In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv('winequalityN.csv')
df.head()
```

Out[2]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

In [3]:
```
# statistical info
df.describe()
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	6487.000000	6489.000000	6494.000000	6495.000000	6495.000000	6497.000000	6497.000000	6497.000000	6488.000000	6493.000000	6497.000000	6497.000000
mean	7.216579	0.339691	0.318722	5.444326	0.056042	30.525319	115.744574	0.994697	3.218395	0.531215	10.491801	5.818378
std	1.296750	0.164649	0.145265	4.758125	0.035036	17.749400	56.521855	0.002999	0.160748	0.148814	1.192712	0.873255
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000	8.000000	3.000000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000	9.500000	5.000000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000	10.300000	6.000000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000	11.300000	6.000000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000	14.900000	9.000000

In [4]:
```
# datatype info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   type                  6497 non-null   object 
 1   fixed acidity         6487 non-null   float64
 2   volatile acidity      6489 non-null   float64
 3   citric acid           6494 non-null   float64
 4   residual sugar        6495 non-null   float64
 5   chlorides             6495 non-null   float64
 6   free sulfur dioxide   6497 non-null   float64
 7   total sulfur dioxide  6497 non-null   float64
 8   density               6497 non-null   float64
 9   pH                    6488 non-null   float64
 10  sulphates             6493 non-null   float64
 11  alcohol               6497 non-null   float64
 12  quality               6497 non-null   int64  
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
```

In [5]:
```
# check for null values
df.isnull().sum()
```

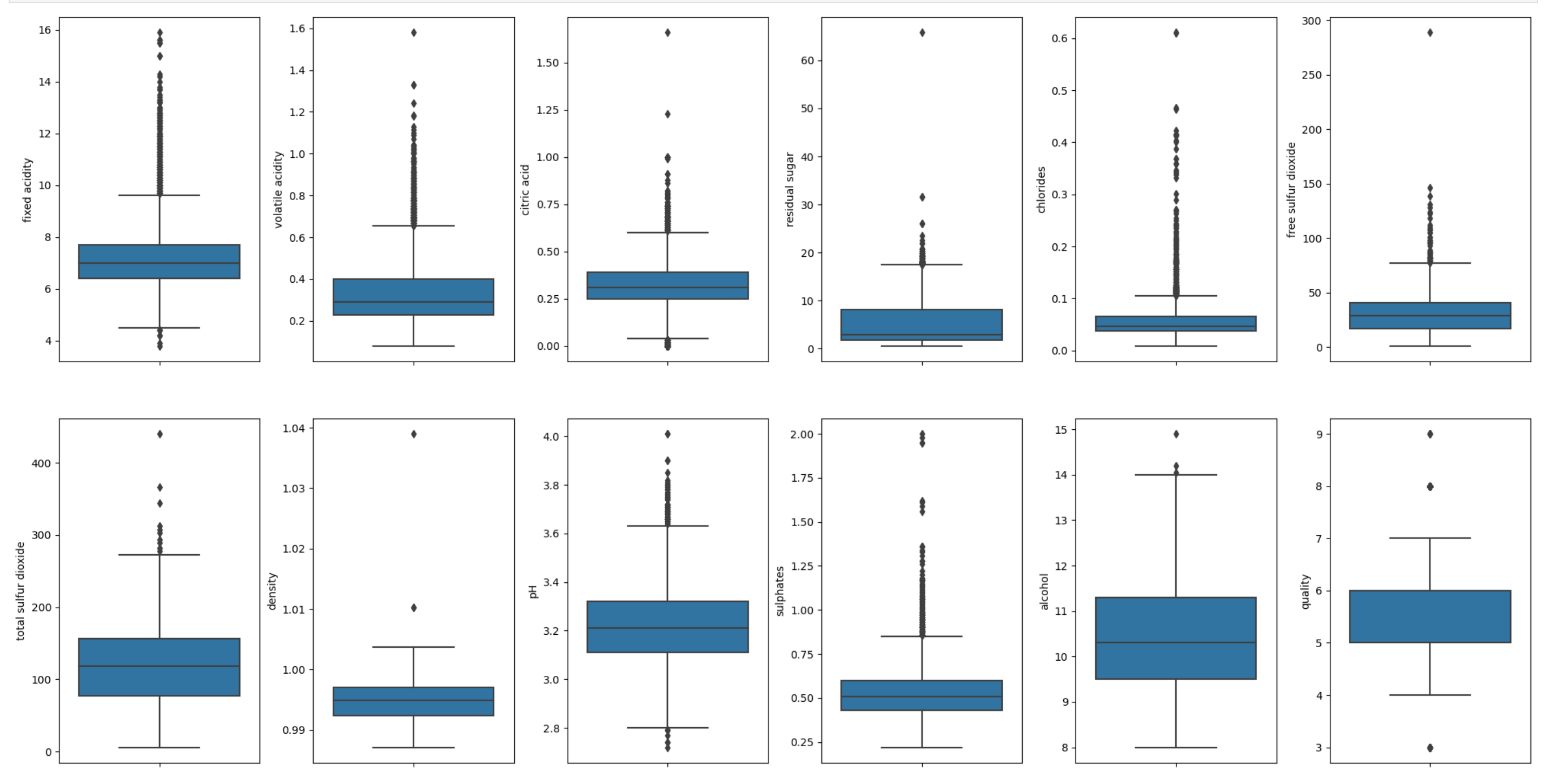
Out[5]:
```
type                     0
fixed acidity           10
volatile acidity         8
citric acid              3
residual sugar           2
chlorides                2
free sulfur dioxide      0
total sulfur dioxide     0
density                  0
pH                       9
sulphates                4
alcohol                  0
quality                  0
dtype: int64
```

In [6]:
```
# fill the missing values
for col, value in df.items():
    if col != 'type':
        df[col] = df[col].fillna(df[col].mean())
```

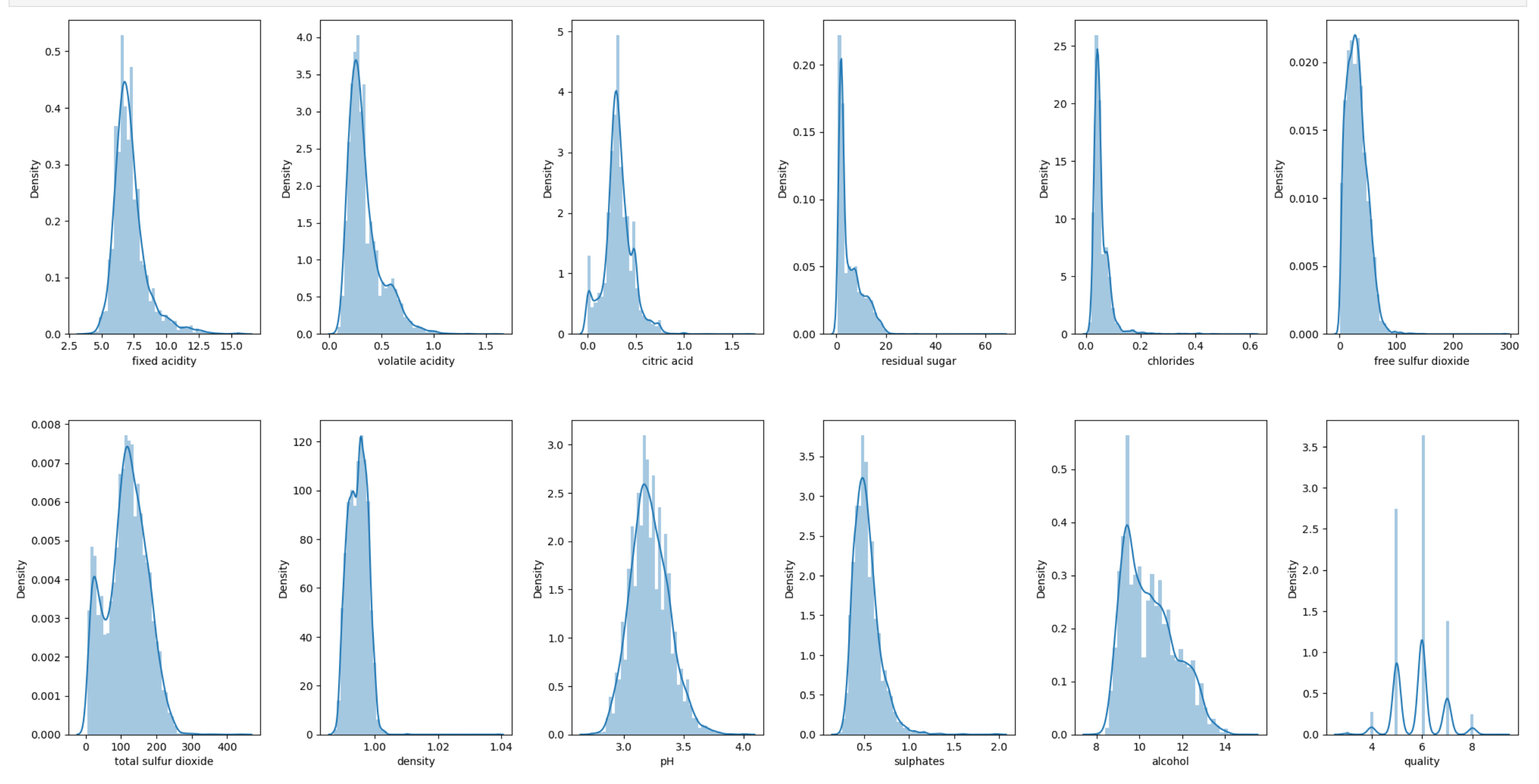
In [7]:
```
df.isnull().sum()
```

Out[7]:
```
type                    0
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
```

In [8]:
```
# create box plots
fig, ax = plt.subplots(ncols=6, nrows=2, figsize=(20,10))
index = 0
ax = ax.flatten()

for col, value in df.items():
    if col != 'type':
        sns.boxplot(y=col, data=df, ax=ax[index])
        index += 1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```



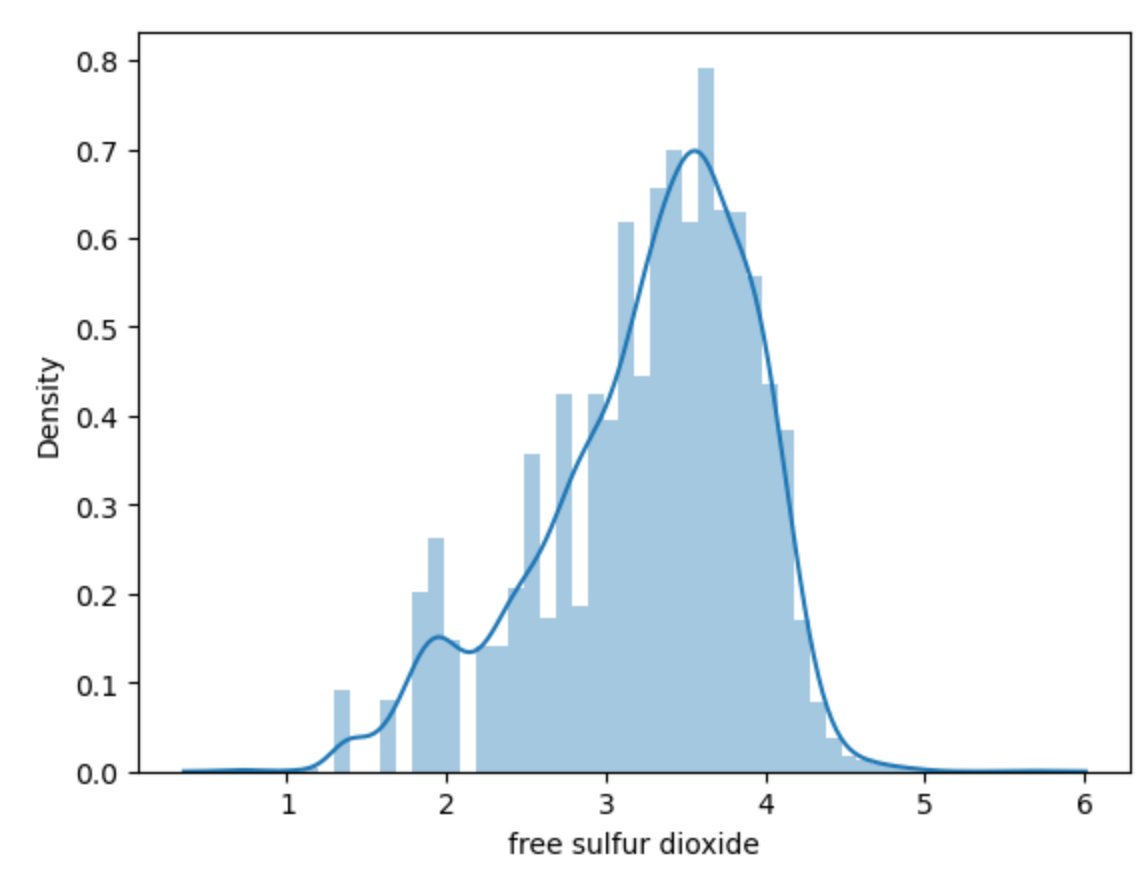
In [9]:
```
# create dist plot
fig, ax = plt.subplots(ncols=6, nrows=2, figsize=(20,10))
index = 0
ax = ax.flatten()

for col, value in df.items():
    if col != 'type':
        sns.distplot(value, ax=ax[index])
        index += 1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```



In [10]:
```
# log transformation
df['free sulfur dioxide'] = np.log(1 + df['free sulfur dioxide'])
```

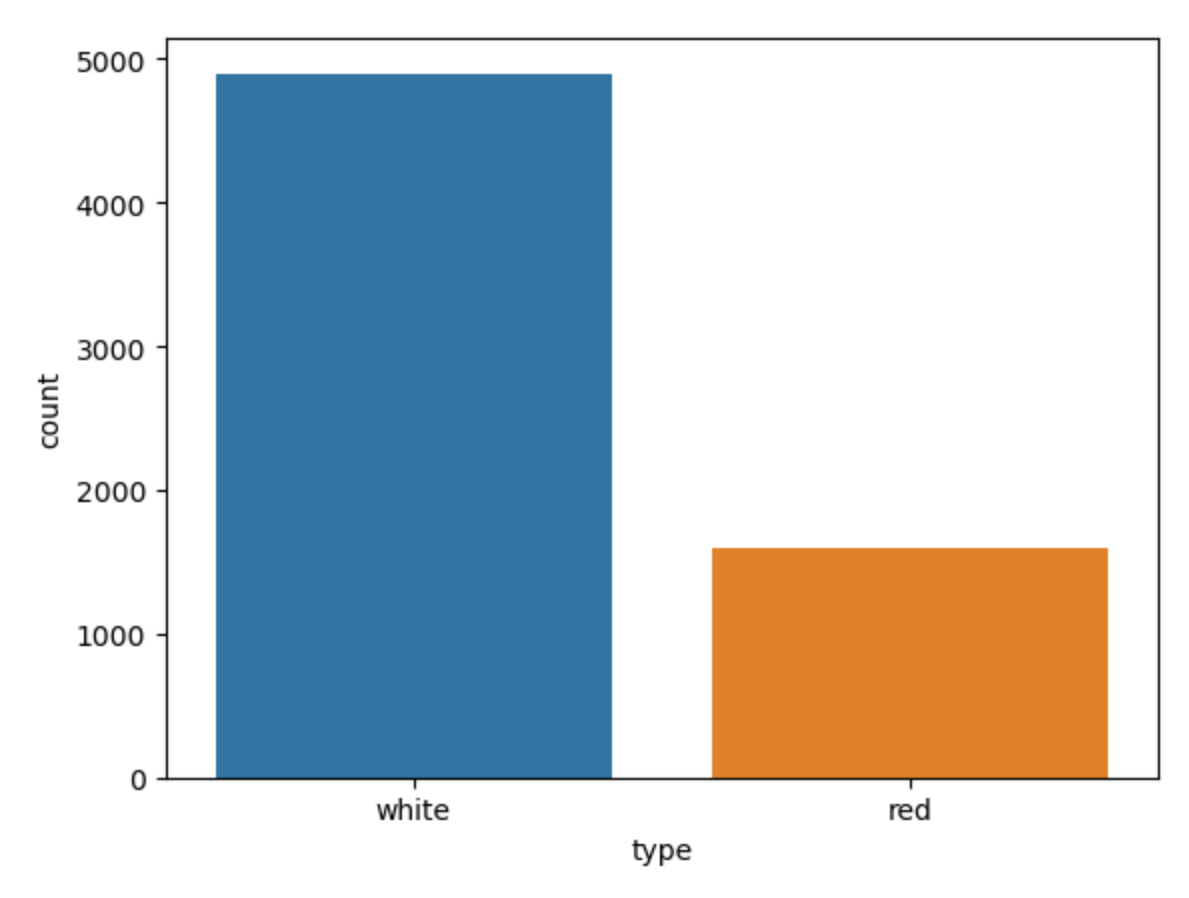
In [11]:
```
sns.distplot(df['free sulfur dioxide'])
```

Out[11]: <AxesSubplot:xlabel='free sulfur dioxide', ylabel='Density'>



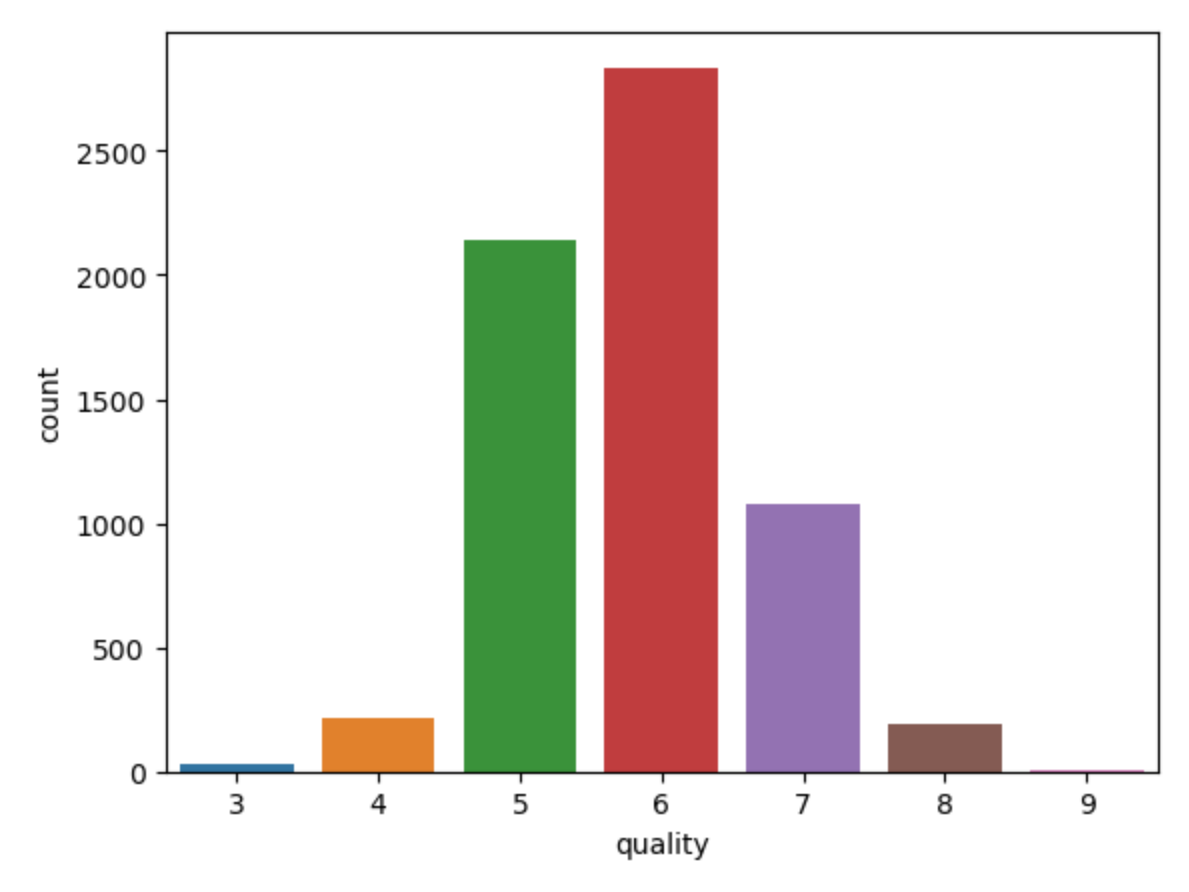
In [12]:
```
sns.countplot(df['type'])
```

Out[12]: <AxesSubplot:xlabel='type', ylabel='count'>



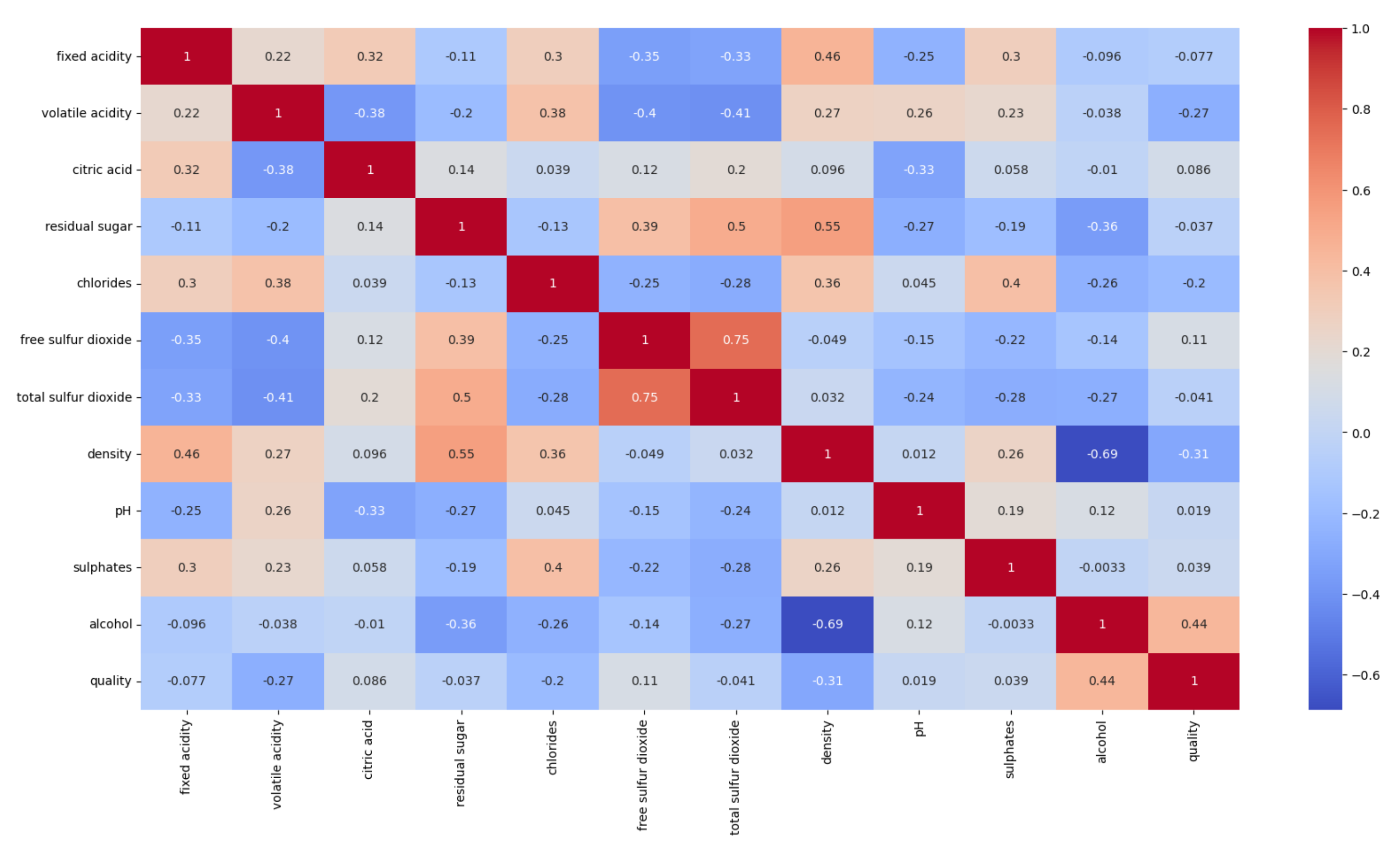
In [13]:
```
sns.countplot(df['quality'])
```

Out[13]: <AxesSubplot:xlabel='quality', ylabel='count'>



In [14]:
```
corr = df.corr()
plt.figure(figsize=(20,10))
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

Out[14]: <AxesSubplot:>



In [15]:
```
x = df.drop(columns=['type', 'quality'])
y = df['quality']
```

In [16]:
```
y.value_counts()
```

Out[16]:
```
6    2836
5    2138
7    1079
4     216
8     193
3      30
9       5
Name: quality, dtype: int64
```

In [17]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state = 42)
```

In [18]:
```
reg_model = LinearRegression()
```

In [19]:
```
reg_model.fit(x_train,y_train)
```

Out[19]: LinearRegression()

In [20]:
```
y_pred = reg_model.predict(x_test)
```

In [21]:
```
y_pred
```

Out[21]:
```
array([6.40413403, 6.05030782, 5.80728141, ..., 5.62784588, 5.75685697,
       5.44931822])
```

In [22]:
```
from sklearn.metrics import mean_squared_error
```

In [23]:
```
##cross validation
from sklearn.model_selection import cross_val_score
mse = cross_val_score(reg_model,x_train,y_train,scoring="neg_mean_squared_error",cv=10)
```

In [24]:
```
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 0.46998806175910307
```

In [25]:
```
np.mean(mse)
```

Out[25]: 0.46998806175910307

In [26]:
```
print(x_train.shape,y_train.shape)
```

```
(5197, 11) (5197,)
```

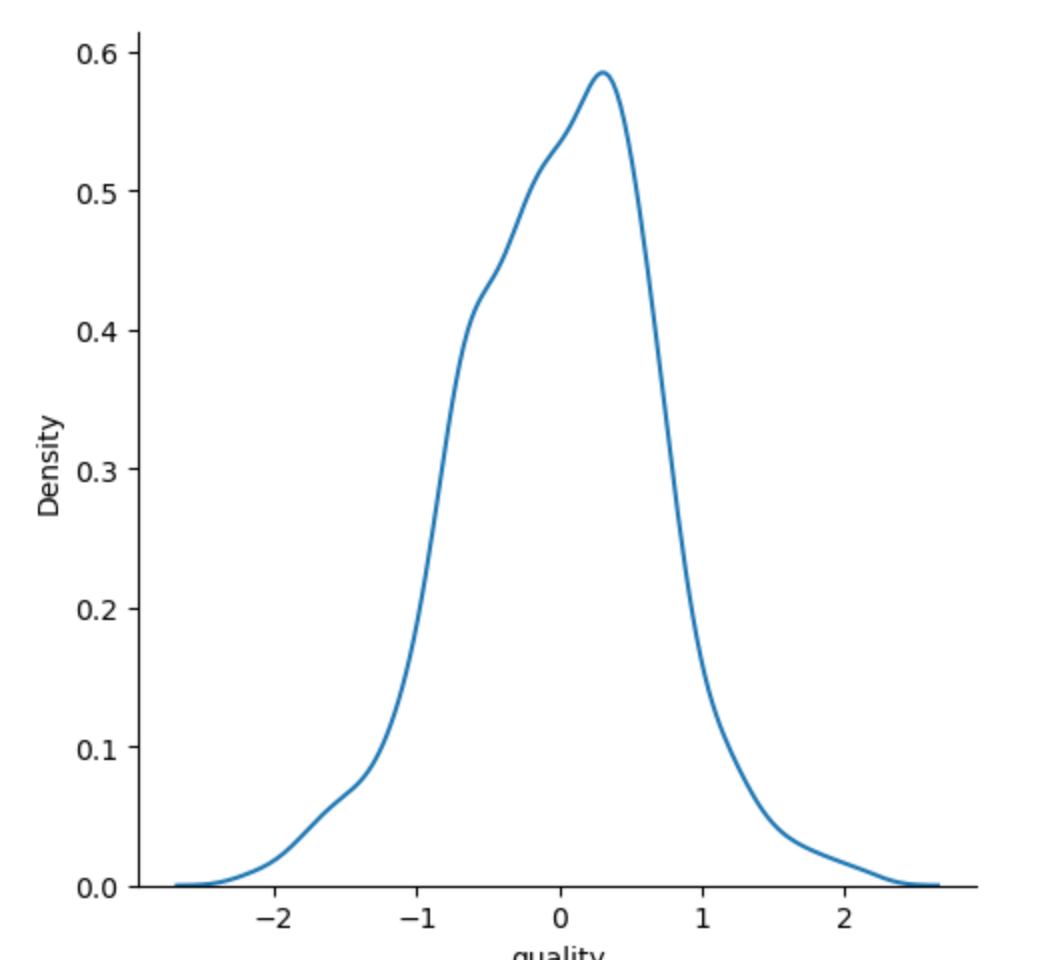
In [27]:
```
##prediction
reg_pred=reg_model.predict(x_test)
```

In [28]:
```
reg_pred
```

Out[28]:
```
array([6.40413403, 6.05030782, 5.80728141, ..., 5.62784588, 5.75685697,
       5.44931822])
```

In [29]:
```
import seaborn as sns
sns.displot(reg_pred-y_test,kind='kde')
```

Out[29]: <seaborn.axisgrid.FacetGrid at 0x19aafd4cd30>



In [30]:
```
from sklearn.metrics import r2_score
```

In [31]:
```
score = r2_score(reg_pred,y_test)
```

In [32]:
```
score
```

Out[32]: -1.1009516027955146

In [ ]: