mlli7oyzb

June 4, 2023

1 Prediction of red wine quality

How could we not select this dataset and still call ourselves french people after that. It includes 11 input variables on different wines, determined by physical and chemical tests, and one output data based on sensory data, being the wine quality.

Dataset's reference: P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

Our objective is to rate a wine and label it as good or bad using its combination of physical and chemical properties. To achieve this, we will analyze our dataset (distribution of wine quality, correlation between properties, presence of outliers in each property). Based on our findings, we will then optimize the dataset to get a more accurate wine classification. Finally, we will test various algorithms to find which one gives the best results.

1.1 1 - Dataset anaysis

We start by loading the libraries we will use to predict the quality of a red wine.

[319]:
```python
#!/usr/bin/python3

from sys import argv, exit

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression, Ridge, Lasso
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import MinMaxScaler
import seaborn as sns
from sklearn.svm import LinearSVC
```

We load the dataset from the 'dataset.csv' file and then display the information from this dataset.

[320]:
```
dataframe = pd.read_csv("dataset.csv")
dataframe.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
```

We can see that there are 1599 rows, 12 columns, no null values and that there are different data types: Int, Float.

We then display an example of the data within this dataset.

[321]:
```
dataframe.head(10)
```

[321]:
```
   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
```

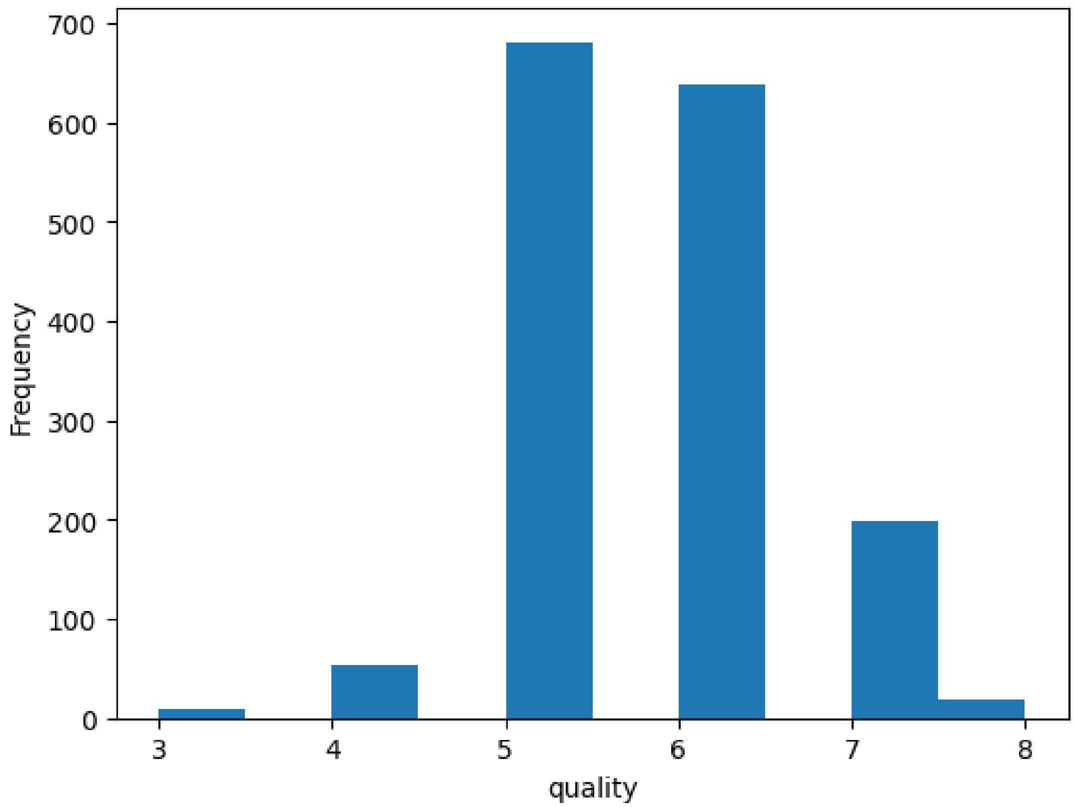
```
5                 13.0                    40.0   0.9978  3.51       0.56
6                 15.0                    59.0   0.9964  3.30       0.46
7                 15.0                    21.0   0.9946  3.39       0.47
8                  9.0                    18.0   0.9968  3.36       0.57
9   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

1.2 2 - Dataset optimization

We display the distribution of the wine quality.

[322]:
```
dataframe['quality'].plot(kind='hist').set(xlabel="quality")
```

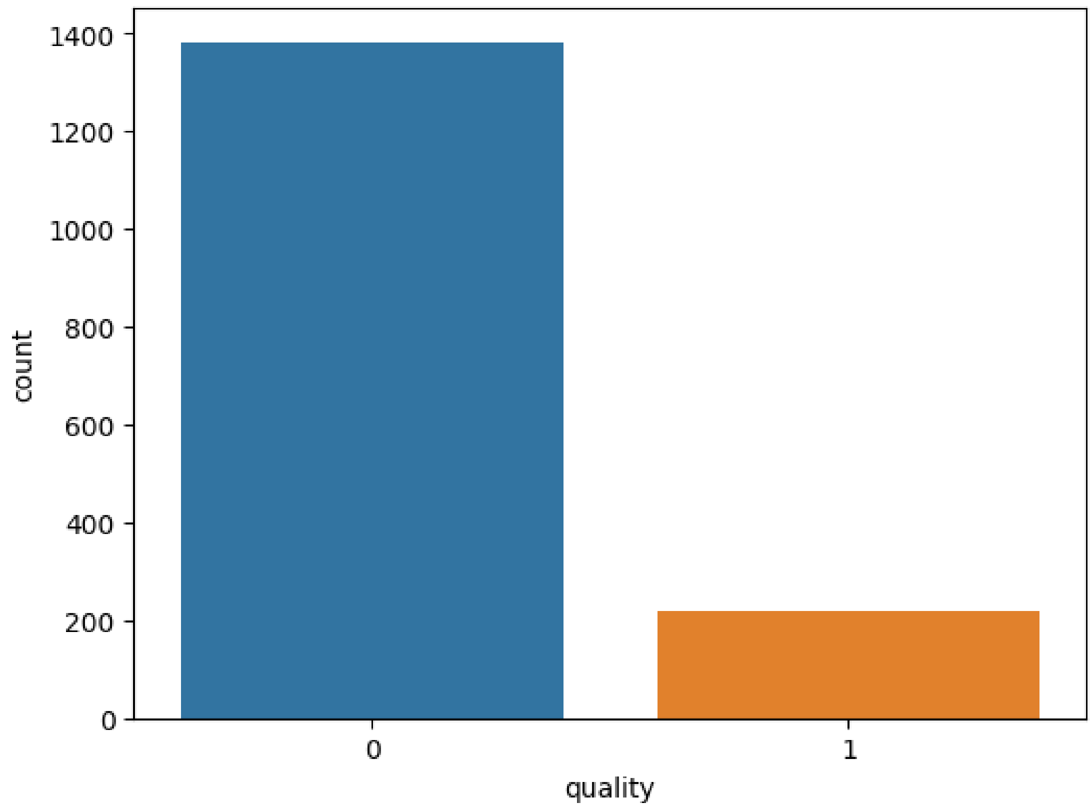
[322]:
```
[Text(0.5, 0, 'quality')]
```



We chose to go with the quality as our criteria for deciding if a wine is good or bad, and most of the wines in the dataset hang around either 6 and 5 on the quality axis, so they will represent our average, as well as our baseline, such as below 5 will be considered the bad ones and above 6 the good ones.

[323]:
```
dataframe['quality'] = dataframe['quality'].apply(lambda x: 0 if x < 7 else 1 )
sns.countplot(x='quality', data=dataframe)
```

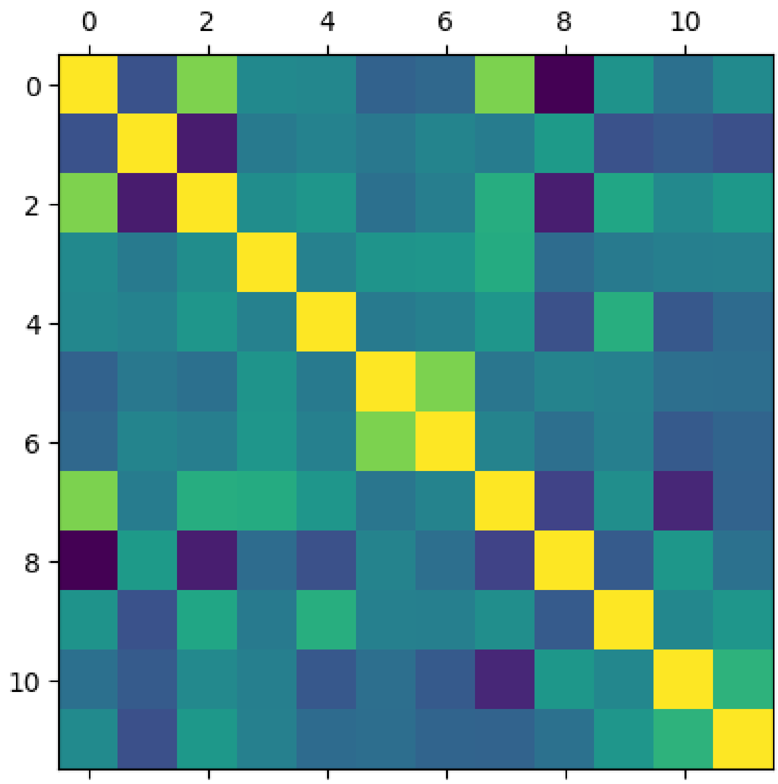
[323]:
```
<Axes: xlabel='quality', ylabel='count'>
```



We have decided to separate the wines into two categories: good and bad. We have about 200 good wines and about 1400 bad ones.

A correlation diagram is displayed to see how columns are related to each other.

[324]:
```
plt.matshow(dataframe.corr())
plt.show()
```



[325]:
```
dataframe.corr()
```

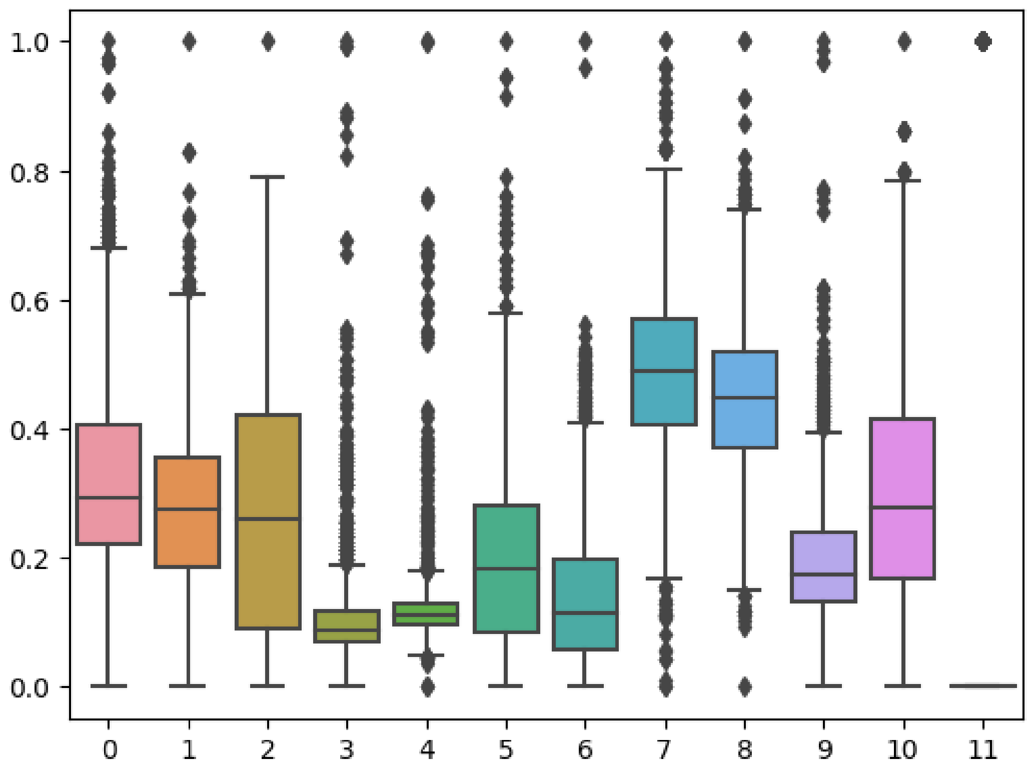
[325]:
```
                      fixed acidity  volatile acidity  citric acid  \
fixed acidity              1.000000         -0.256131     0.671703
volatile acidity          -0.256131          1.000000    -0.552496
citric acid                0.671703         -0.552496     1.000000
residual sugar             0.114777          0.001918     0.143577
chlorides                  0.093705          0.061298     0.203823
free sulfur dioxide       -0.153794         -0.010504    -0.060978
total sulfur dioxide      -0.113181          0.076470     0.035533
density                    0.668047          0.022026     0.364947
pH                        -0.682978          0.234937    -0.541904
sulphates                  0.183006         -0.260987     0.312770
alcohol                   -0.061668         -0.202288     0.109903
quality                    0.120061         -0.270712     0.214716

                      residual sugar  chlorides  free sulfur dioxide  \
fixed acidity               0.114777   0.093705            -0.153794
volatile acidity            0.001918   0.061298            -0.010504
citric acid                 0.143577   0.203823            -0.060978
residual sugar              1.000000   0.055610             0.187049
chlorides                   0.055610   1.000000             0.005562
```

```
free sulfur dioxide          0.187049   0.005562             1.000000
total sulfur dioxide         0.203028   0.047400             0.667666
density                      0.355283   0.200632            -0.021946
pH                          -0.085652  -0.265026             0.070377
sulphates                    0.005527   0.371260             0.051658
alcohol                      0.042075  -0.221141            -0.069408
quality                      0.047779  -0.097308            -0.071747

                      total sulfur dioxide   density        pH  sulphates  \
fixed acidity                    -0.113181  0.668047 -0.682978   0.183006
volatile acidity                  0.076470  0.022026  0.234937  -0.260987
citric acid                       0.035533  0.364947 -0.541904   0.312770
residual sugar                    0.203028  0.355283 -0.085652   0.005527
chlorides                         0.047400  0.200632 -0.265026   0.371260
free sulfur dioxide               0.667666 -0.021946  0.070377   0.051658
total sulfur dioxide              1.000000  0.071269 -0.066495   0.042947
density                           0.071269  1.000000 -0.341699   0.148506
pH                               -0.066495 -0.341699  1.000000  -0.196648
sulphates                         0.042947  0.148506 -0.196648   1.000000
alcohol                          -0.205654 -0.496180  0.205633   0.093595
quality                          -0.139517 -0.150460 -0.057283   0.199485

                       alcohol   quality
fixed acidity        -0.061668  0.120061
volatile acidity     -0.202288 -0.270712
citric acid           0.109903  0.214716
residual sugar        0.042075  0.047779
chlorides            -0.221141 -0.097308
free sulfur dioxide  -0.069408 -0.071747
total sulfur dioxide -0.205654 -0.139517
density              -0.496180 -0.150460
pH                    0.205633 -0.057283
sulphates             0.093595  0.199485
alcohol               1.000000  0.407315
quality               0.407315  1.000000
```

In this section we will try to find outliers and remove them.

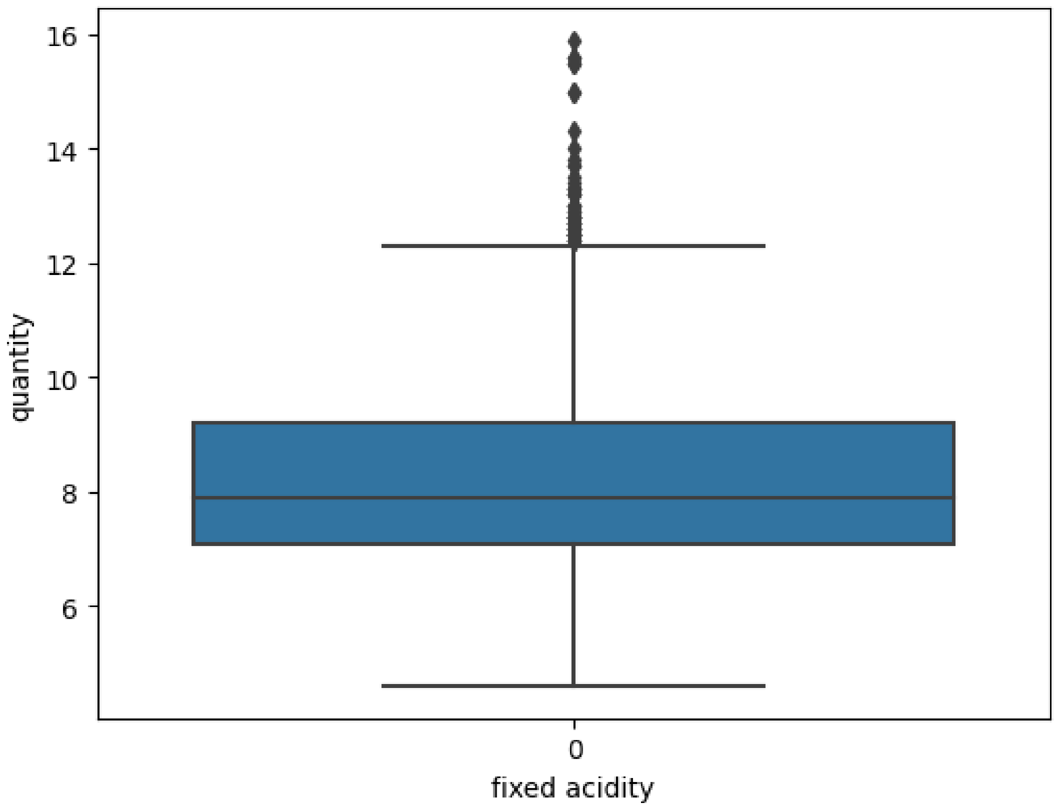
[326]:
```
dataframe_scaled = pd.DataFrame(MinMaxScaler().fit_transform(dataframe))
sns.boxplot(dataframe_scaled)
```

[326]:
```
<Axes: >
```



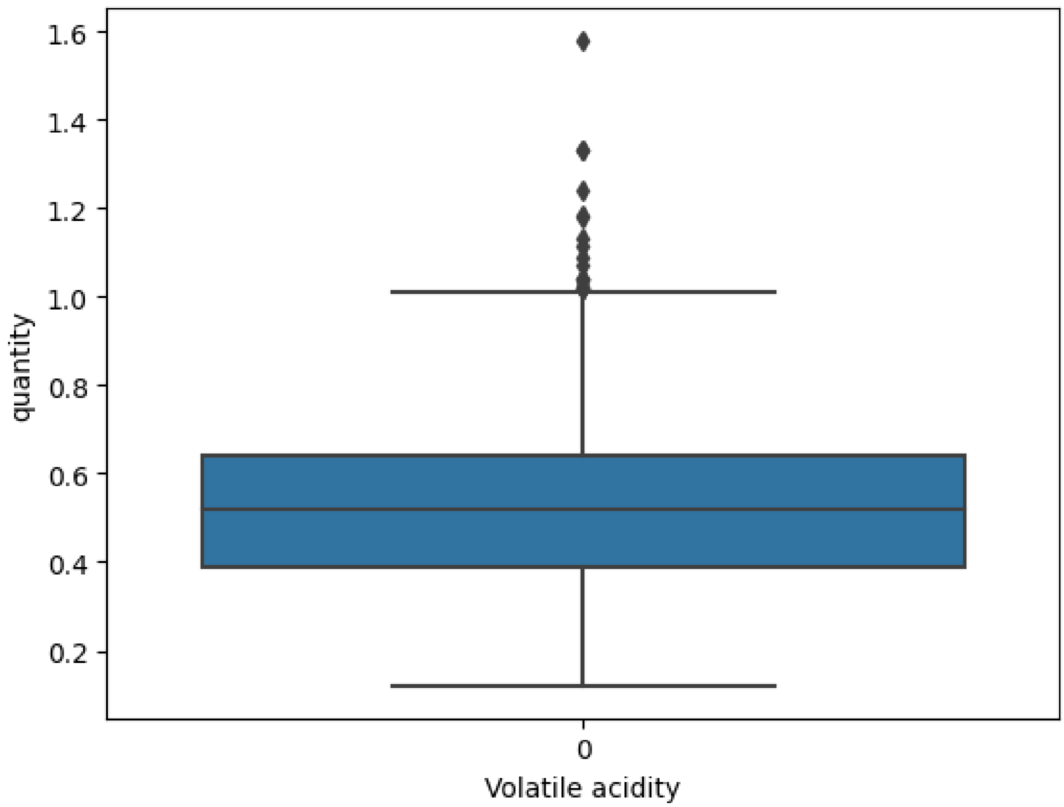
[327]:
```
sns.boxplot(dataframe['fixed acidity']).set(ylabel = "quantity",xlabel = "fixed␣
↪acidity")
```

[327]: [Text(0, 0.5, 'quantity'), Text(0.5, 0, 'fixed acidity')]



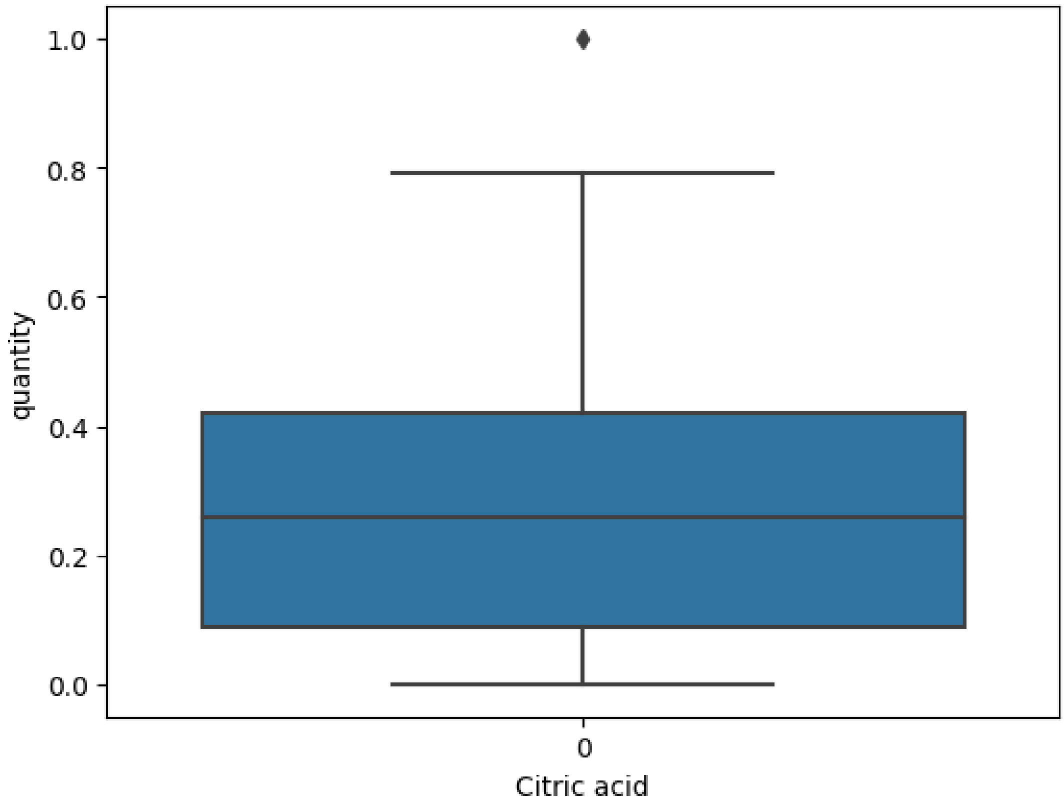
[328]:
```
sns.boxplot(dataframe['volatile acidity']).set(ylabel = "quantity",xlabel =␣
↪"Volatile acidity")
```

[328]: [Text(0, 0.5, 'quantity'), Text(0.5, 0, 'Volatile acidity')]



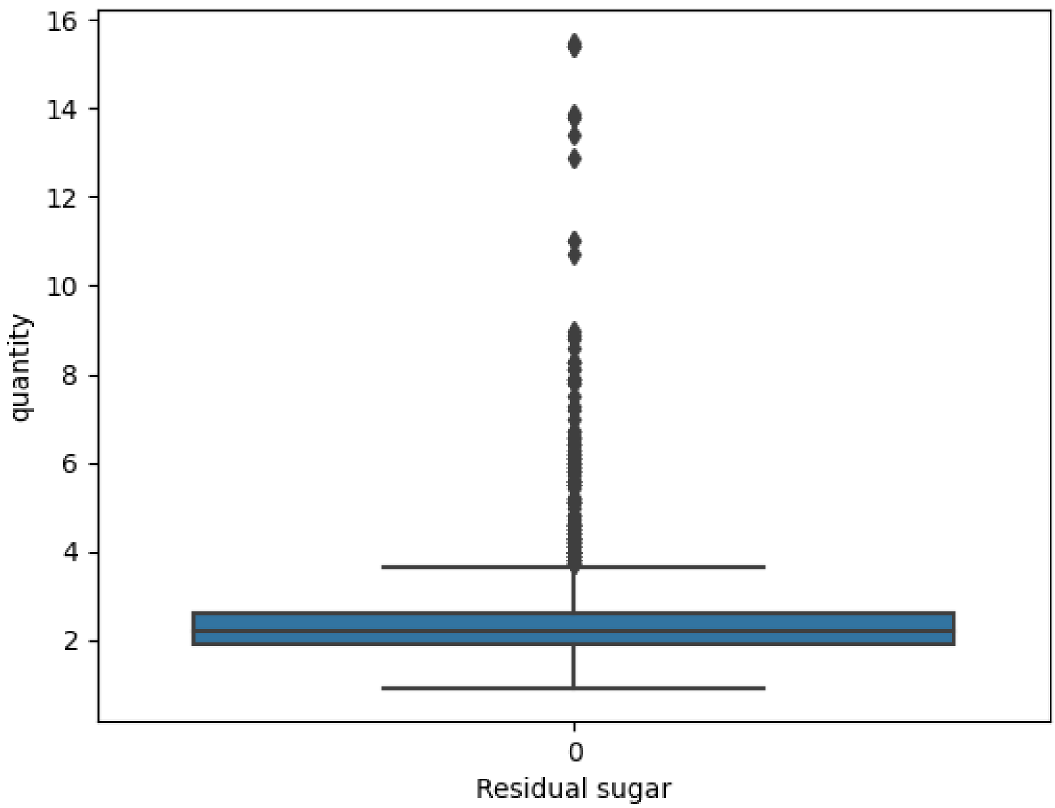
[329]:
```
sns.boxplot(dataframe['citric acid']).set(ylabel = "quantity",xlabel = "Citric␣
↪acid")
```

[329]: [Text(0, 0.5, 'quantity'), Text(0.5, 0, 'Citric acid')]



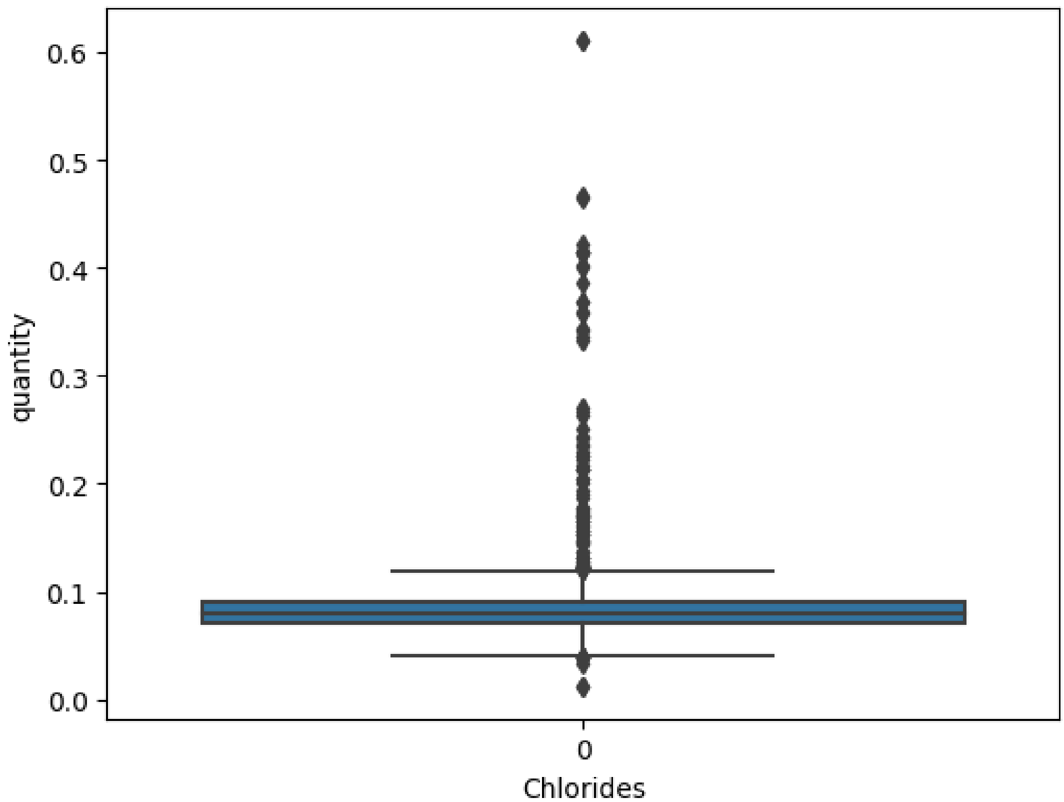
[330]:
```
sns.boxplot(dataframe['residual sugar']).set(ylabel = "quantity",xlabel =␣
↪"Residual sugar")
```

[330]:
```
[Text(0, 0.5, 'quantity'), Text(0.5, 0, 'Residual sugar')]
```



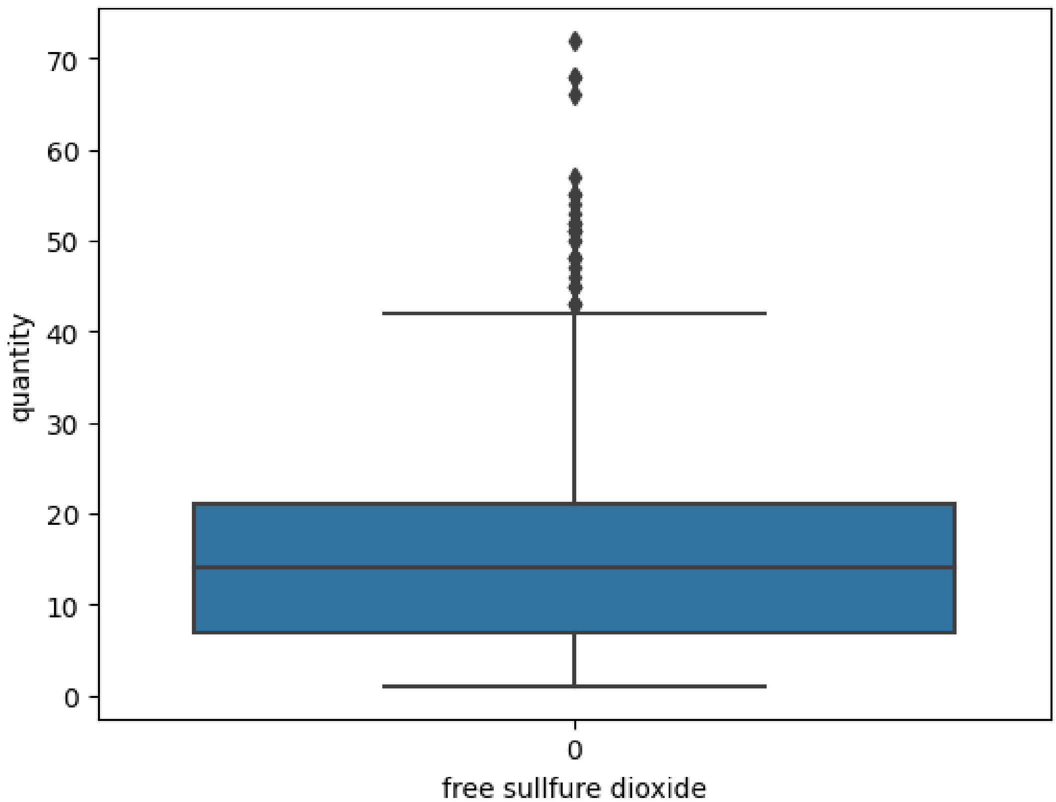
[331]:
```
sns.boxplot(dataframe['chlorides']).set(ylabel = "quantity",xlabel =␣
↪"Chlorides")
```

[331]: [Text(0, 0.5, 'quantity'), Text(0.5, 0, 'Chlorides')]



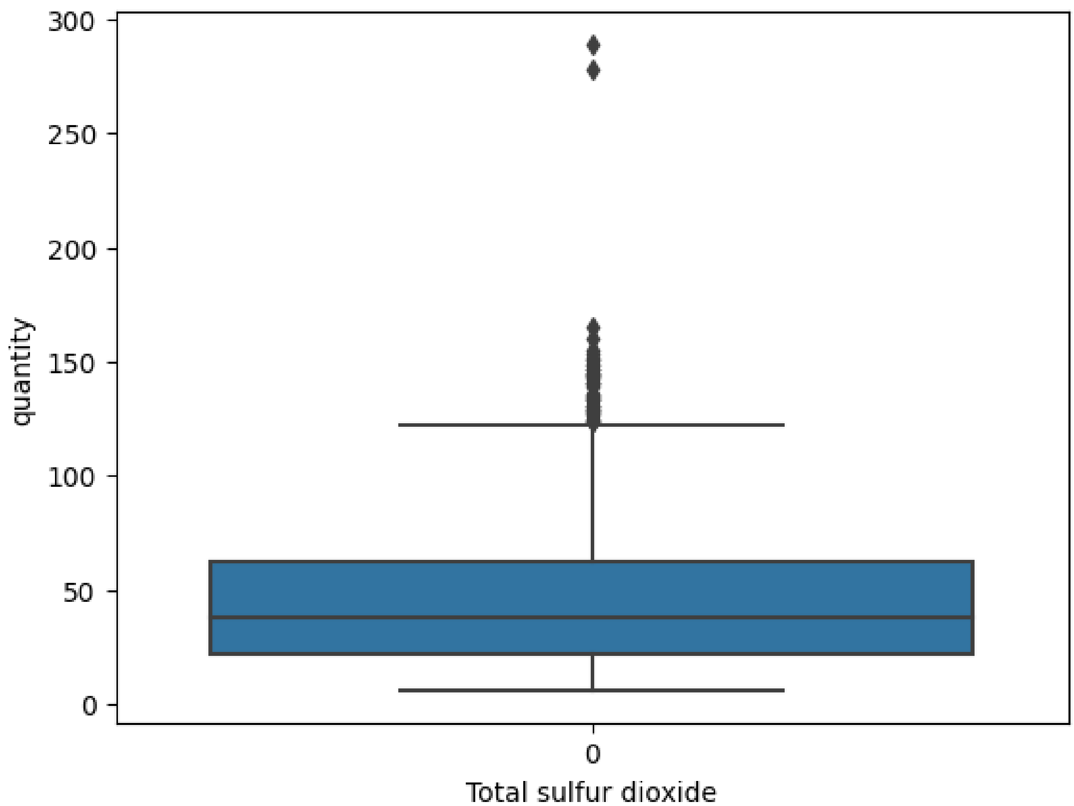
[332]:
```
sns.boxplot(dataframe['free sulfur dioxide']).set(ylabel = "quantity",xlabel =
↪"free sullfure dioxide")
```

[332]: [Text(0, 0.5, 'quantity'), Text(0.5, 0, 'free sullfure dioxide')]



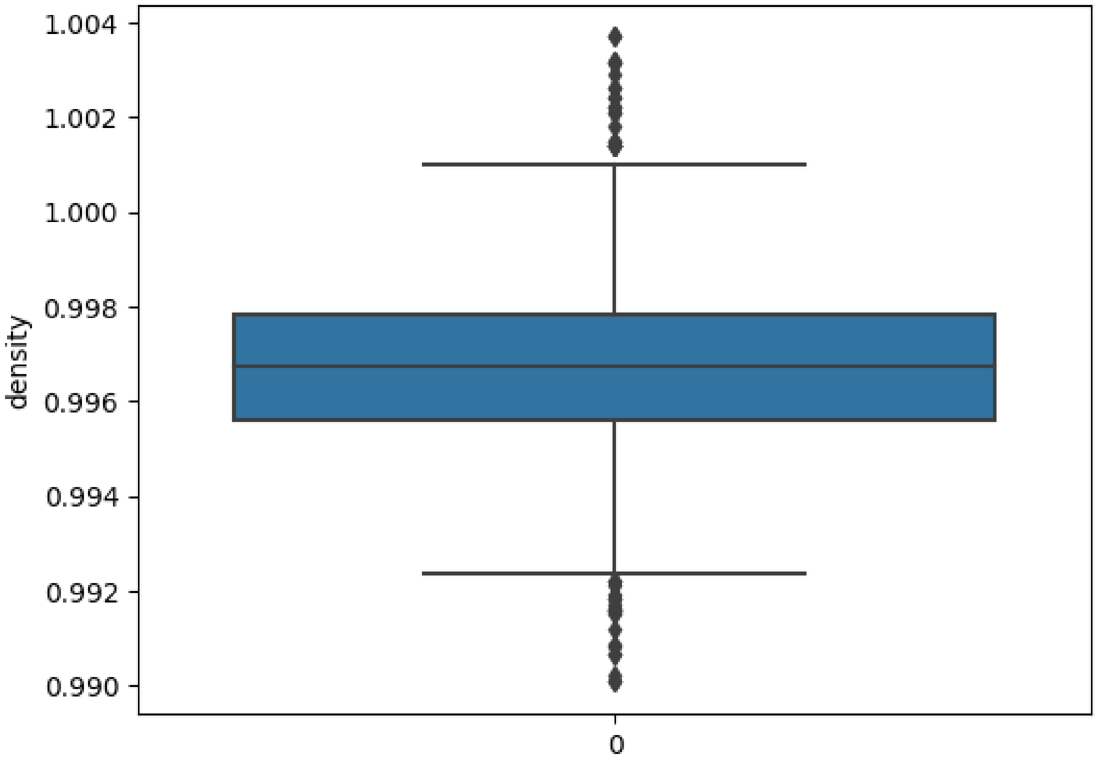

```
[333]: sns.boxplot(dataframe['total sulfur dioxide']).set(ylabel = "quantity",xlabel =␣
↪"Total sulfur dioxide")
```

```
[333]: [Text(0, 0.5, 'quantity'), Text(0.5, 0, 'Total sulfur dioxide')]
```



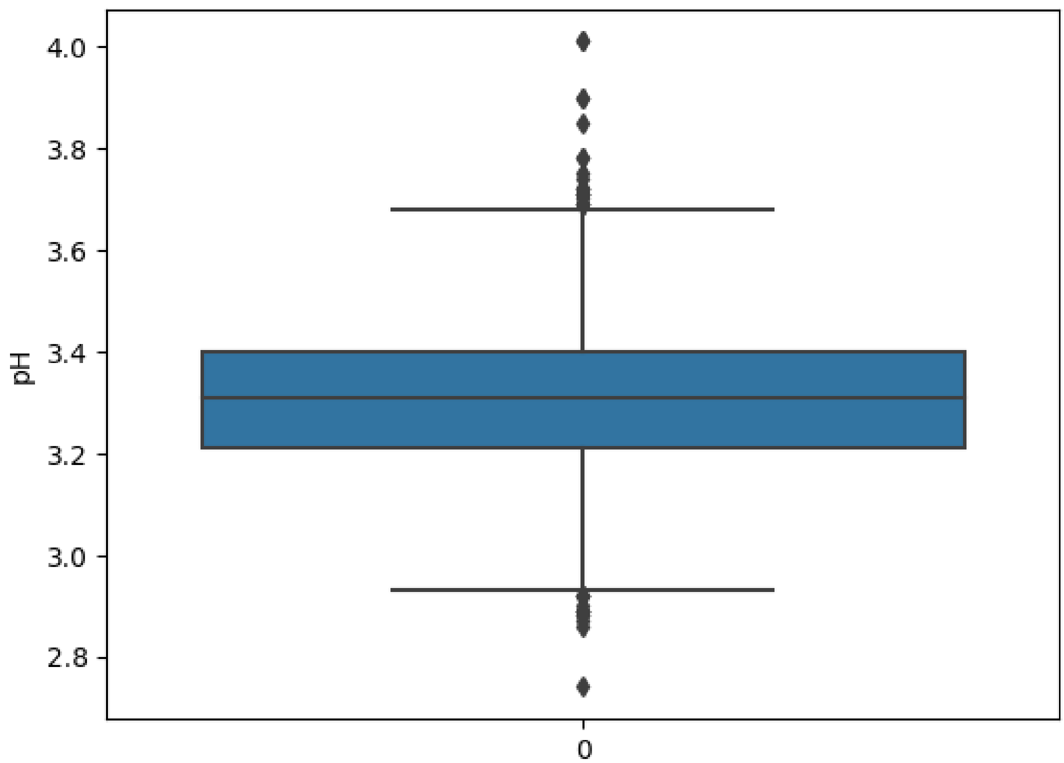
[334]:
```
sns.boxplot(dataframe['density']).set(ylabel = "density")
```

[334]: [Text(0, 0.5, 'density')]



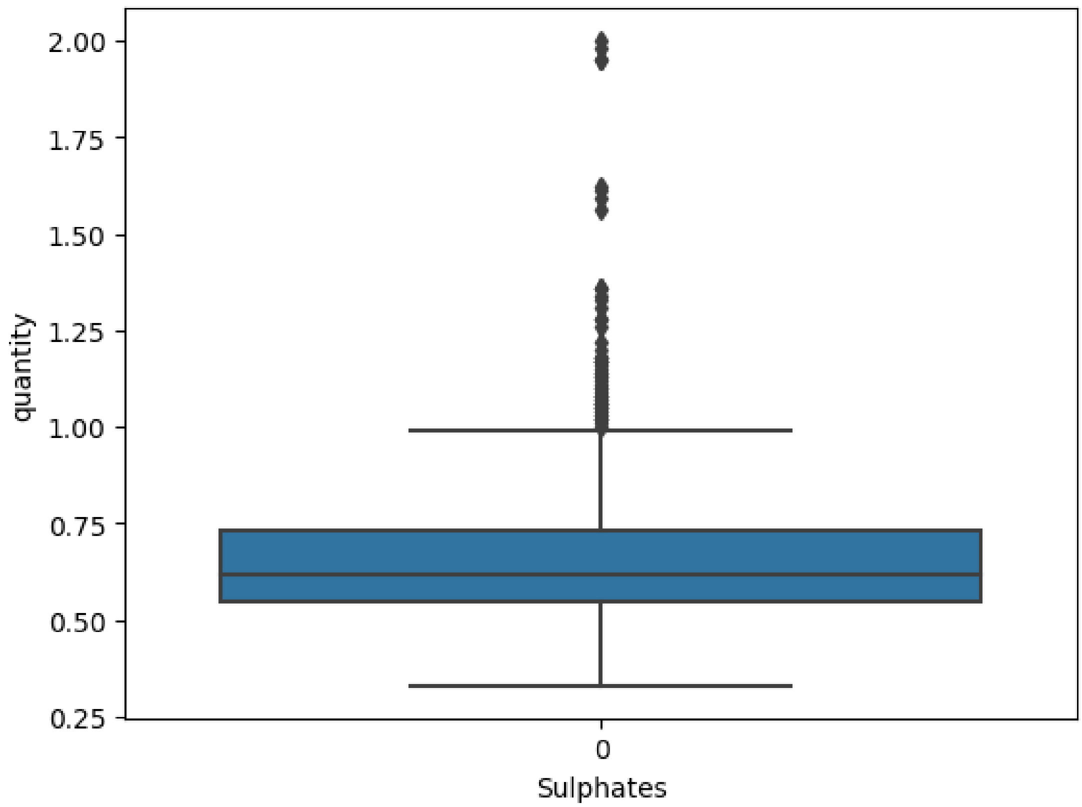
[335]:
```
sns.boxplot(dataframe['pH']).set(ylabel = "pH")
```

[335]: [Text(0, 0.5, 'pH')]



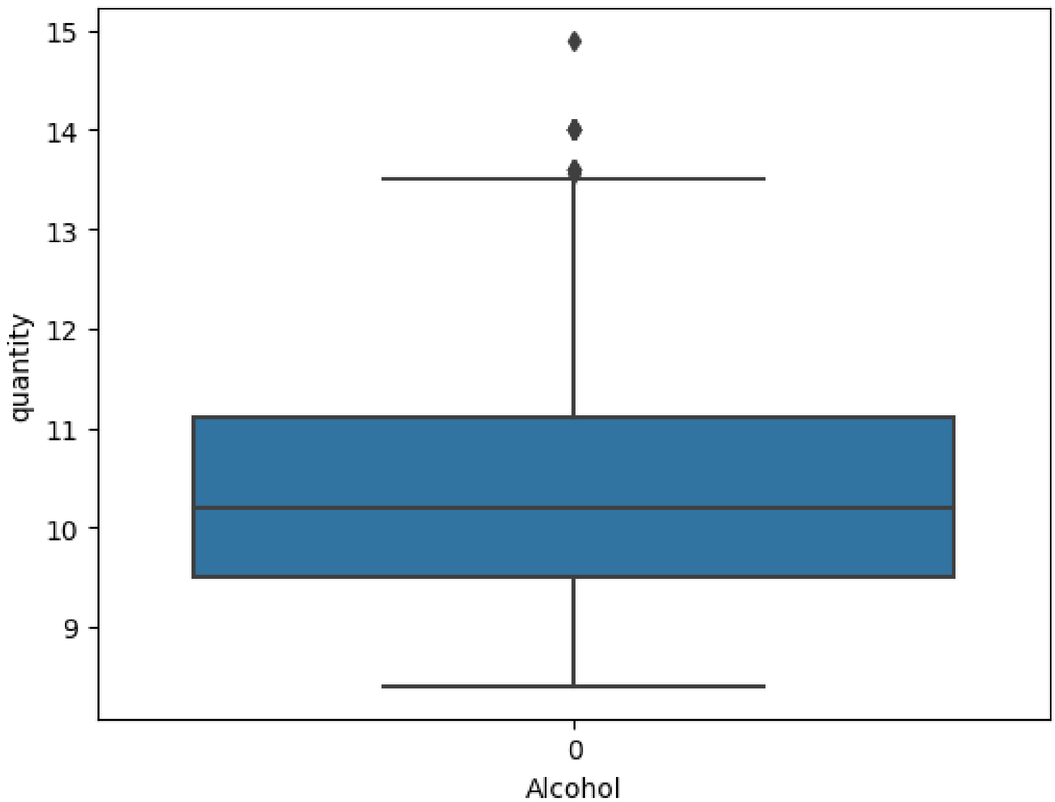
[336]:
```
sns.boxplot(dataframe['sulphates']).set(ylabel = "quantity",xlabel =␣
↪"Sulphates")
```

[336]: [Text(0, 0.5, 'quantity'), Text(0.5, 0, 'Sulphates')]



[337]:
```
sns.boxplot(dataframe['alcohol']).set(ylabel = "quantity",xlabel = "Alcohol")
```

[337]: [Text(0, 0.5, 'quantity'), Text(0.5, 0, 'Alcohol')]



[338]:
```
dataframe = dataframe.drop(dataframe[(dataframe['fixed acidity']>15)].index)
dataframe = dataframe.drop(dataframe[(dataframe['volatile acidity']>1.4)].index)
dataframe = dataframe.drop(dataframe[(dataframe['citric acid']>0.9)].index)
dataframe = dataframe.drop(dataframe[(dataframe['residual sugar']>15)].index)
dataframe = dataframe.drop(dataframe[(dataframe['chlorides']>0.5)].index)
dataframe = dataframe.drop(dataframe[(dataframe['free sulfur dioxide']>70)].
 ↪index)
dataframe = dataframe.drop(dataframe[(dataframe['total sulfur dioxide']>250)].
 ↪index)

dataframe = dataframe.drop(dataframe[(dataframe['pH']>4)].index)
dataframe = dataframe.drop(dataframe[(dataframe['pH']<2.8)].index)
dataframe = dataframe.drop(dataframe[(dataframe['sulphates']>1.75)].index)
dataframe = dataframe.drop(dataframe[(dataframe['alcohol']>14.5)].index)
```

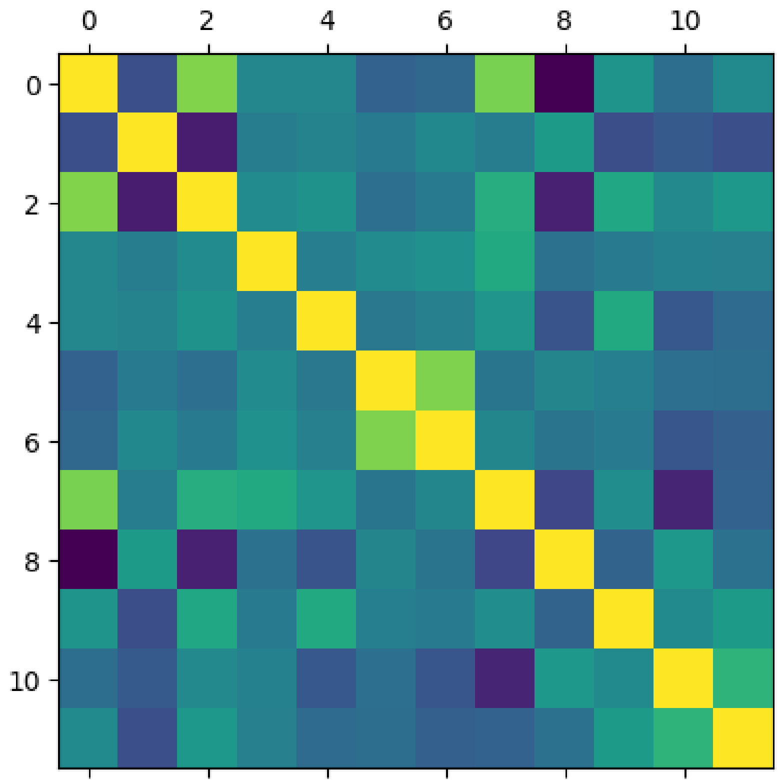
We now have removed most of the outliers.

[339]:
```
dataframe.shape
```

[339]: (1580, 12)

And we can see that 19 lines have been removed from the dataset.

[340]:
```
plt.matshow(dataframe.corr())
plt.show()
```



After removing the outliers we can see that the correlation diagram has changed a little and that some columns have a greater impact on the quality.

1.3 3 - Prediction

We separate the quality column from the dataset.

[341]:
```
X = dataframe.drop(columns=['quality'])
y = dataframe['quality']
```

We separate the dataset in two parts: train and test, 80% of the dataset going in the train part.

[342]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42,␣
↪test_size=0.2)
```

We will use classification algorithms to find out if the red wine is good.

[343]:
```
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X_train, y_train.ravel())
```

```
score = neigh.score(X_test, y_test)
print("Score: ", score)
```

```
Score:  0.8639240506329114
```

Using KNeighborsClassifier, we get a score of 0.86 rounded, which is a good result in this case.

[344]:
```
clf = LogisticRegression(random_state=0).fit(X_train, y_train)

print("test mean accuracy:")
print(clf.score(X_test, y_test))
```

```
test mean accuracy:
0.8639240506329114
```

```
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

With the LogisticRegression, the end result is 0.86 rouned, meaning we get a similar score compared to the previous method (same if we are to be precise).

[345]:
```
lsvc = LinearSVC(verbose=0, dual=False, C=1, multi_class='ovr',␣
 ↪fit_intercept=False)
lsvc.fit(X_train, y_train)

score = lsvc.score(X_test, y_test)
print("Score: ", score)
```

```
Score:  0.8670886075949367
```

The LinearSVC model gives us about the same score as the previous ones (0.86 rounded), but is slightly better when looking at the numbers after the coma.

[346]:
```
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, y_train)

score = rfc.score(X_test, y_test)
print("Score: ", score)
```

```
Score:  0.9272151898734177
```

And finally, when using the RandomForestClassifier method with several parameters, we reached a score of about 0.92 which is the best score out of all the models tested so far.

For each algorithm we tested, we used several parameters and kept the parameter that had the best score (hyperparamters used here).

1.4 4 - Conclusion

We came to the conclusion that the RandomForestClassifier provided the best results. This achievement was only possible through the use of the classification method. Initially, we attempted to predict the exact wine quality with regression algorithms but the results were not that good (about a score of 0.42 rounded). We then decided to predict whether the wine would be of good quality or bad quality, which gave us much better results. Finally, we removed outliers to increase the accuracy by a few percentage points, reaching a score of 0.92 like we saw just earlier.

Adress of the dataset: https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009.

Created in Deepnote