In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

from warnings import filterwarnings
filterwarnings(action='ignore')
```

In [5]:
```python
wine = pd.read_csv("winequality-red.csv")
print("Successfully Imported Data!")
wine.head()
```

```
Successfully Imported Data!
```

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25	67	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15	54	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17	60	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5

In [6]:
```python
print(wine.shape)
```

```
(1599, 12)
```

In [7]:
```python
wine.describe(include='all')
```

Out[7]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.875547	46.468418
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460434	32.895920
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000

In [8]:
```python
print(wine.isna().sum())
```

```
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 [9]:
```python
wine.corr()
```

Out[9]:

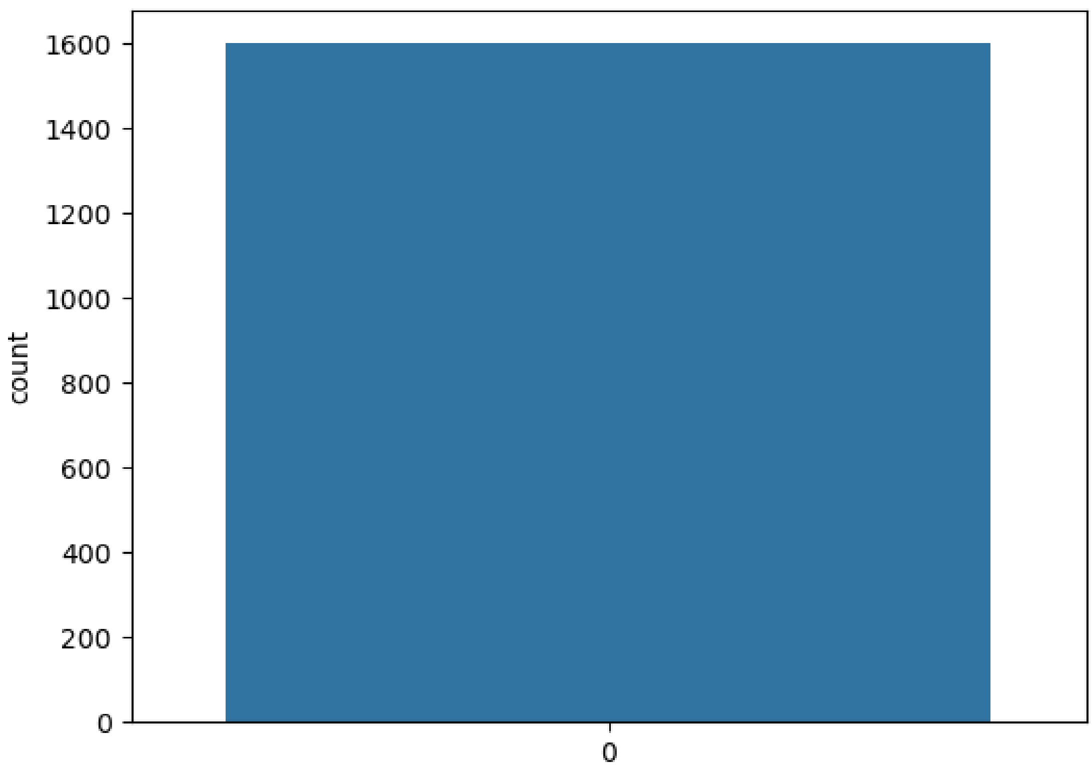
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	p
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153791	-0.113198	0.668047	-0.68297
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010487	0.076479	0.022026	0.23493
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060885	0.035506	0.364947	-0.54190
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187310	0.203048	0.355283	-0.08565
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005627	0.047402	0.200632	-0.26502
free sulfur dioxide	-0.153791	-0.010487	-0.060885	0.187310	0.005627	1.000000	0.668025	-0.021981	0.07028
total sulfur dioxide	-0.113198	0.076479	0.035506	0.203048	0.047402	0.668025	1.000000	0.071256	-0.06650
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021981	0.071256	1.000000	-0.34169
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070288	-0.066507	-0.341699	1.00000
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051606	0.042923	0.148506	-0.19664
alcohol	-0.061668	-0.202288	0.109903	0.042075	-0.221141	-0.069346	-0.205667	-0.496180	0.20563
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050554	-0.185112	-0.174919	-0.05773

In [10]:
```
wine.groupby('quality').mean()
```

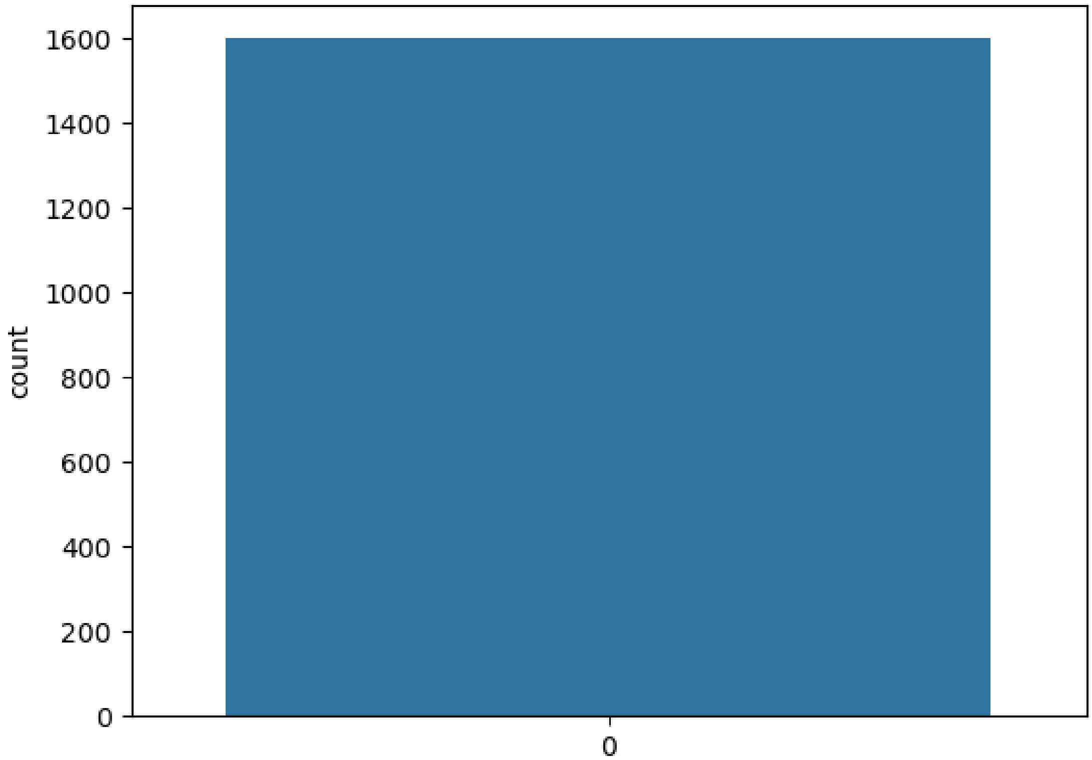
Out[10]:

quality	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sul
3	8.360000	0.884500	0.171000	2.635000	0.122500	11.000000	24.900000	0.997464	3.398000	0.
4	7.779245	0.693962	0.174151	2.694340	0.090679	12.264151	36.245283	0.996542	3.381509	0.
5	8.167254	0.577041	0.243686	2.528855	0.092736	16.983847	56.515419	0.997104	3.304949	0.
6	8.347179	0.497484	0.273824	2.477194	0.084956	15.711599	40.869906	0.996615	3.318072	0.
7	8.872362	0.403920	0.375176	2.720603	0.076588	14.050251	35.020101	0.996104	3.290754	0.
8	8.566667	0.423333	0.391111	2.577778	0.068444	13.277778	33.444444	0.995212	3.267222	0.

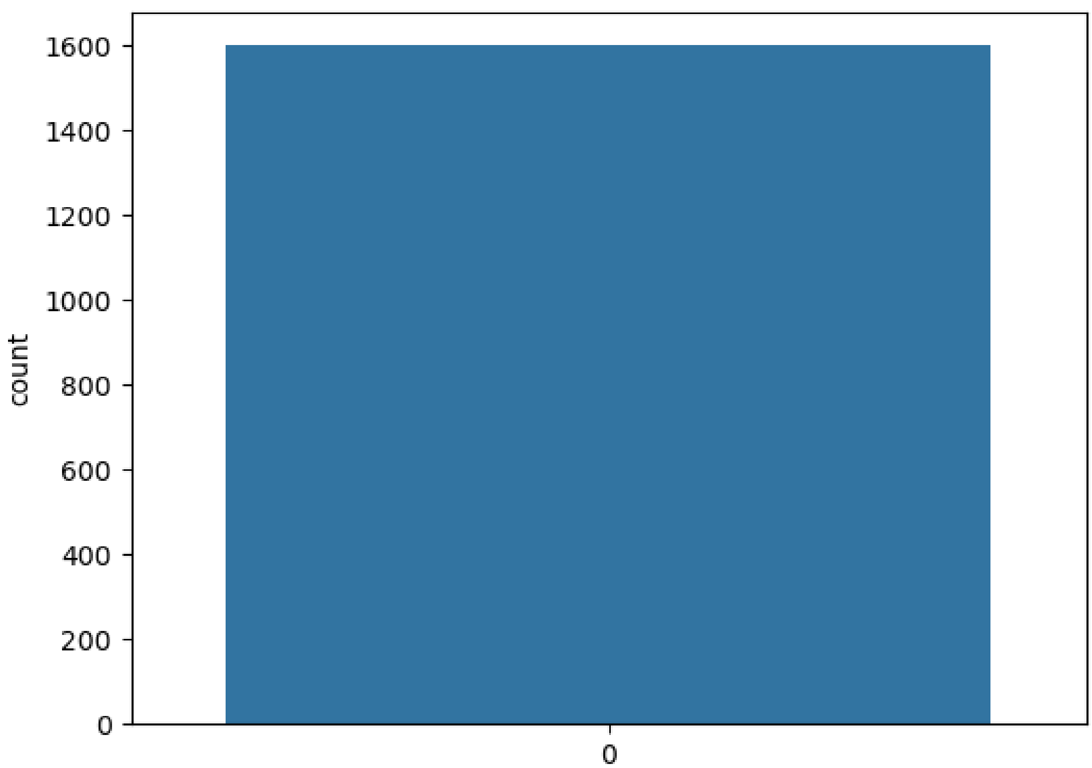
In [13]:
```
sns.countplot(wine['quality'])
plt.show()
```



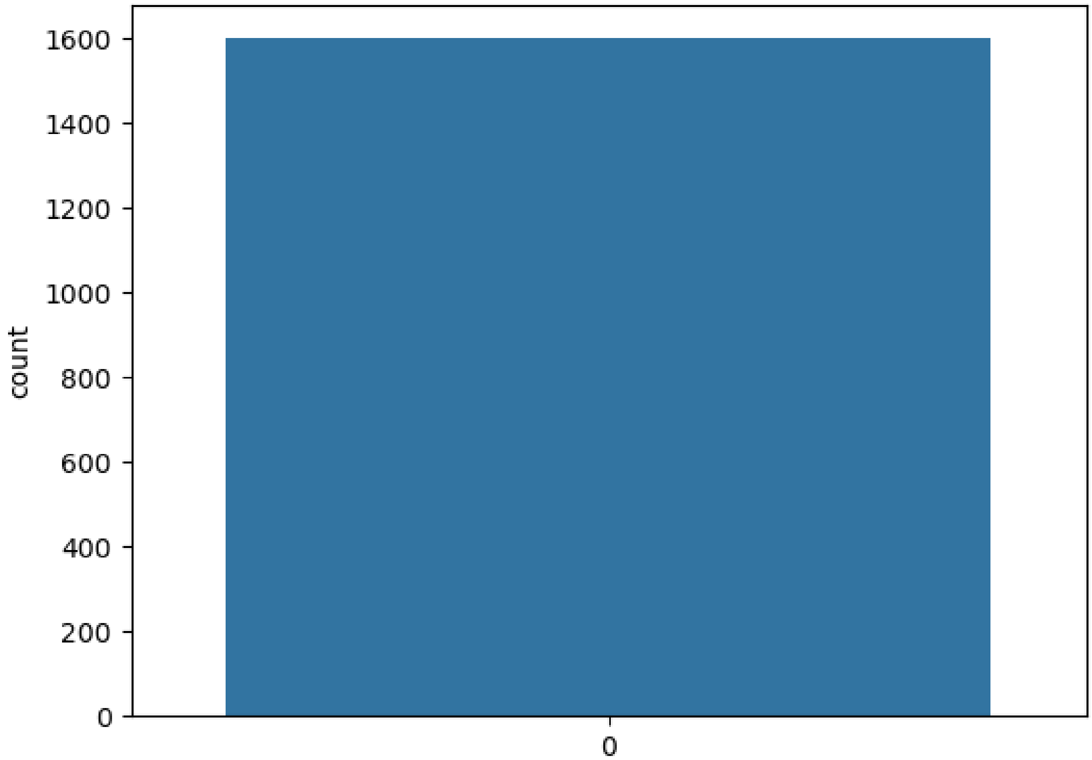
In [12]:
```
sns.countplot(wine['pH'])
plt.show()
```



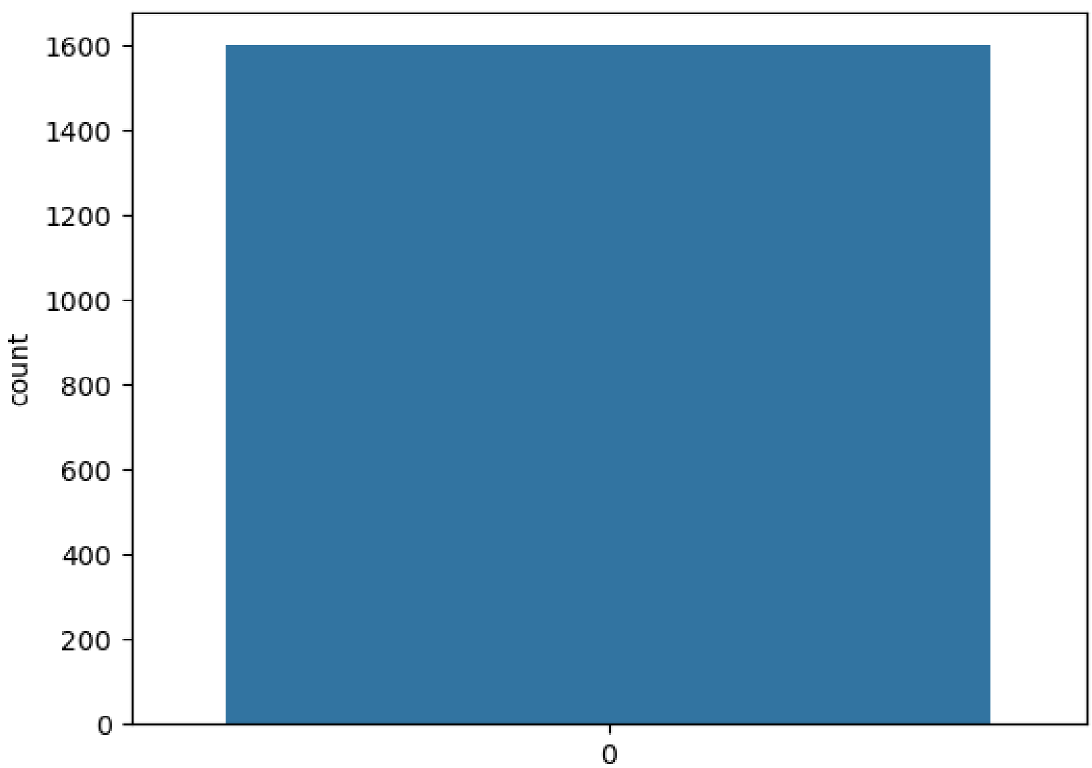
In [14]:
```
sns.countplot(wine['alcohol'])
plt.show()
```



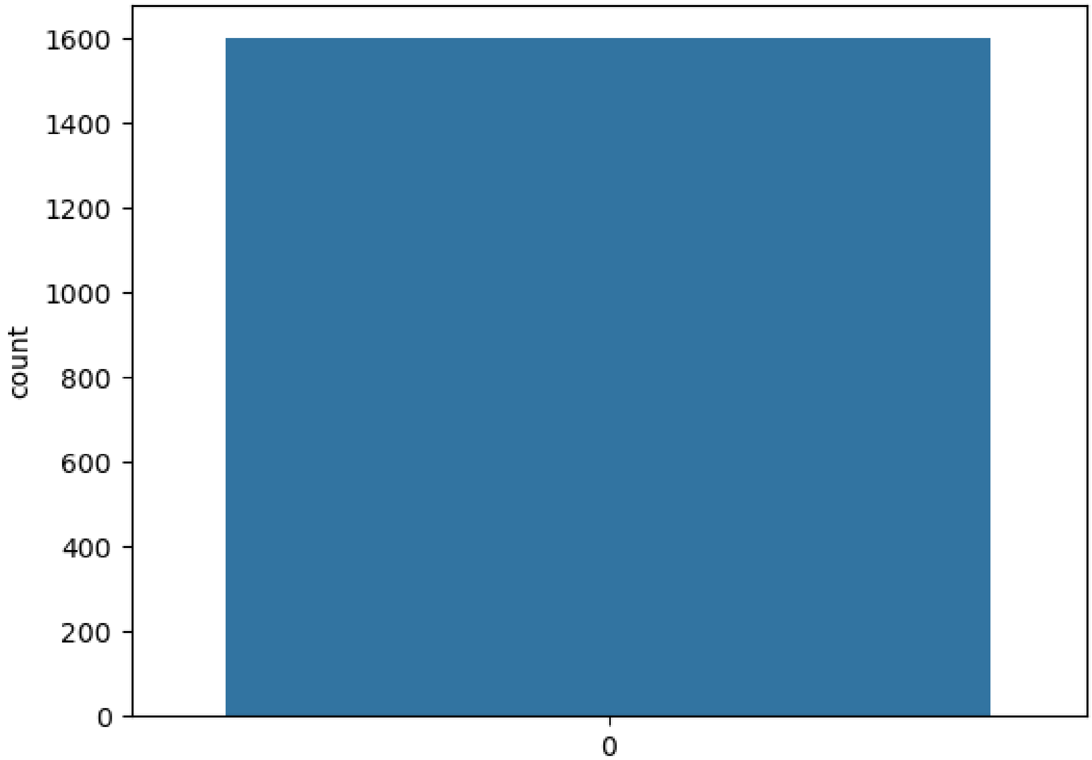
In [15]:
```
sns.countplot(wine['fixed acidity'])
plt.show()
```



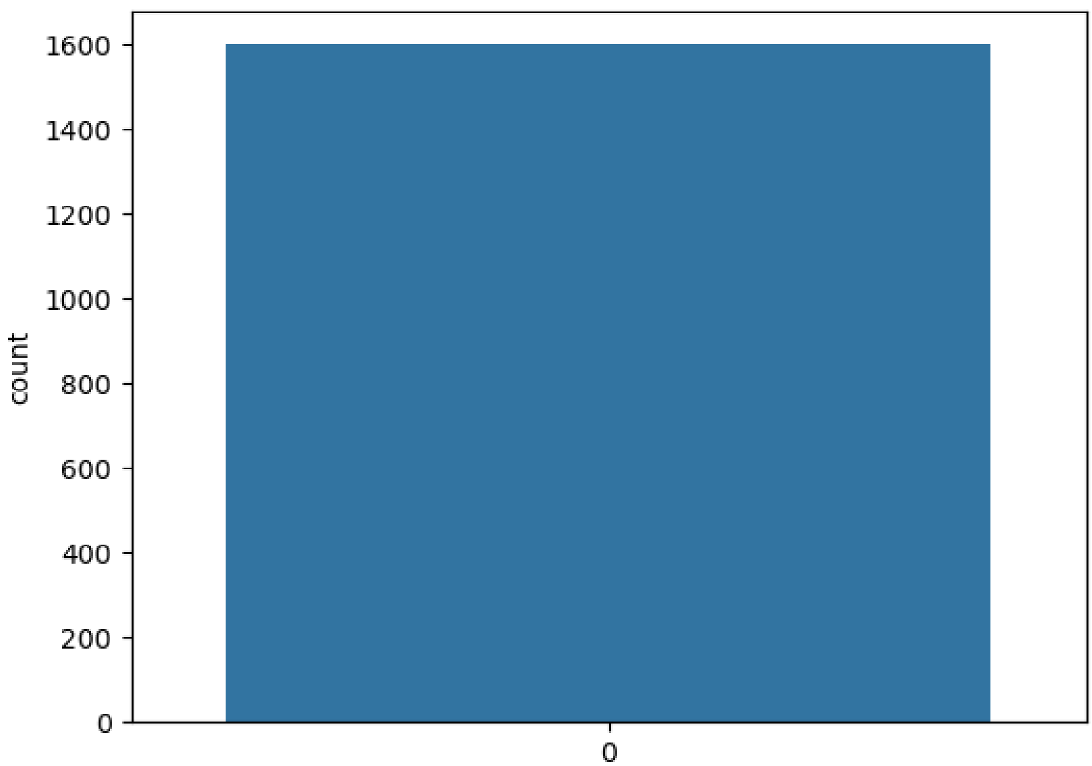
In [16]:
```
sns.countplot(wine['volatile acidity'])
plt.show()
```



In [17]:
```
sns.countplot(wine['citric acid'])
plt.show()
```

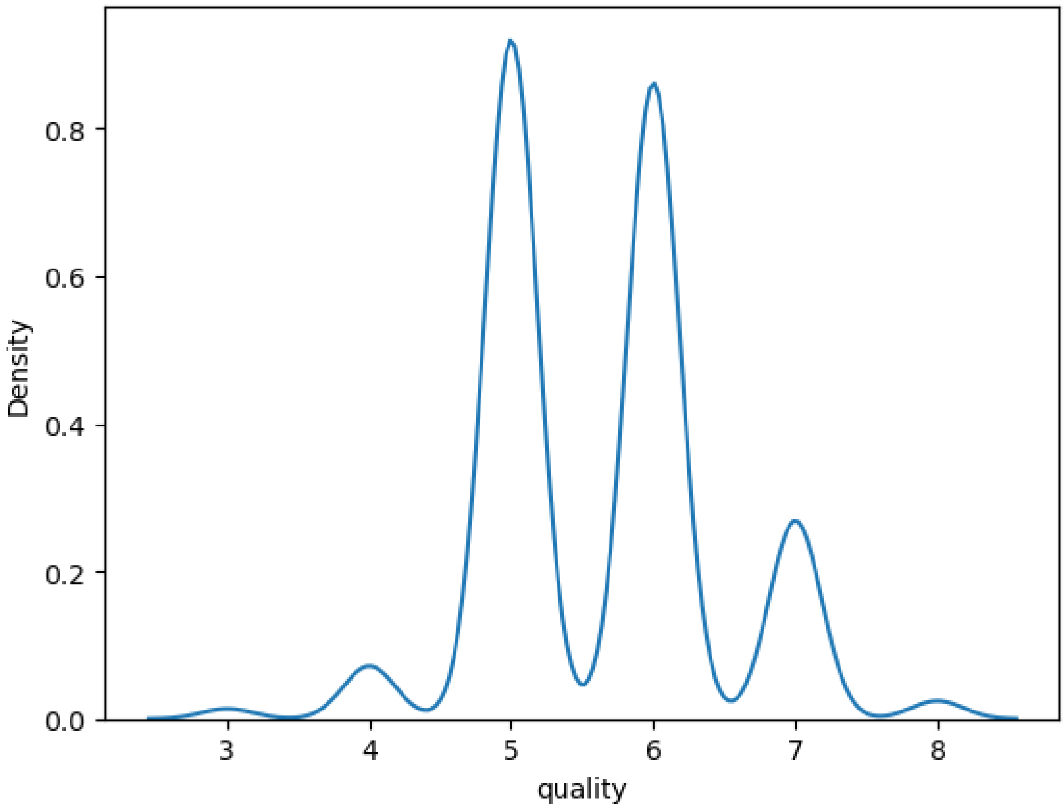


In [18]:
```
sns.countplot(wine['density'])
plt.show()
```



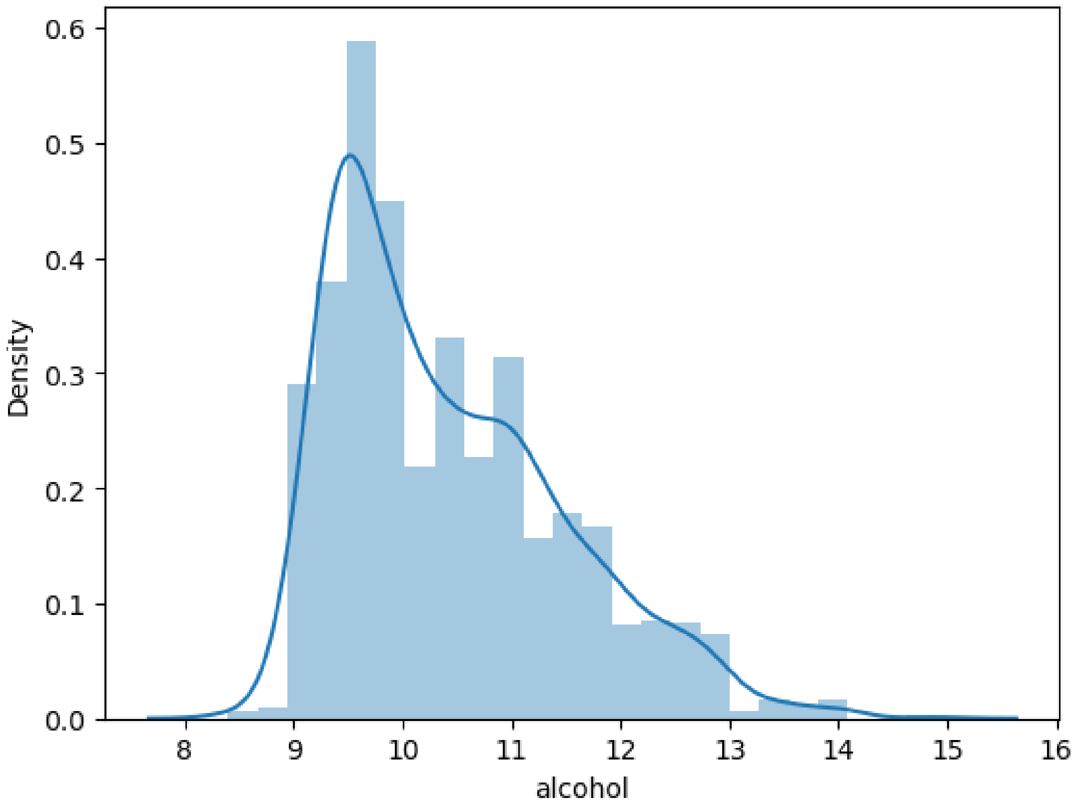
In [19]:
```
sns.kdeplot(wine.query('quality > 2').quality)
```

Out[19]: <Axes: xlabel='quality', ylabel='Density'>



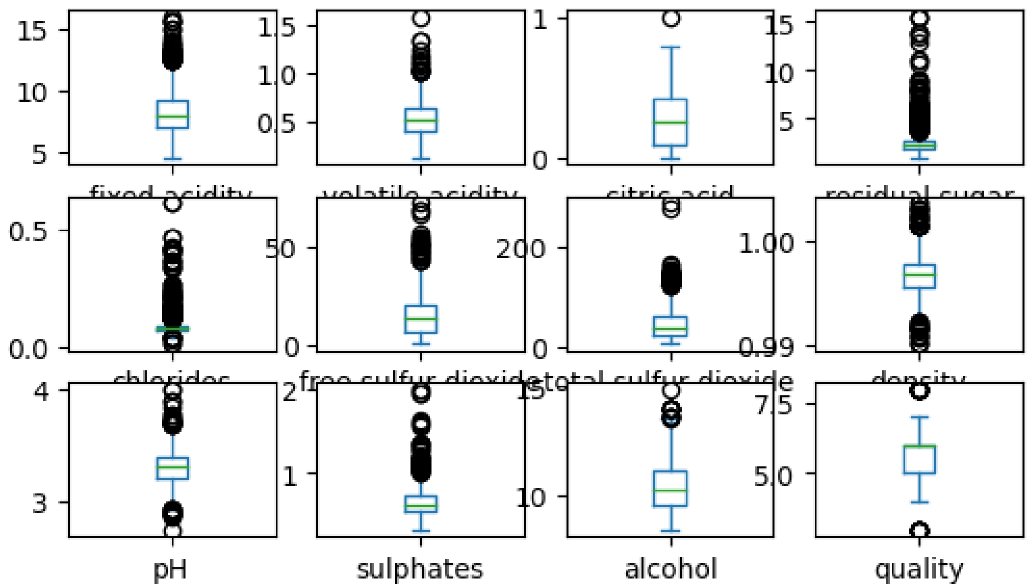
In [20]:
```
sns.distplot(wine['alcohol'])
```

Out[20]:
```
<Axes: xlabel='alcohol', ylabel='Density'>
```



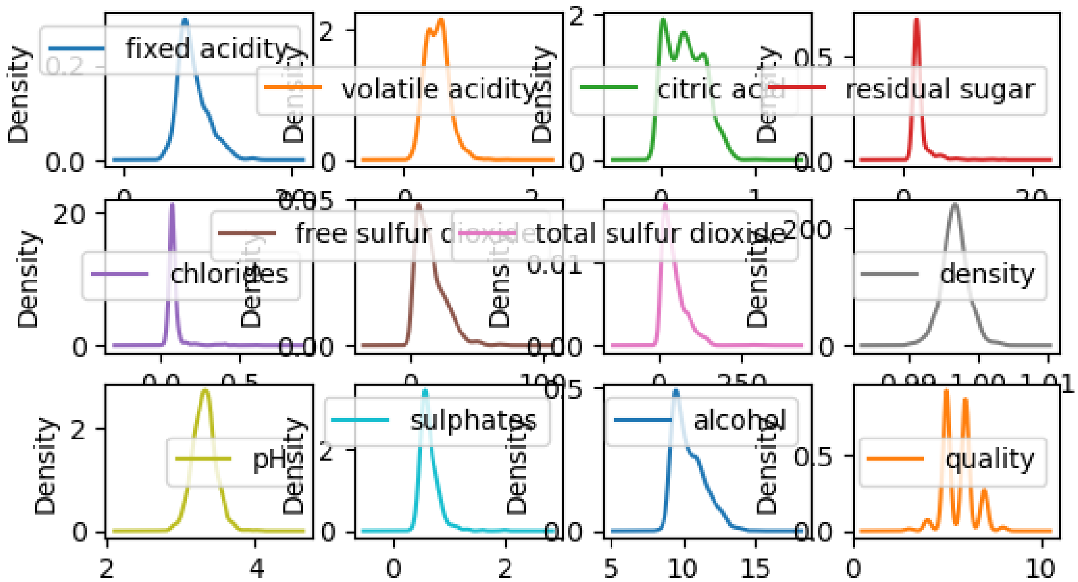
In [21]:
```
wine.plot(kind ='box',subplots = True, layout =(4,4),sharex = False)
```

Out[21]:
```
fixed acidity             Axes(0.125,0.712609;0.168478x0.167391)
volatile acidity       Axes(0.327174,0.712609;0.168478x0.167391)
citric acid            Axes(0.529348,0.712609;0.168478x0.167391)
residual sugar         Axes(0.731522,0.712609;0.168478x0.167391)
chlorides                 Axes(0.125,0.511739;0.168478x0.167391)
free sulfur dioxide    Axes(0.327174,0.511739;0.168478x0.167391)
total sulfur dioxide   Axes(0.529348,0.511739;0.168478x0.167391)
density                Axes(0.731522,0.511739;0.168478x0.167391)
pH                         Axes(0.125,0.31087;0.168478x0.167391)
sulphates               Axes(0.327174,0.31087;0.168478x0.167391)
alcohol                 Axes(0.529348,0.31087;0.168478x0.167391)
quality                 Axes(0.731522,0.31087;0.168478x0.167391)
dtype: object
```

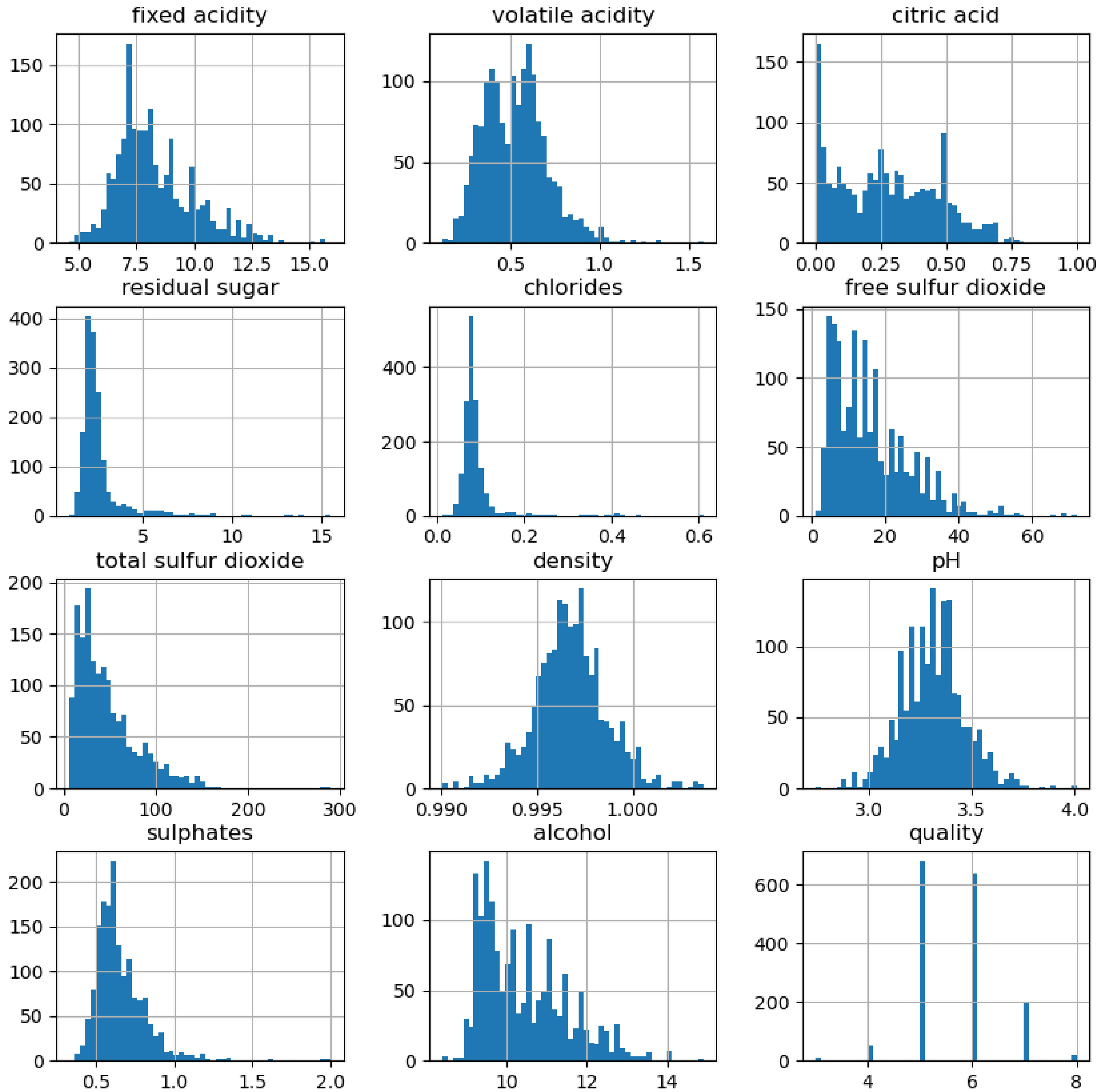


In [22]:
```
wine.plot(kind ='density',subplots = True, layout =(4,4),sharex = False)
```

Out[22]:
```
array([[<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>],
       [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>],
       [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>],
       [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>]], dtype=object)
```

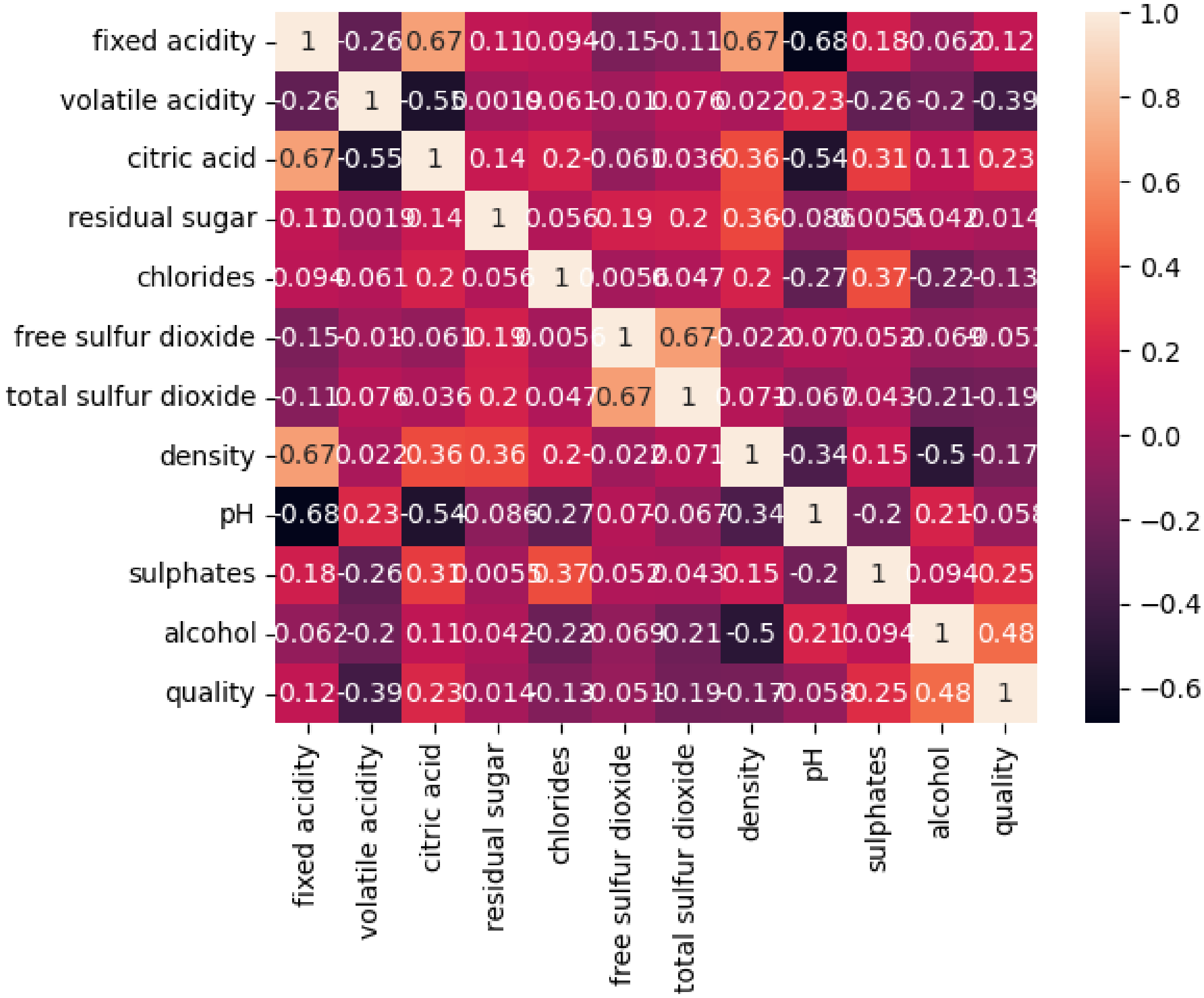


In [23]:
```
wine.hist(figsize=(10,10),bins=50)
plt.show()
```



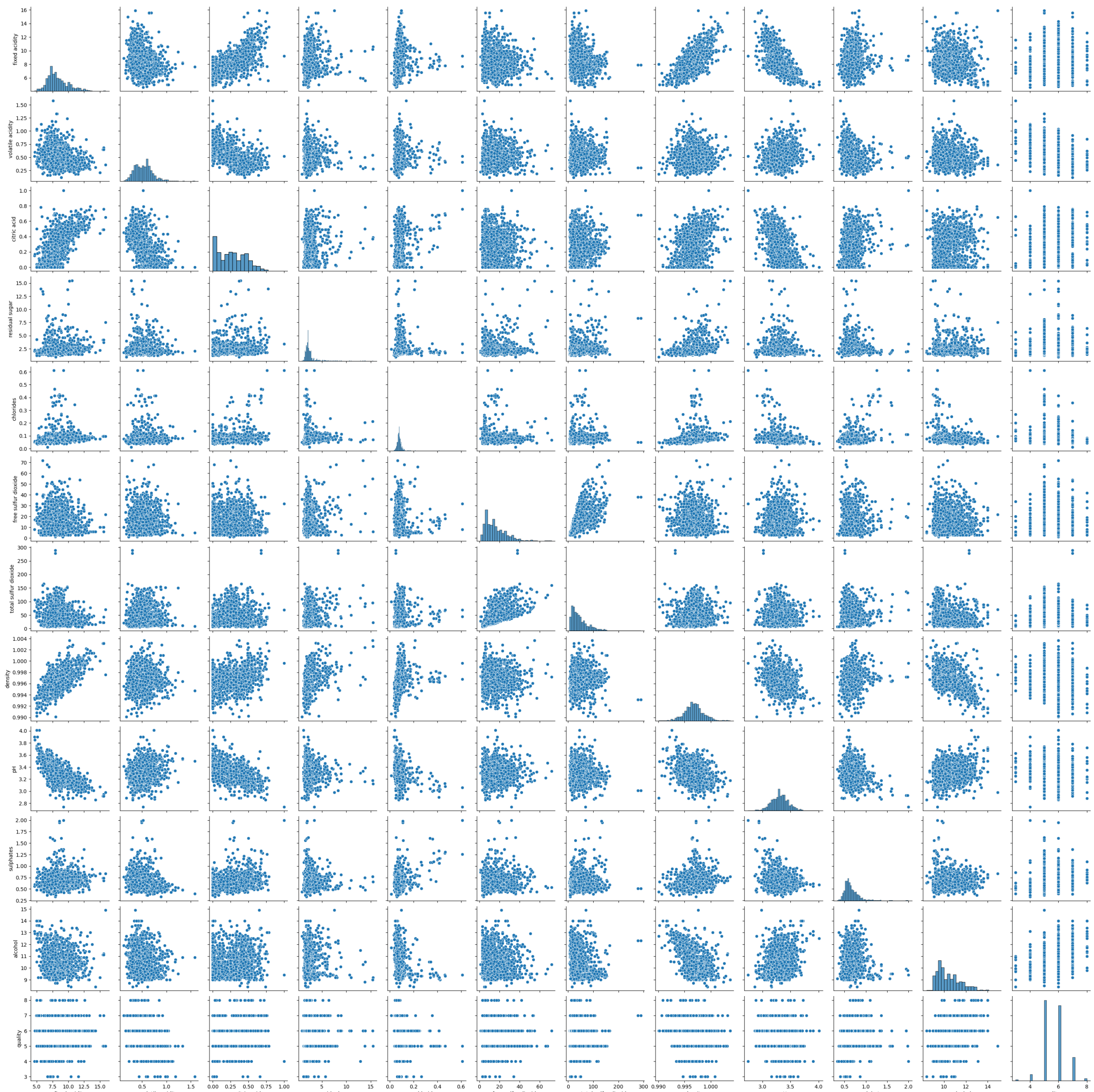
In [24]:
```python
corr = wine.corr()
sns.heatmap(corr,annot=True)
```

Out[24]: <Axes: >



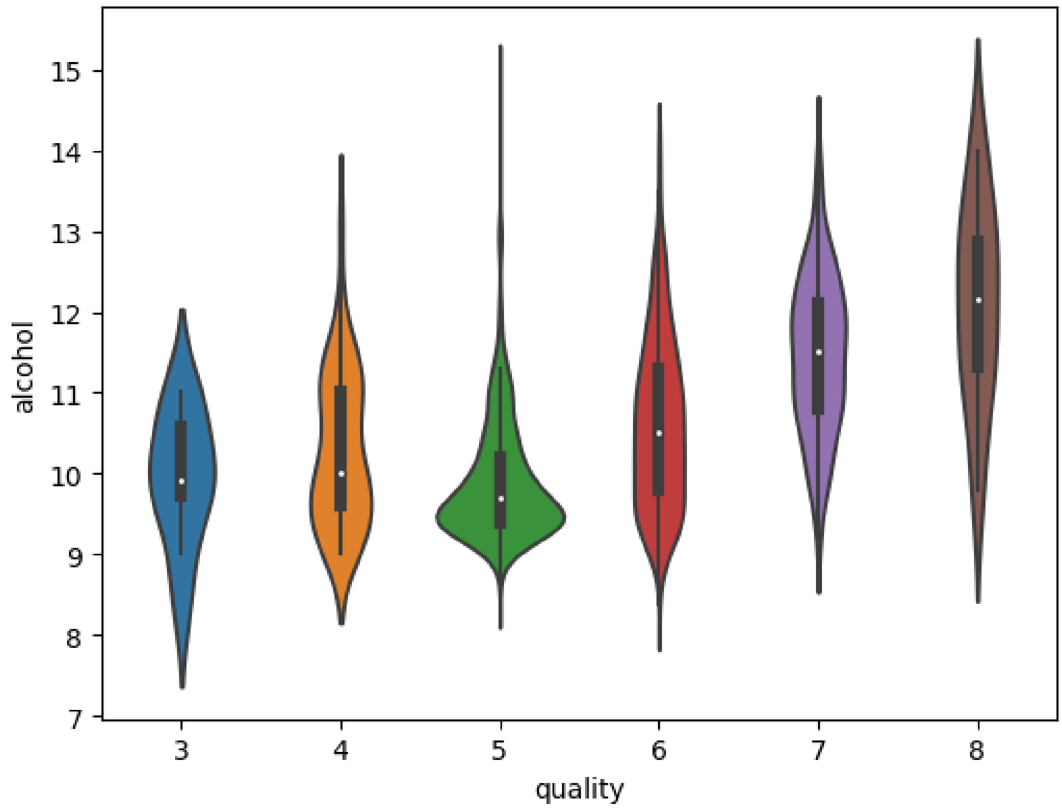
In [25]:
```
sns.pairplot(wine)
```

Out[25]:
```
<seaborn.axisgrid.PairGrid at 0x1b34f1e76a0>
```



In [26]:
```python
sns.violinplot(x='quality', y='alcohol', data=wine)
```

Out[26]:
```
<Axes: xlabel='quality', ylabel='alcohol'>
```



In [27]:
```python
# Create Classification version of target variable
wine['goodquality'] = [1 if x >= 7 else 0 for x in wine['quality']]# Separate featur
X = wine.drop(['quality','goodquality'], axis = 1)
Y = wine['goodquality']
```

In [28]:
```python
# See proportion of good vs bad wines
wine['goodquality'].value_counts()
```

Out[28]:
```
0    1382
1     217
Name: goodquality, dtype: int64
```

In [29]:
```
X
```

Out[29]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	11	34	0.99780	3.51	0.56	9.4
1	7.8	0.880	0.00	2.6	0.098	25	67	0.99680	3.20	0.68	9.8
2	7.8	0.760	0.04	2.3	0.092	15	54	0.99700	3.26	0.65	9.8
3	11.2	0.280	0.56	1.9	0.075	17	60	0.99800	3.16	0.58	9.8
4	7.4	0.700	0.00	1.9	0.076	11	34	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32	44	0.99490	3.45	0.58	10.5
1595	5.9	0.550	0.10	2.2	0.062	39	51	0.99512	3.52	0.76	11.2
1596	6.3	0.510	0.13	2.3	0.076	29	40	0.99574	3.42	0.75	11.0
1597	5.9	0.645	0.12	2.0	0.075	32	44	0.99547	3.57	0.71	10.2
1598	6.0	0.310	0.47	3.6	0.067	18	42	0.99549	3.39	0.66	11.0

1599 rows × 11 columns

In [30]:
```
print(Y)
```

```
0       0
1       0
2       0
3       0
4       0
       ..
1594    0
1595    0
1596    0
1597    0
1598    0
Name: goodquality, Length: 1599, dtype: int64
```

In [31]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()

from sklearn.ensemble import ExtraTreesClassifier
classifiern = ExtraTreesClassifier()
classifiern.fit(X,Y)
score = classifiern.feature_importances_
print(score)
```

```
[0.07661194 0.10278398 0.09552541 0.07436251 0.07255703 0.06720896
 0.08029628 0.08500498 0.06626141 0.11127777 0.16810974]
```

In [34]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3,random_state=7
```

In [35]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,Y_train)
Y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score,confusion_matrix
print("Accuracy Score:",accuracy_score(Y_test,Y_pred))
```

```
Accuracy Score: 0.875
```

In [36]:
```
confusion_mat = confusion_matrix(Y_test,Y_pred)
print(confusion_mat)
```

```
[[401  16]
 [ 44  19]]
```

In [37]:
```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred))
```

```
Accuracy Score: 0.8729166666666667
```

In [38]:
```
from sklearn.svm import SVC
model = SVC()
model.fit(X_train,Y_train)
pred_y = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,pred_y))
```

```
Accuracy Score: 0.86875
```

In [39]:
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion='entropy',random_state=7)
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred))
```

```
Accuracy Score: 0.8645833333333334
```

In [40]:
```
from sklearn.naive_bayes import GaussianNB
model3 = GaussianNB()
model3.fit(X_train,Y_train)
y_pred3 = model3.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred3))
```

```
Accuracy Score: 0.8333333333333334
```

In [41]:
```
from sklearn.ensemble import RandomForestClassifier
model2 = RandomForestClassifier(random_state=1)
model2.fit(X_train, Y_train)
y_pred2 = model2.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred2))
```

```
Accuracy Score: 0.89375
```

In [42]:
```
import xgboost as xgb
model5 = xgb.XGBClassifier(random_state=1)
model5.fit(X_train, Y_train)
y_pred5 = model5.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred5))
```

```
Accuracy Score: 0.89375
```

In [43]:
```
results = pd.DataFrame({
    'Model': ['Logistic Regression','KNN', 'SVC','Decision Tree' ,'GaussianNB','Rand
    'Score': [0.870,0.872,0.868,0.864,0.833,0.893,0.879]})

result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
result_df
```

Out[43]:

	Model
Score	
0.893	Random Forest
0.879	Xgboost
0.872	KNN
0.870	Logistic Regression
0.868	SVC
0.864	Decision Tree
0.833	GaussianNB

```
In [ ]:
```