Machine Learning Lab 1

Data Preprocessing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing. Data preprocessing steps include but are not limited to data cleaning, data integration, data transformation, data reduction and data discretization.

The dataset

The dataset used to perform this experiment is the wine quality dataset, it is a combination of data on two types of wine variants, namely red wine and white wine, of the portuguese "Vinho Verde" wine. The dataset contains information on the parameters for fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol.

Experiment

In this experiment, I have performed operations to clean the data, done statistical analysis of the data and then used various visualization tools to visualize the data in different ways which in turn reveals different information about the data which are otherwise not easily discernible.

Using the pandas library in I loaded the red wine and white wine datasets into the memory from their respective csv files and then merged the two datasets into one single pandas dataframe.

Using the pandas.Dataframe.describe() function in pandas I calculated the various statistical measures of each of the columns of the dataset.

* The dataset has a total of 4898 rows.
* The means for each of the columns are calculated as:
    - Fixed acidity -> 6.85
    - Volatile acidity -> 0.27
    - Citric acid -> 0.33
    - Residual sugar -> 6.39
    - Chlorides -> 0.045
    - Free sulphur dioxides -> 35.30
    - Total sulphur dioxides -> 138.36
    - Density -> 0.99
    - pH -> 3.18
    - Sulphates -> 0.48
    - Alcohol -> 10.51

Using the pandas.Dataframe.dtypes method gives the datatypes of all the columns in the dataframe.

The pairplot function in seaborn library in python I was able to plot each column vs every other column in the dataset in the form of a scatter plot. And for also look at the values of each column in the form of a histogram.

Using the violin plot function in seaborn I am able to plot a violin plot for a column is the dataset. Violin plots are similar to box plots except that they also show the probability density of the data at different values. Thy also include a marker for median of the data and a box indicating the inter quartile range.

Using the violin plots I was able to infer that the median for the density lies between 1.00 and 0.99, and that for citric acid lies between 0.25 and 0.5, and for sulphates it lies between 0.6 and 0.4.

Seaborn library also allows to plot box plot for a dataset with its boxplot function. A boxplot depicts groups through their quartiles. It has whiskers indicating variability outside the upper and lower quartiles. Outliers are plotted as individual points as can be seen in the diagram in the jupyter notebook.

The code and plots can be found in the accompanying jupyter notebook.

Data Preprocessing

October 25, 2018

1 Lab Assignment 1

1.1 Data Preprocessing

1.1.1 Submitted to : Prof. Sweetlin Hemlatha

1.1.2 Submitted by: Prateek Singh (15BCE1091)

```
In [3]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt

        %matplotlib inline
```

```
In [3]: data = pd.read_csv("iris-data.csv")
        data.head()
```

```
Out[3]:    sepal_length_cm  sepal_width_cm  petal_length_cm  petal_width_cm  \
        0              5.1             3.5              1.4             0.2
        1              4.9             3.0              1.4             0.2
        2              4.7             3.2              1.3             0.2
        3              4.6             3.1              1.5             0.2
        4              5.0             3.6              1.4             0.2

                 class
        0  Iris-setosa
        1  Iris-setosa
        2  Iris-setosa
        3  Iris-setosa
        4  Iris-setosa
```

```
In [4]: data.dtypes
```

```
Out[4]: sepal_length_cm    float64
        sepal_width_cm     float64
        petal_length_cm    float64
        petal_width_cm     float64
        class               object
        dtype: object
```

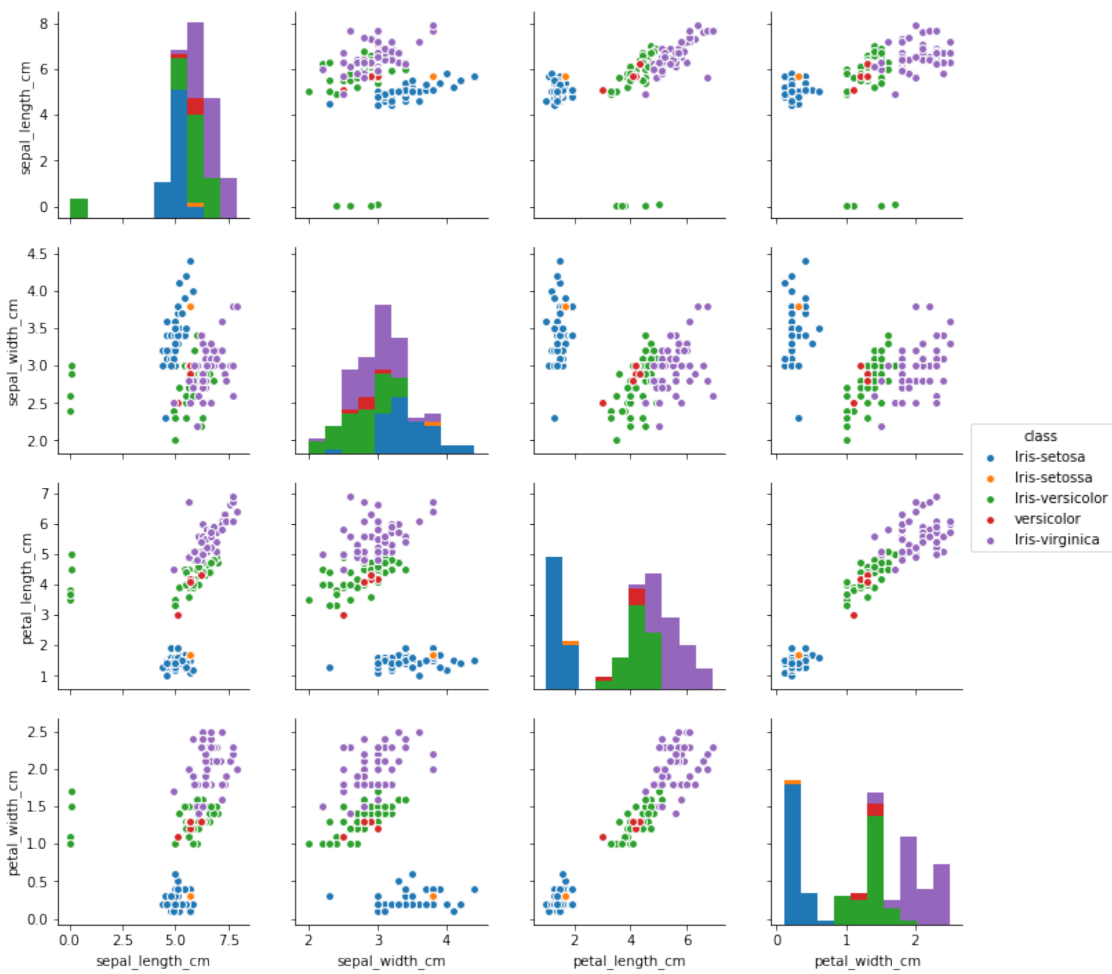
```
In [5]: data.describe()
```

Out[5]:

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm
count	150.000000	150.000000	150.000000	145.000000
mean	5.644627	3.054667	3.758667	1.236552
std	1.312781	0.433123	1.764420	0.755058
min	0.055000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.400000
50%	5.700000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
In [6]: sns.pairplot(data=data.dropna(),hue='class')
```

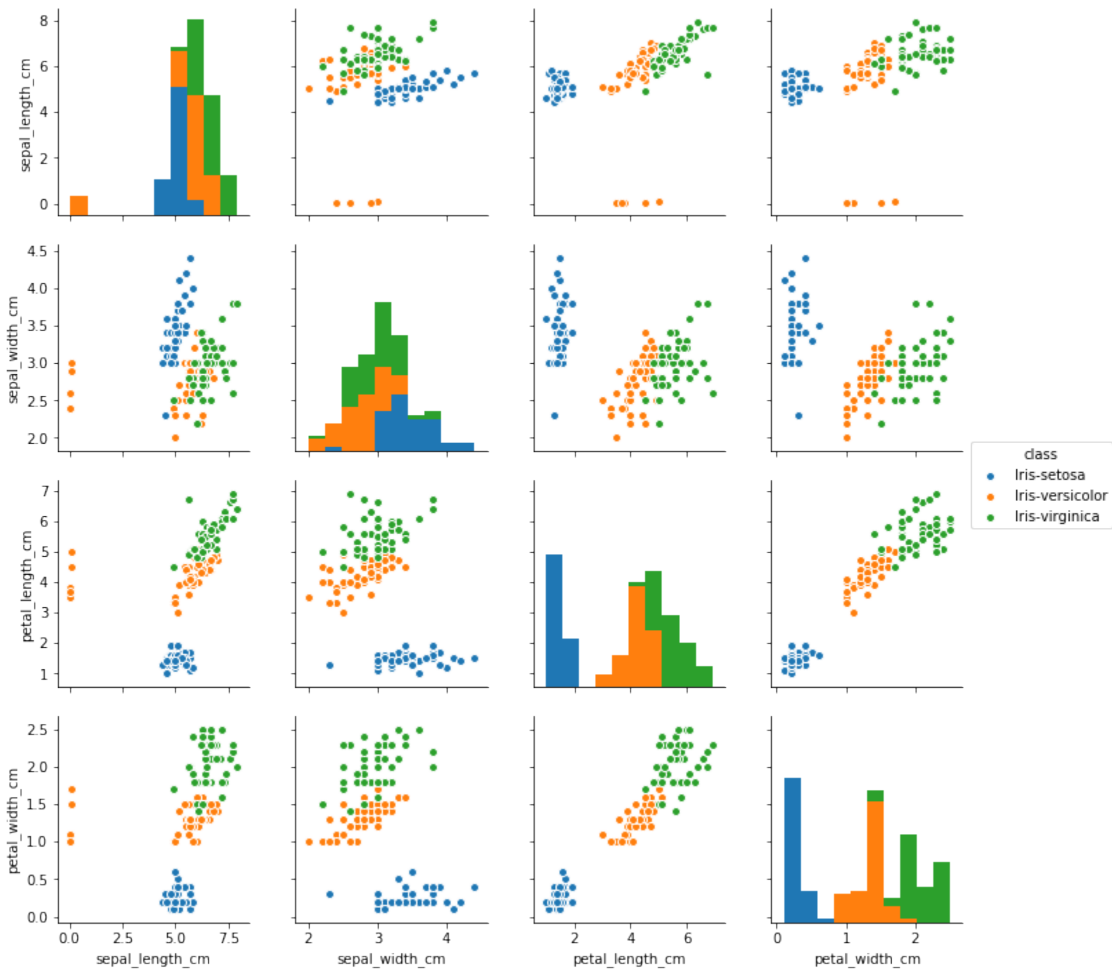
```
Out[6]: <seaborn.axisgrid.PairGrid at 0x7fddb39ba400>
```




```
In [7]: # Inference from previous cell missmatching class label (versicolor and setosa)
        data.loc[data["class"] == "versicolor","class"] = "Iris-versicolor"
        data.loc[data["class"] == "Iris-setossa","class"] = "Iris-setosa"
```

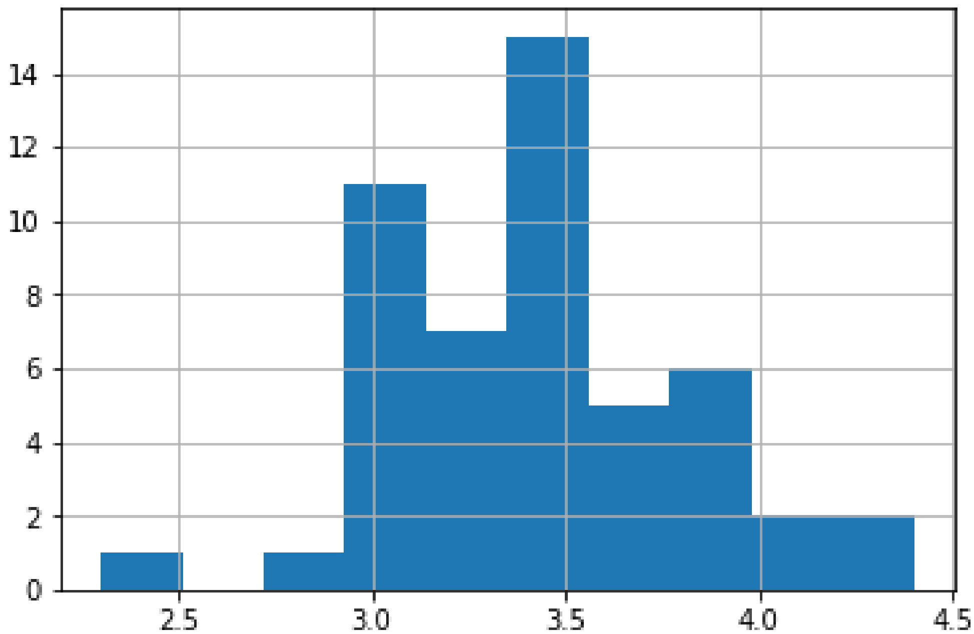
```
In [8]: sns.pairplot(data=data.dropna(),hue='class') #Plot after fixing class labels
```

```
Out[8]: <seaborn.axisgrid.PairGrid at 0x7fddb0fed080>
```




```
In [9]: data.loc[data["class"]=="Iris-setosa","sepal_width_cm"].hist()
```

```
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7fddaabda7b8>
```



In [10]:
```
data.loc[data["petal_width_cm"].isnull()]
```

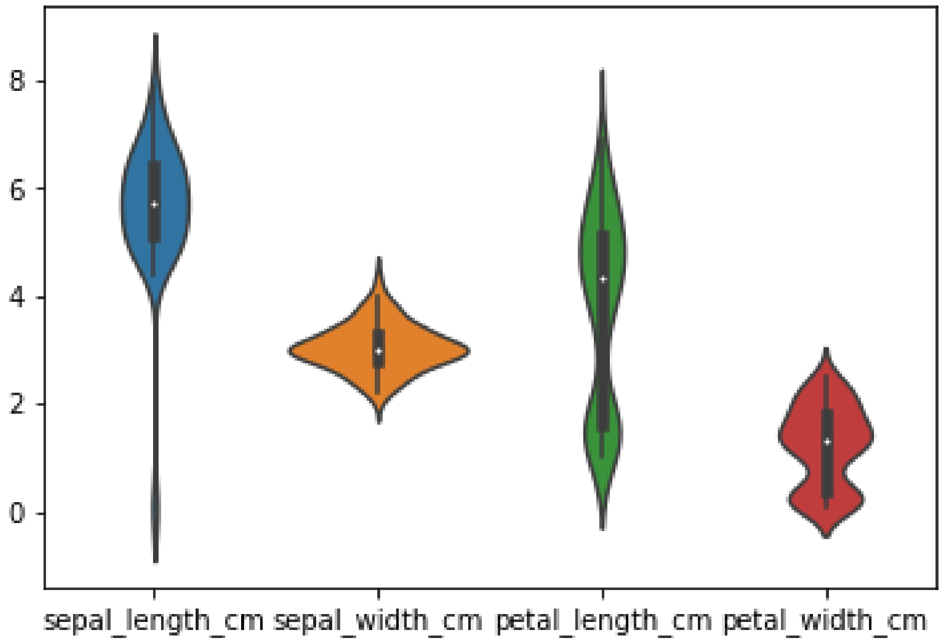
Out[10]:
```
    sepal_length_cm  sepal_width_cm  petal_length_cm  petal_width_cm  \
7               5.0             3.4              1.5             NaN
8               4.4             2.9              1.4             NaN
9               4.9             3.1              1.5             NaN
10              5.4             3.7              1.5             NaN
11              4.8             3.4              1.6             NaN

          class
7   Iris-setosa
8   Iris-setosa
9   Iris-setosa
10  Iris-setosa
11  Iris-setosa
```

In [11]:
```
#Fixing missing values of petal_width_cm from data.describe() mean
data.loc[data["petal_width_cm"].isnull(),"petal_width_cm"] = 1.236552
data.loc[data["petal_width_cm"].isnull()]
```

Out[11]:
```
Empty DataFrame
Columns: [sepal_length_cm, sepal_width_cm, petal_length_cm, petal_width_cm, class]
Index: []
```

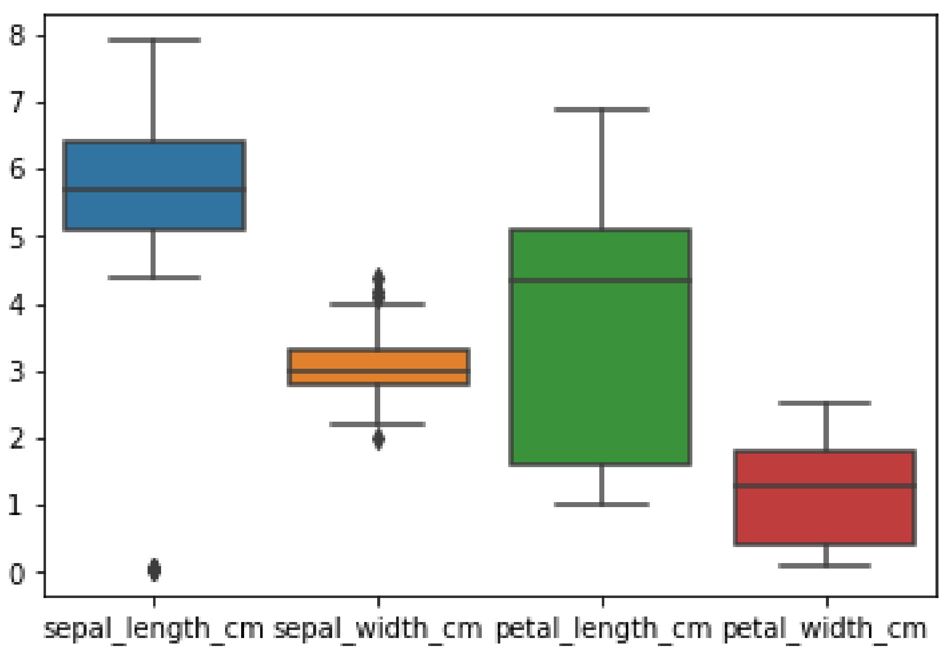
In [12]:
```
sns.violinplot(data=data) # Violin Plot// They represent probability density also whe
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fddaa45af60>
```




```
In [13]: sns.boxplot(data=data) #Box plot representing mean and quantiles
```

```
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7fddaa3e5ac8>
```



Applying the above functions to my own dataset

In [4]:
```
wine_data = pd.read_csv('../Dataset/winequality-white.csv', sep=';')
red_wine = pd.read_csv('../Dataset/winequality-red.csv', sep=';')
sns.set(style='whitegrid', context='notebook', font_scale=1)
```

In [5]:
```
    wine_data.append(red_wine)
wine_data["quality"] = wine_data["quality"].astype(str)
wine_data.head(10)
```

Out[5]:
```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.0              0.27         0.36            20.7      0.045
1            6.3              0.30         0.34             1.6      0.049
2            8.1              0.28         0.40             6.9      0.050
3            7.2              0.23         0.32             8.5      0.058
4            7.2              0.23         0.32             8.5      0.058
5            8.1              0.28         0.40             6.9      0.050
6            6.2              0.32         0.16             7.0      0.045
7            7.0              0.27         0.36            20.7      0.045
8            6.3              0.30         0.34             1.6      0.049
9            8.1              0.22         0.43             1.5      0.044

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 45.0                 170.0   1.0010  3.00       0.45
1                 14.0                 132.0   0.9940  3.30       0.49
2                 30.0                  97.0   0.9951  3.26       0.44
3                 47.0                 186.0   0.9956  3.19       0.40
4                 47.0                 186.0   0.9956  3.19       0.40
5                 30.0                  97.0   0.9951  3.26       0.44
6                 30.0                 136.0   0.9949  3.18       0.47
7                 45.0                 170.0   1.0010  3.00       0.45
8                 14.0                 132.0   0.9940  3.30       0.49
9                 28.0                 129.0   0.9938  3.22       0.45

   alcohol quality
0      8.8       6
1      9.5       6
2     10.1       6
3      9.9       6
4      9.9       6
5     10.1       6
6      9.6       6
7      8.8       6
8      9.5       6
9     11.0       6
```

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

Out[6]:
```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    4898.000000       4898.000000  4898.000000     4898.000000
```

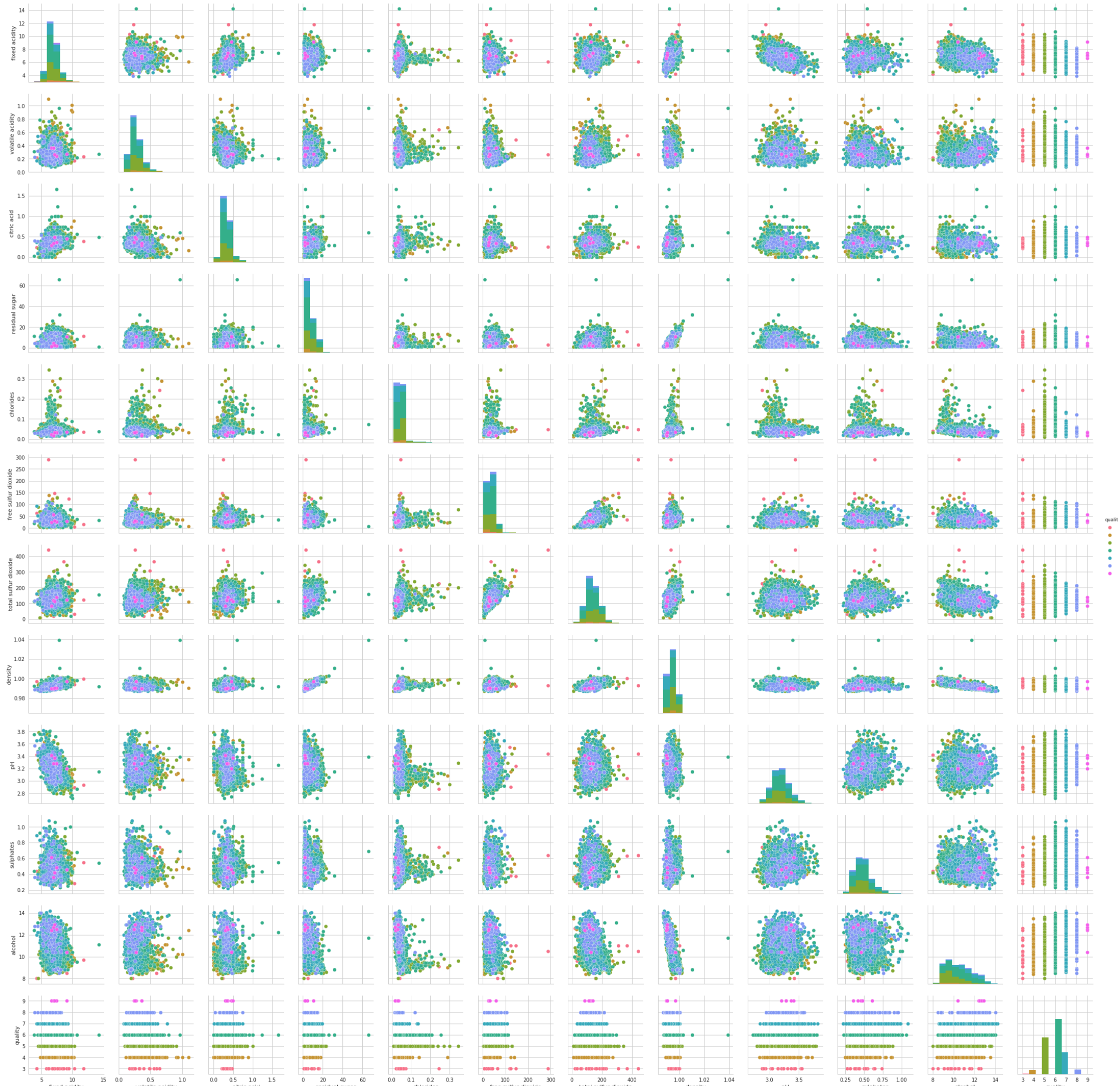
```
mean        6.854788            0.278241     0.334192         6.391415
std         0.843868            0.100795     0.121020         5.072058
min         3.800000            0.080000     0.000000         0.600000
25%         6.300000            0.210000     0.270000         1.700000
50%         6.800000            0.260000     0.320000         5.200000
75%         7.300000            0.320000     0.390000         9.900000
max        14.200000            1.100000     1.660000        65.800000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  4898.000000          4898.000000           4898.000000  4898.000000
mean      0.045772            35.308085            138.360657     0.994027
std       0.021848            17.007137             42.498065     0.002991
min       0.009000             2.000000              9.000000     0.987110
25%       0.036000            23.000000            108.000000     0.991723
50%       0.043000            34.000000            134.000000     0.993740
75%       0.050000            46.000000            167.000000     0.996100
max       0.346000           289.000000            440.000000     1.038980

                pH    sulphates      alcohol
count  4898.000000  4898.000000  4898.000000
mean      3.188267     0.489847    10.514267
std       0.151001     0.114126     1.230621
min       2.720000     0.220000     8.000000
25%       3.090000     0.410000     9.500000
50%       3.180000     0.470000    10.400000
75%       3.280000     0.550000    11.400000
max       3.820000     1.080000    14.200000
```

In [7]:
```
wine_data.dtypes
```

Out[7]:
```
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
quality                  object
dtype: object
```

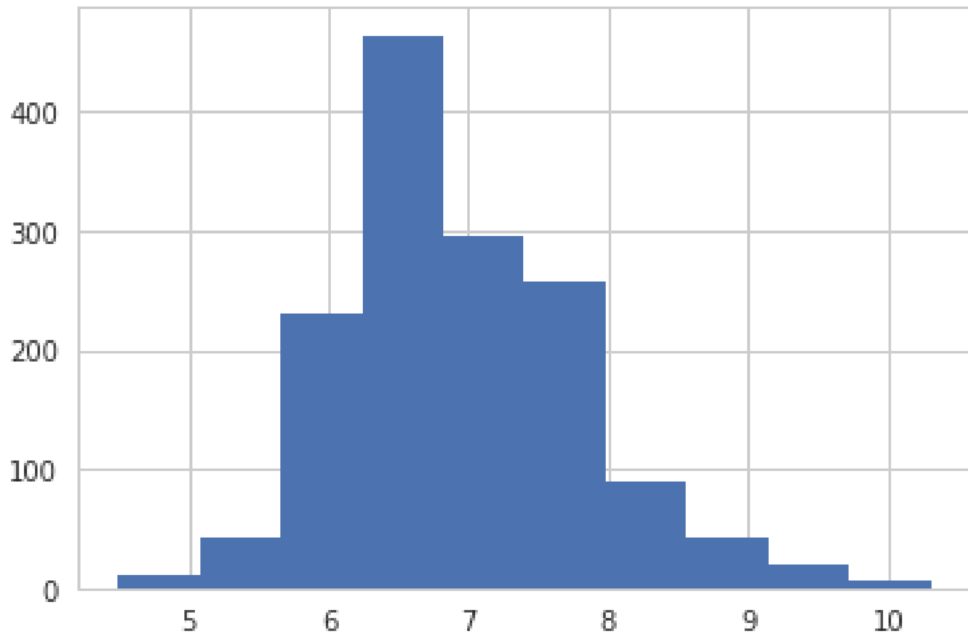
In [8]:
```
sns.pairplot(wine_data.dropna(), size=2.5, hue="quality")
```

Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x7fe2f539cdd8>
```



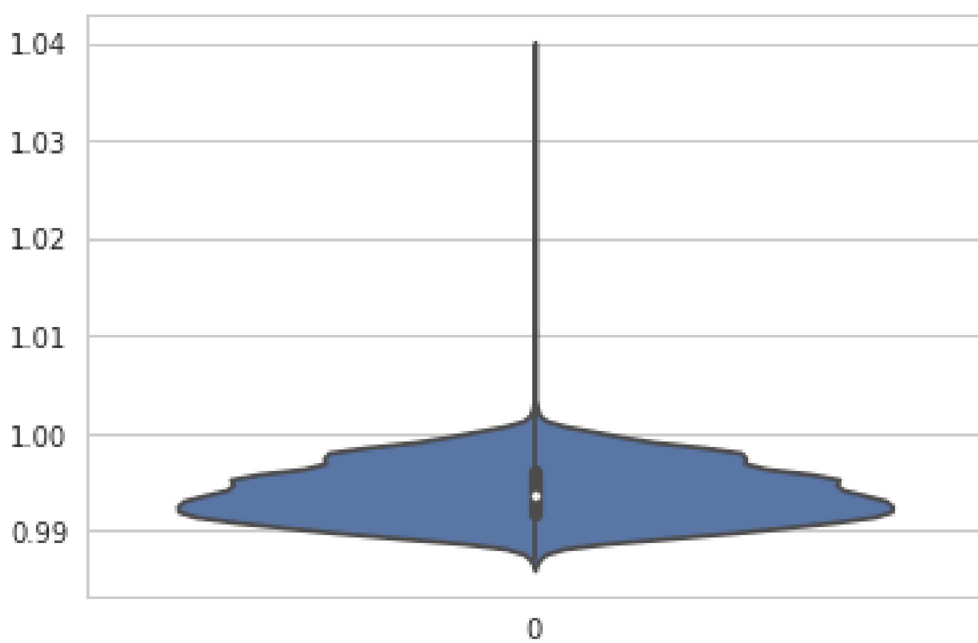
In [9]: `wine_data.loc[wine_data["quality"]=='5',"fixed acidity"].hist()`

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe2eb803b38>



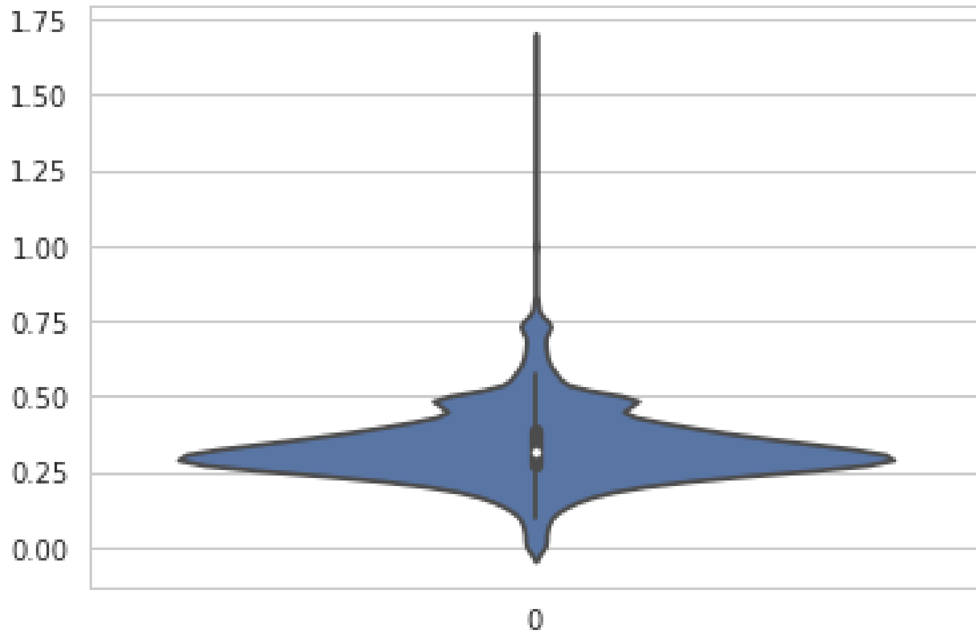

```
In [57]: sns.violinplot(data=wine_data["density"], size=10)
```

```
Out[57]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6474553b70>
```



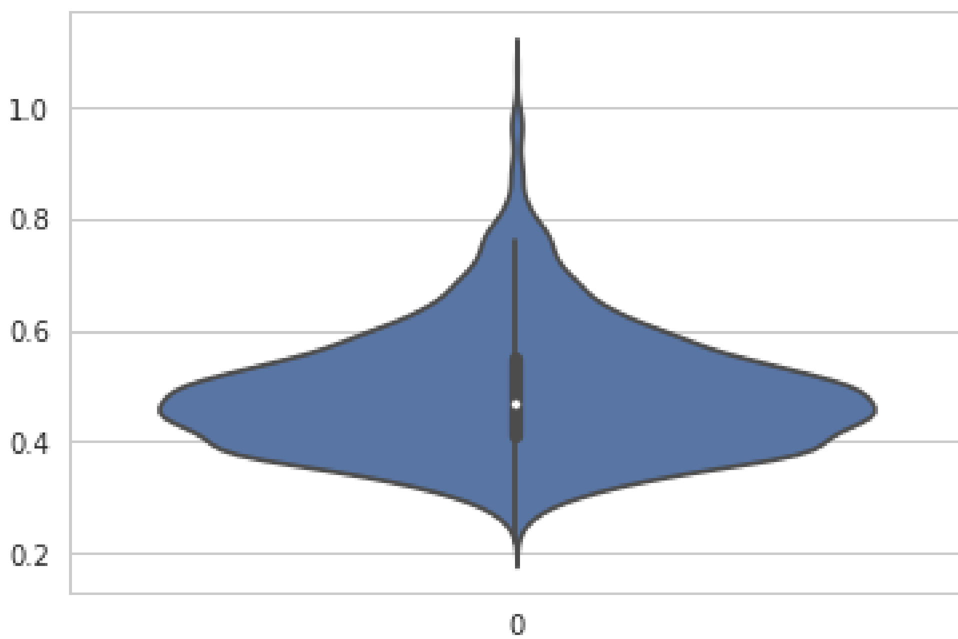
```
In [44]: sns.violinplot(data=wine_data["citric acid"], size=10)
```

```
Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7f647c0660f0>
```



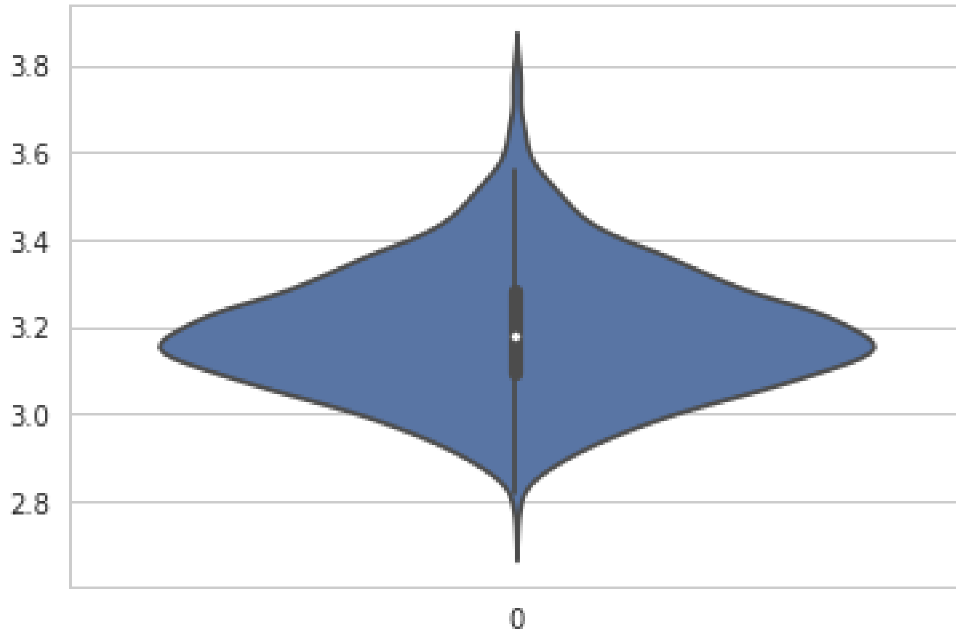

```
In [45]: sns.violinplot(data=wine_data["sulphates"], size=10)
```

```
Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6476e25e10>
```



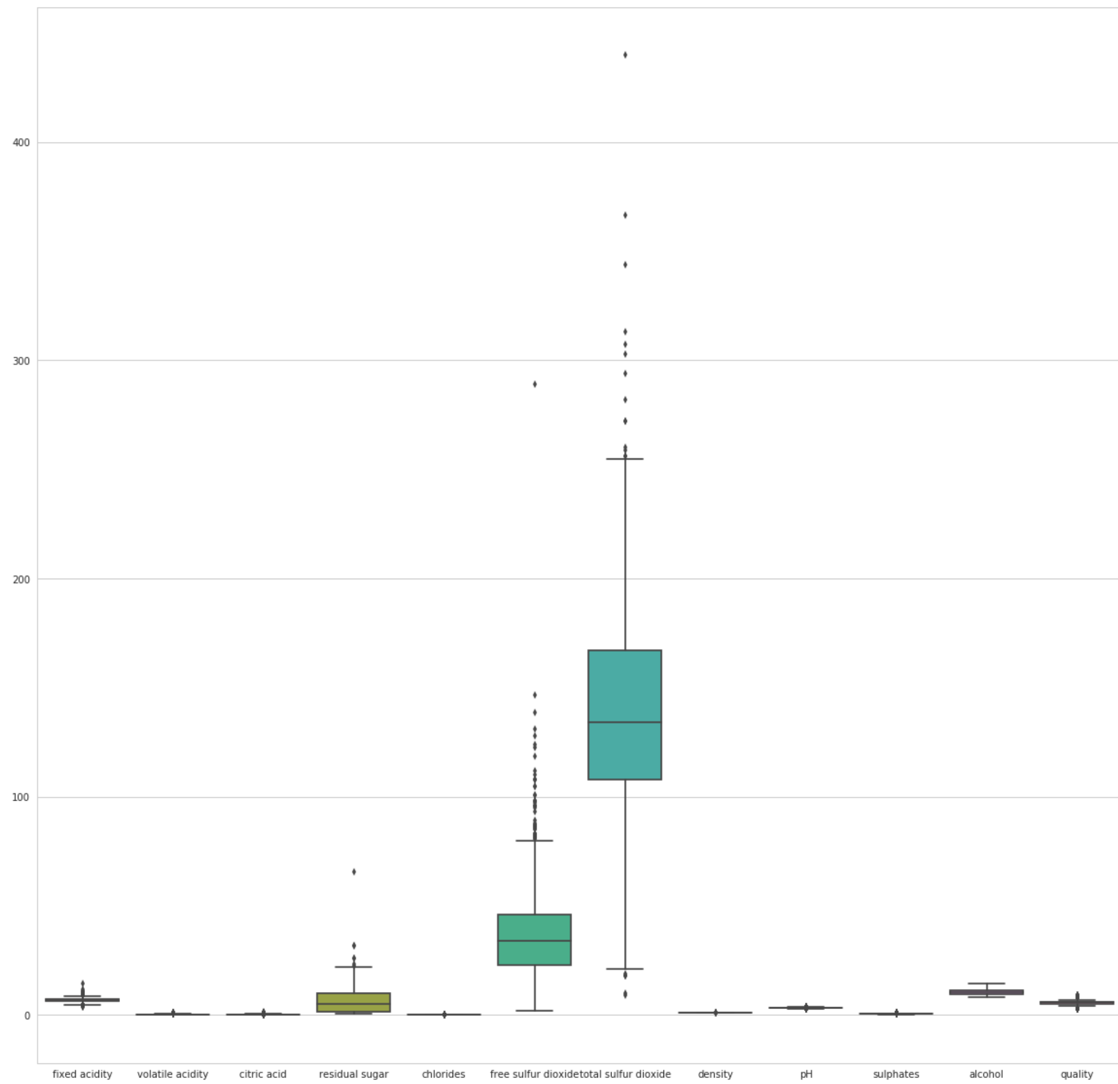
In [46]:
```python
sns.violinplot(data=wine_data["pH"], size=10)
```

Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6476d8f630>



In [55]:
```python
fig, ax = plt.subplots(figsize=(20, 20))
# seaborn.violinplot(ax=ax, data=df, **violin_options)
sns.boxplot(ax=ax, data=wine_data)
```

Out[55]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6476575ef0>


400
300
200
100
0
fixed acidity
volatile acidity
citric acid
residual sugar
chlorides
free sulfur dioxide
total sulfur dioxide
density
pH
sulphates
alcohol
quality