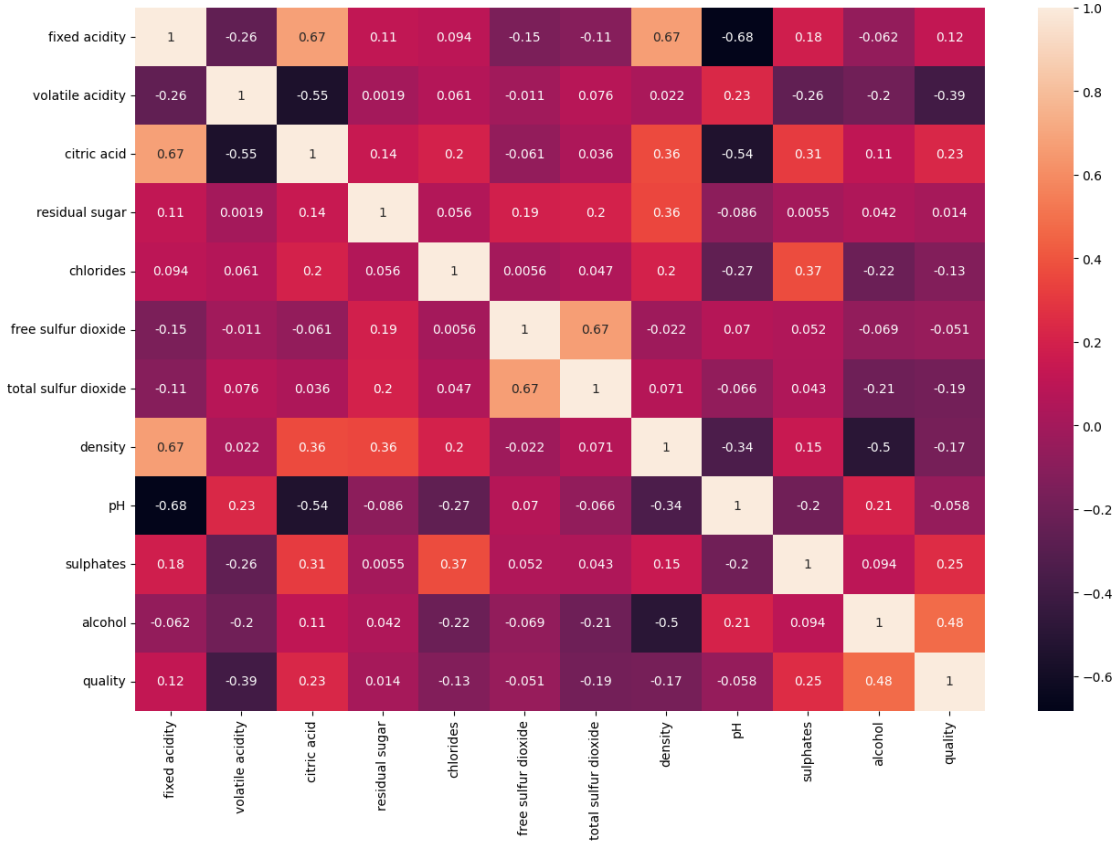
[68]:
```
sns.pairplot(data)
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



[69]:
```
plt=plt.figure(figsize=(15,10))
sns.heatmap(data.corr(),annot=True)
```

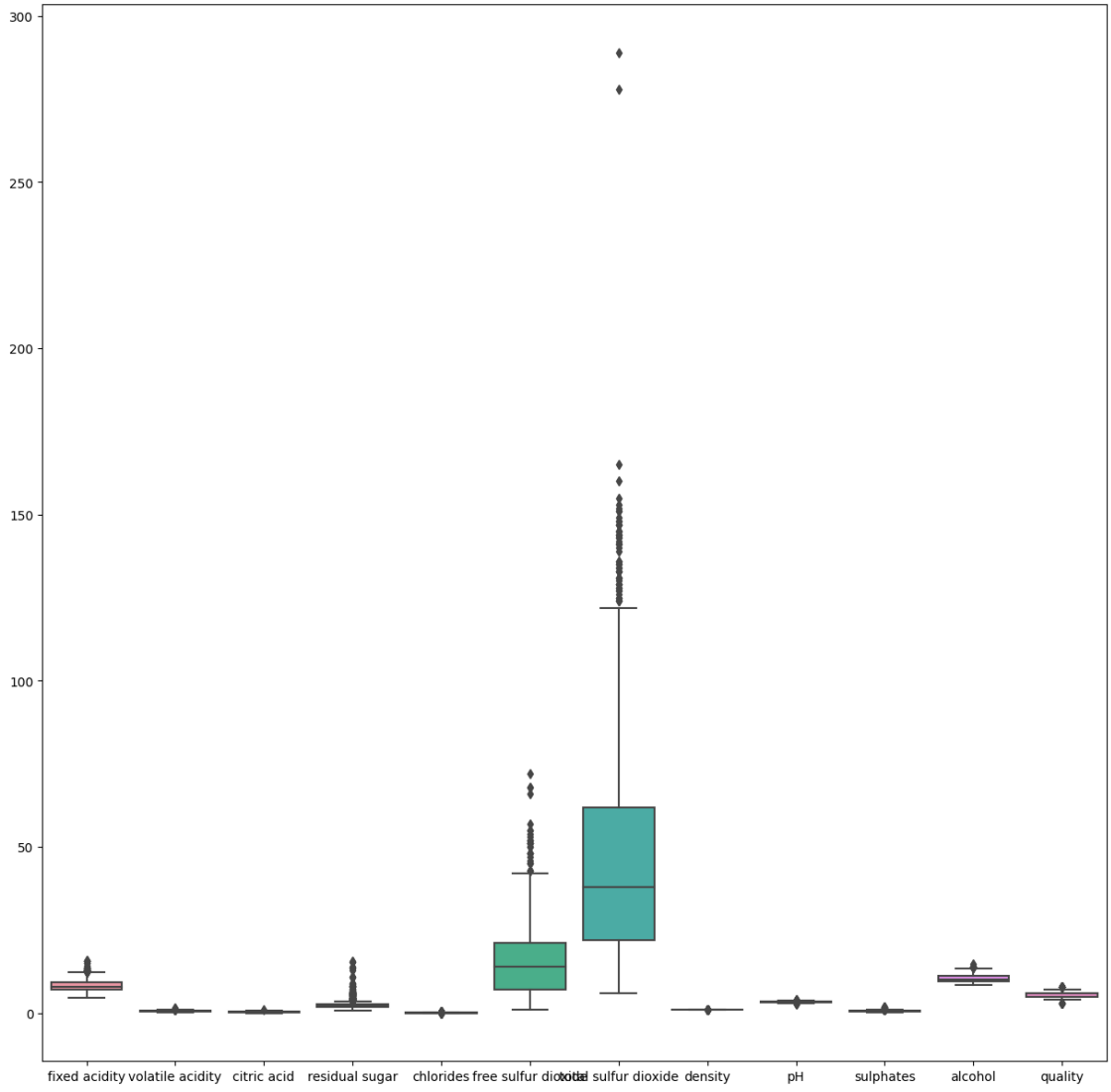
[69]: <Axes: >



[70]:
```
X=data.copy()
```

[75]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(15,15))
sns.boxplot(X)
```

[75]: <Axes: >

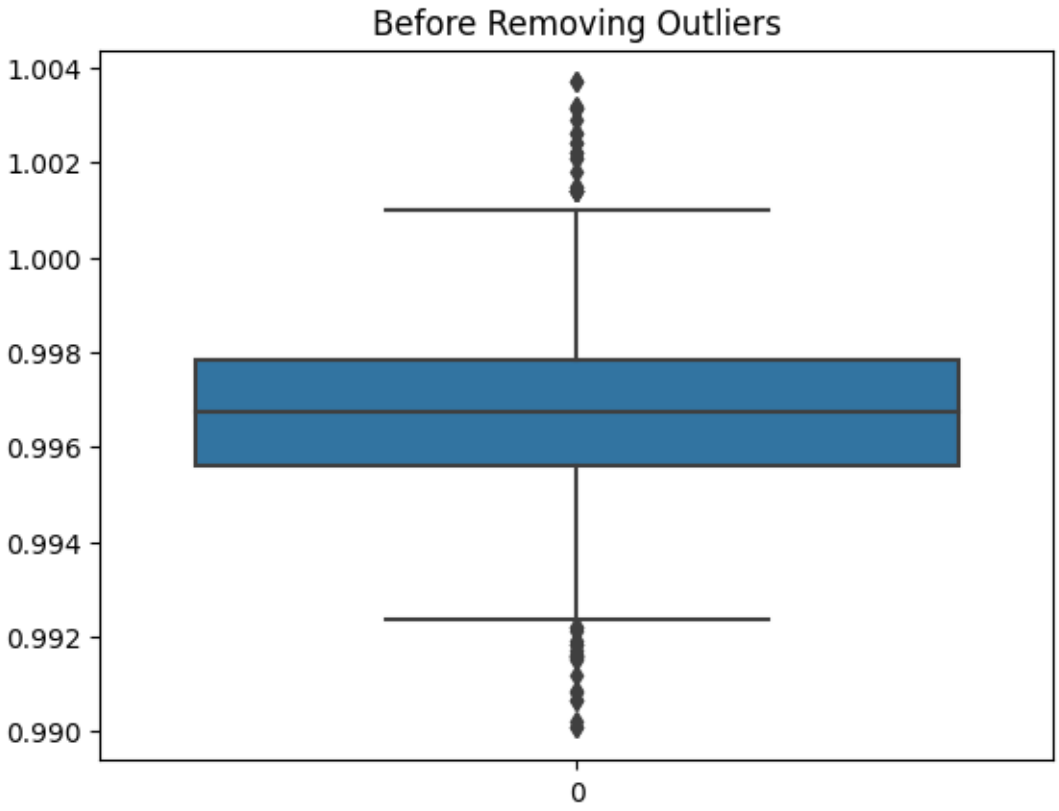


[82]:
```python
def remove_outliers(column):
    q1 = column.quantile(0.25)  # Q1
    q3 = column.quantile(0.75)  # Q3
    IQR = q3 - q1
    upper_limit = q3 + 1.5 * IQR
    lower_limit = q1 - 1.5 * IQR
    return column[(column >= lower_limit) & (column <= upper_limit)]

# Apply the remove_outliers function to all columns (except "quality")
X_filtered = X.copy()  # Create a copy of the original DataFrame
for column in X.columns:
    if column != "quality":
        X_filtered[column] = remove_outliers(X[column])
```

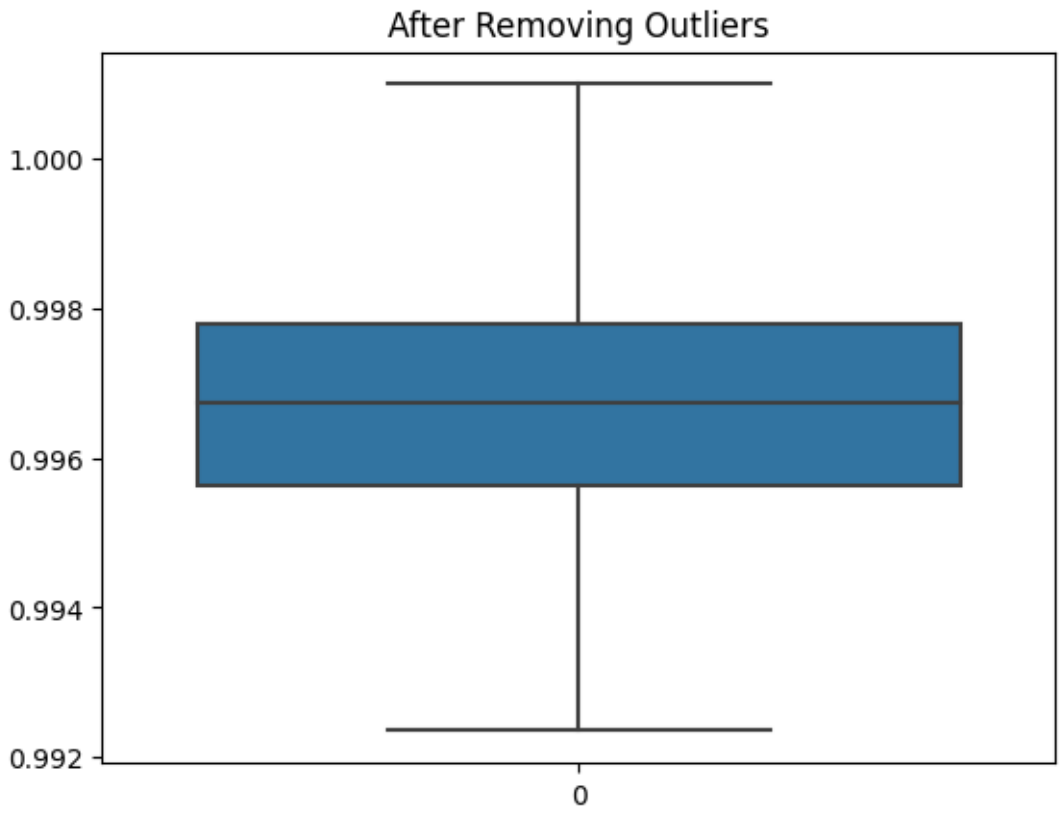
[89]:
```
sns.boxplot(X['density']).set(title="Before Removing Outliers")
```

[89]:
```
[Text(0.5, 1.0, 'Before Removing Outliers')]
```



[91]:
```
sns.boxplot(X_filtered['density']).set(title="After Removing Outliers")
```

[91]:
```
[Text(0.5, 1.0, 'After Removing Outliers')]
```



[104]:
```
x=X.iloc[:,:-1].values
y=X.iloc[:,-1].values
```

[99]:
```
X_filtered['quality'].unique()
```

[99]:
```
array([5, 6, 7, 4, 8, 3])
```

[105]:
```
X.isnull().sum()
```

[105]:
```
fixed acidity           0
volatile acidity        0
citric acid             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
```

[111]:
```
y = X['quality'].apply(lambda y_val: 1 if y_val>=7 else 0)
```

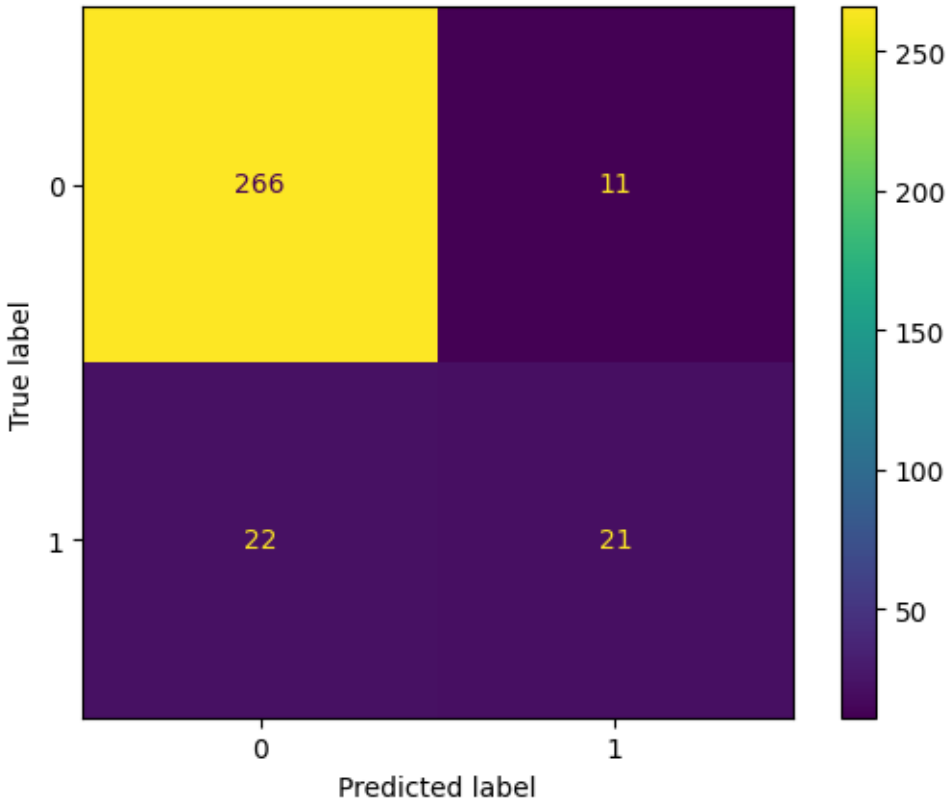
[112]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=2023,test_size=0.
↪2,stratify=y)
```

[113]:
```
from sklearn.metrics import accuracy_score,ConfusionMatrixDisplay
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(n_estimators=300)
rfc.fit(x_train,y_train)
ypred=rfc.predict(x_test)
print(accuracy_score(y_test,ypred))
```

```
0.896875
```

[115]:
```
ConfusionMatrixDisplay.from_estimator(
rfc,
x_test,
y_test
)
```

[115]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f1fb44e4a30>
```



[116]:
```
input_data_3 = [7.9, 1.0, 0, 3.0, 0.08, 30, 100, 0.9562, 3.1, 0.74, 11.5]
prediction = rfc.predict([input_data_3])
if prediction==1:
    print("Alcohol Quality is Good")
else:
    print("Quality is Bad")
```

```
Quality is Bad
```