In [109]:
```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn import metrics                                # FOR MATHS
from sklearn.model_selection import train_test_split      # FOR TESTING AND TRAI
```

OBJECTIVE -- To experiment with different classification methods to see which yields the highest accuracy To determine which features are the most indicative of a good quality wine

IMPORTING DATA

In [110]:
```python
wine=pd.read_csv("/Users/irdam/Downloads/wine_analysis.csv")
wine
```

Out[110]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56
...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.52	0.58
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.52	0.75
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.52	0.71
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.52	0.66

1599 rows × 12 columns

In [111]:
```python
wine.head(10)
```

Out[111]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5
9	7.5	0.50	0.36	6.1	0.071	17.0	NaN	0.9978	3.35	0.80	10.5

In [112]:
```python
wine.tail()
```

Out[112]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.52	0.58
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.52	0.75
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.52	0.71
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.52	0.66

DESCRIBING DATA

In [113]:
```python
wine.describe()
```

Out[113]:

	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	1598.00000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.43304
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.87624
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.00000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.00000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.00000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.00000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.00000

In [114]:
```python
wine.shape
```

Out[114]: (1599, 12)

MISSING VALUES

In [115]:
```python
wine.isna()
```

Out[115]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
1594	False	False	False	False	False	False	False	False	False	False
1595	False	False	False	False	False	False	False	False	False	False
1596	False	False	False	False	False	False	False	False	False	False
1597	False	False	False	False	False	False	False	False	False	False
1598	False	False	False	False	False	False	False	False	False	False

1599 rows × 12 columns

In [116]:
```python
wine.isnull().sum()
```

Out[116]:
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    1
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 1
dtype: int64
```

In [117]:
```python
wine.dropna(inplace=True)
```

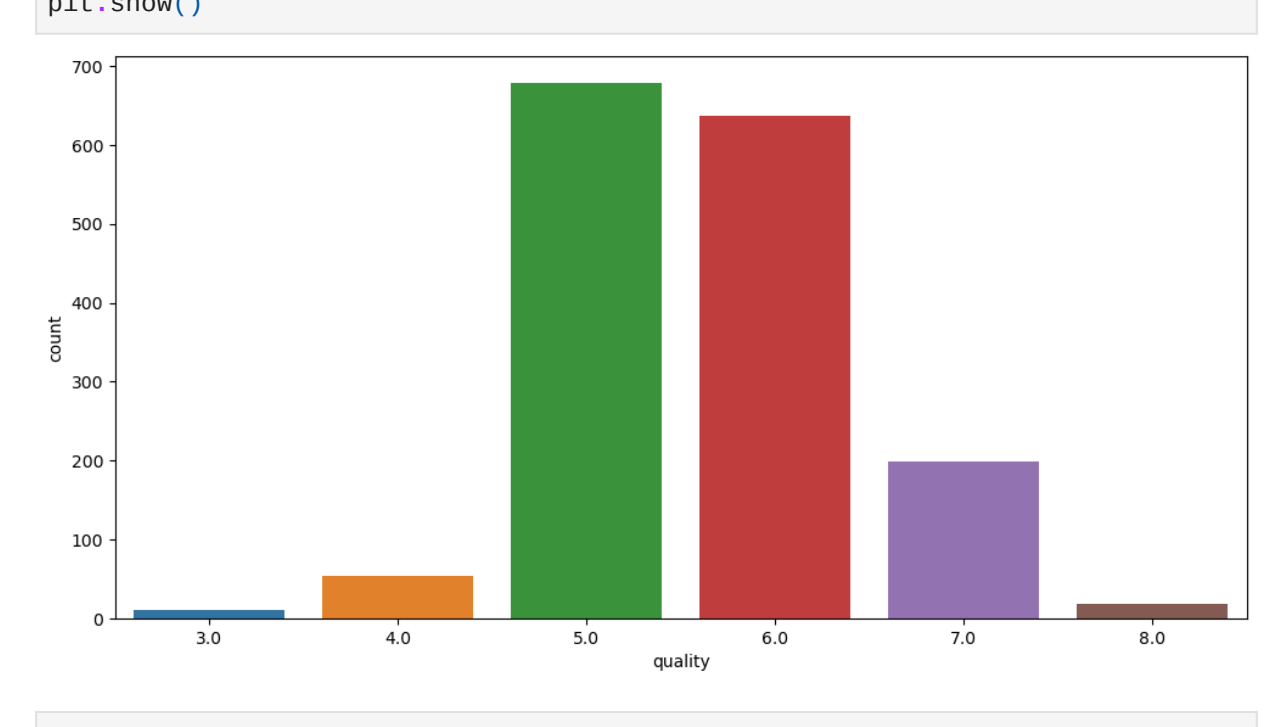
In [118]:
```python
wine['quality'].value_counts()
```

Out[118]:
```
5.0    679
6.0    637
7.0    199
4.0     53
8.0     18
3.0     10
Name: quality, dtype: int64
```

PLOTTING

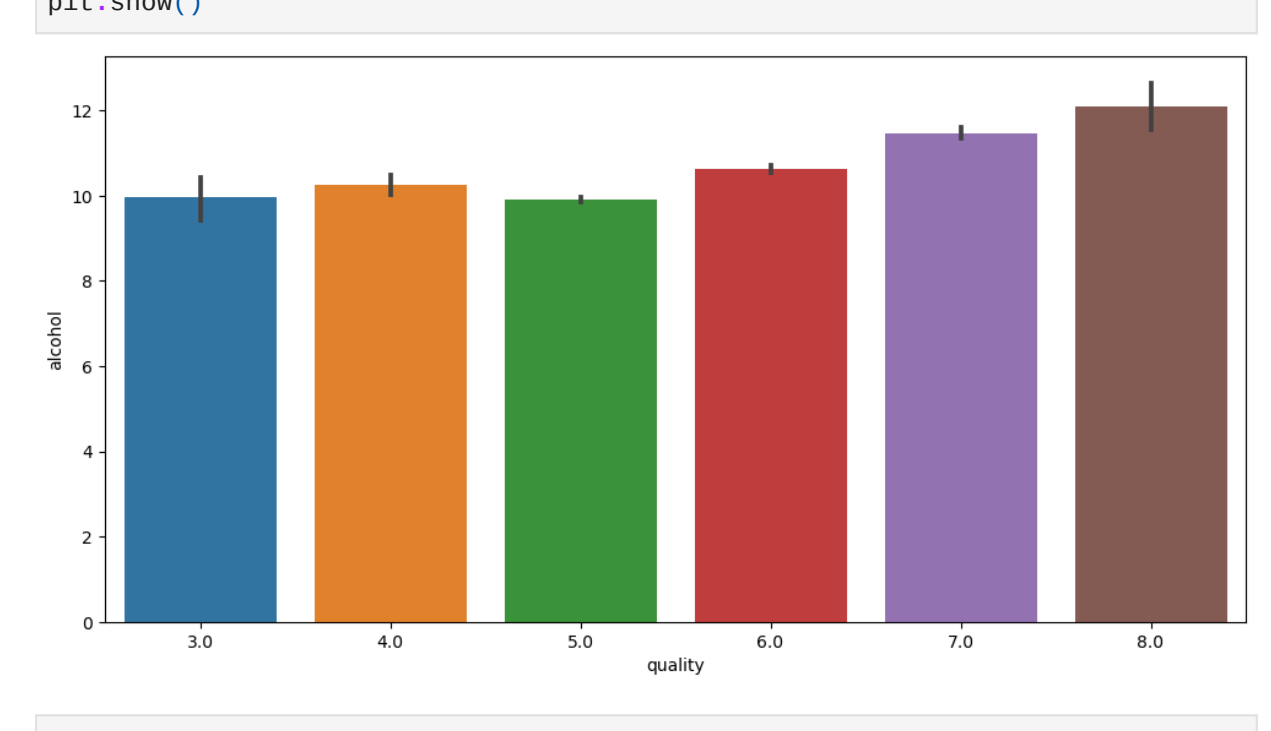
In [119]:
```python
# QUALITY

plt.figure(figsize=(12,6))
sns.countplot(x='quality',data=wine)
plt.show()
```



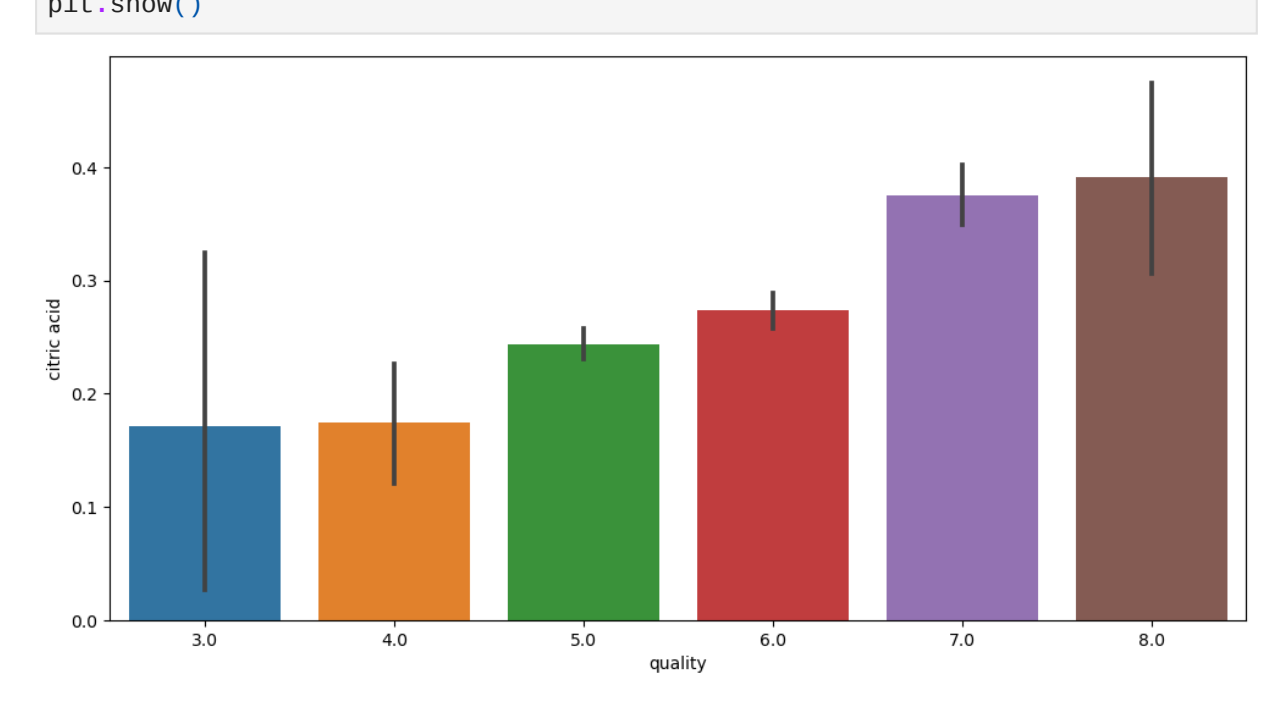
In [120]:
```python
# QUALITY VS ALCOHOL

plt.figure(figsize=(12,6))
sns.barplot(x='quality',y='alcohol',data=wine)
plt.show()
```



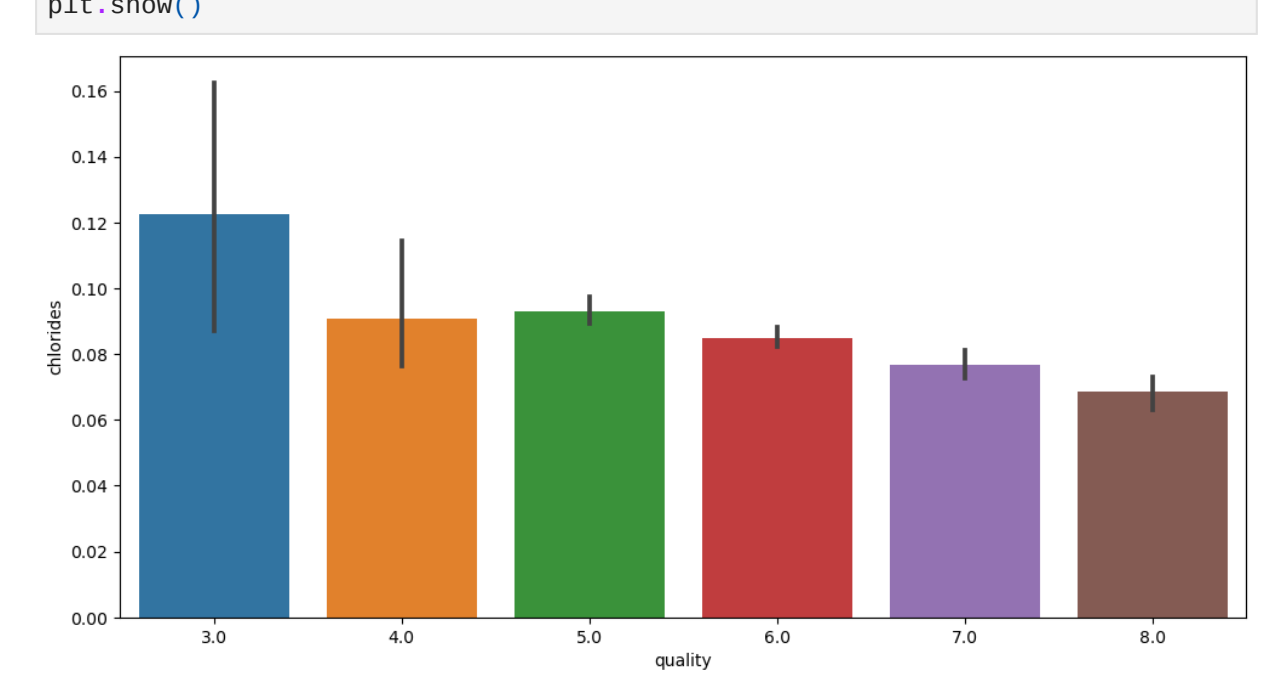
In [121]:
```python
# QUALITY VS CITRIC ACID

plt.figure(figsize=(12,6))
sns.barplot(x='quality',y='citric acid',data=wine)
plt.show()
```



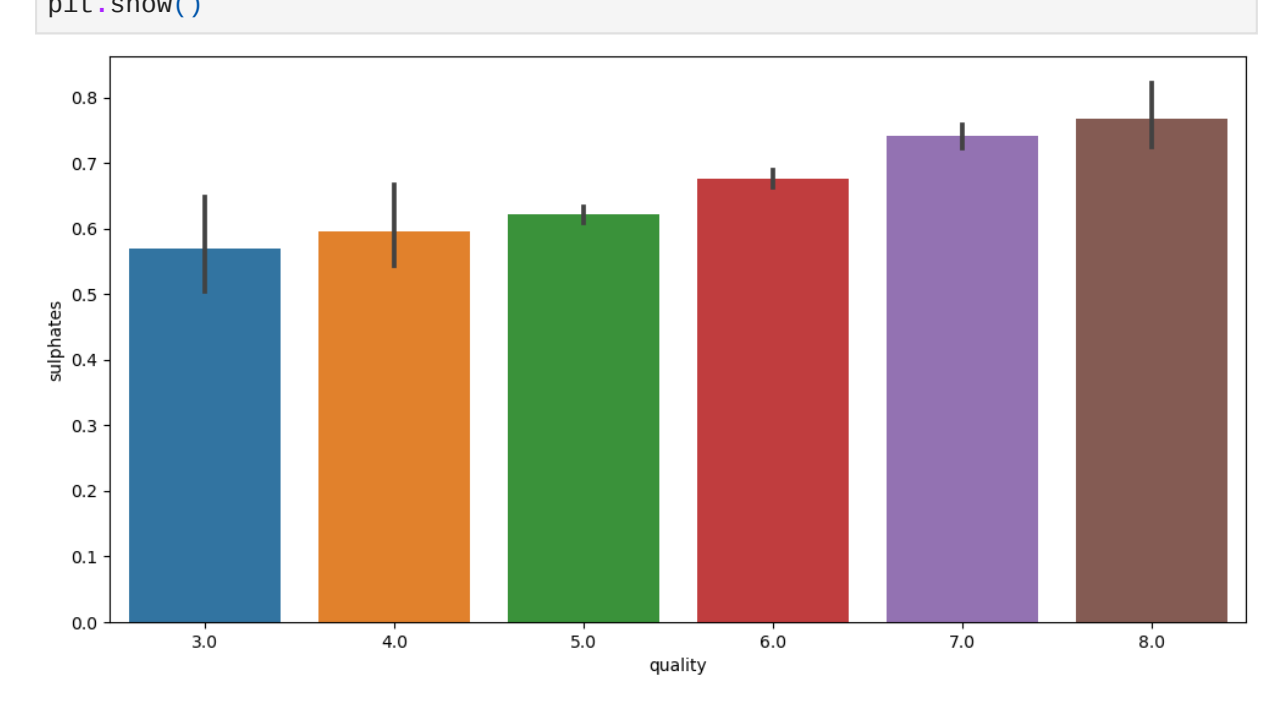
In [122]:
```python
# QUALITY VS CHLORIDES

plt.figure(figsize=(12,6))
sns.barplot(x='quality',y='chlorides',data=wine)
plt.show()
```



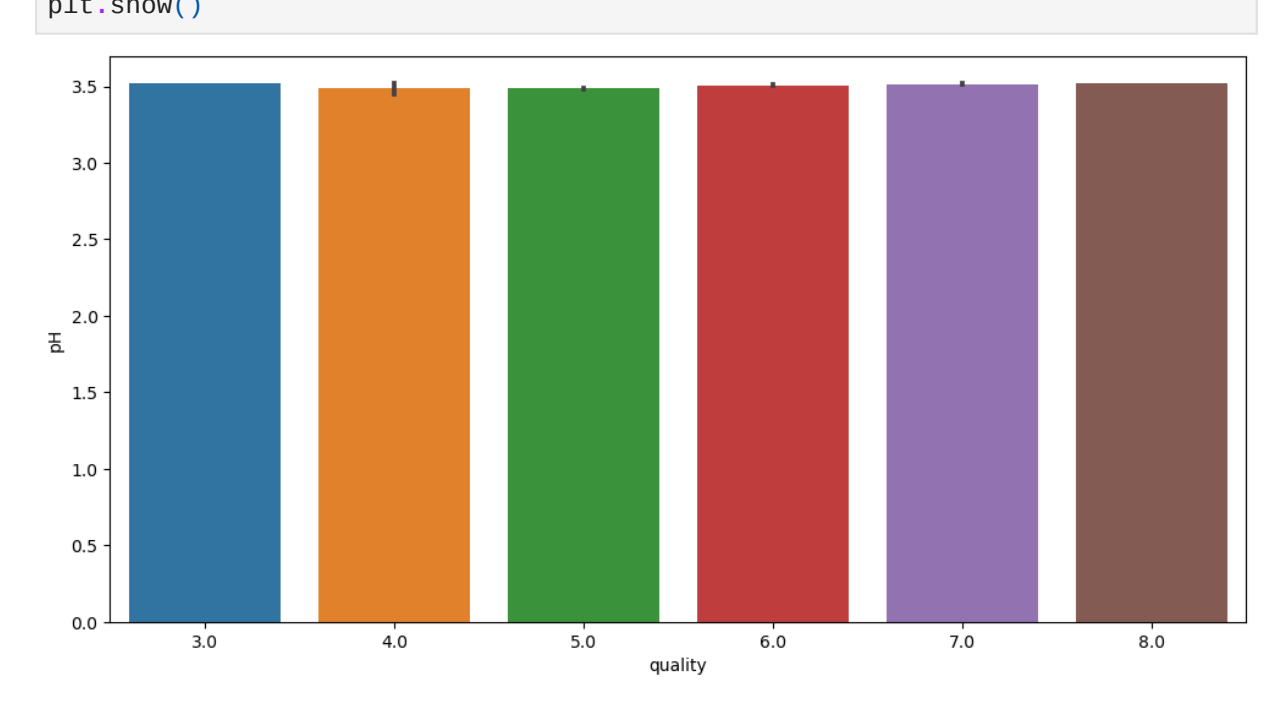
In [123]:
```python
# QUALITY VS SULPHATES

plt.figure(figsize=(12,6))
sns.barplot(x='quality',y='sulphates',data=wine)
plt.show()
```



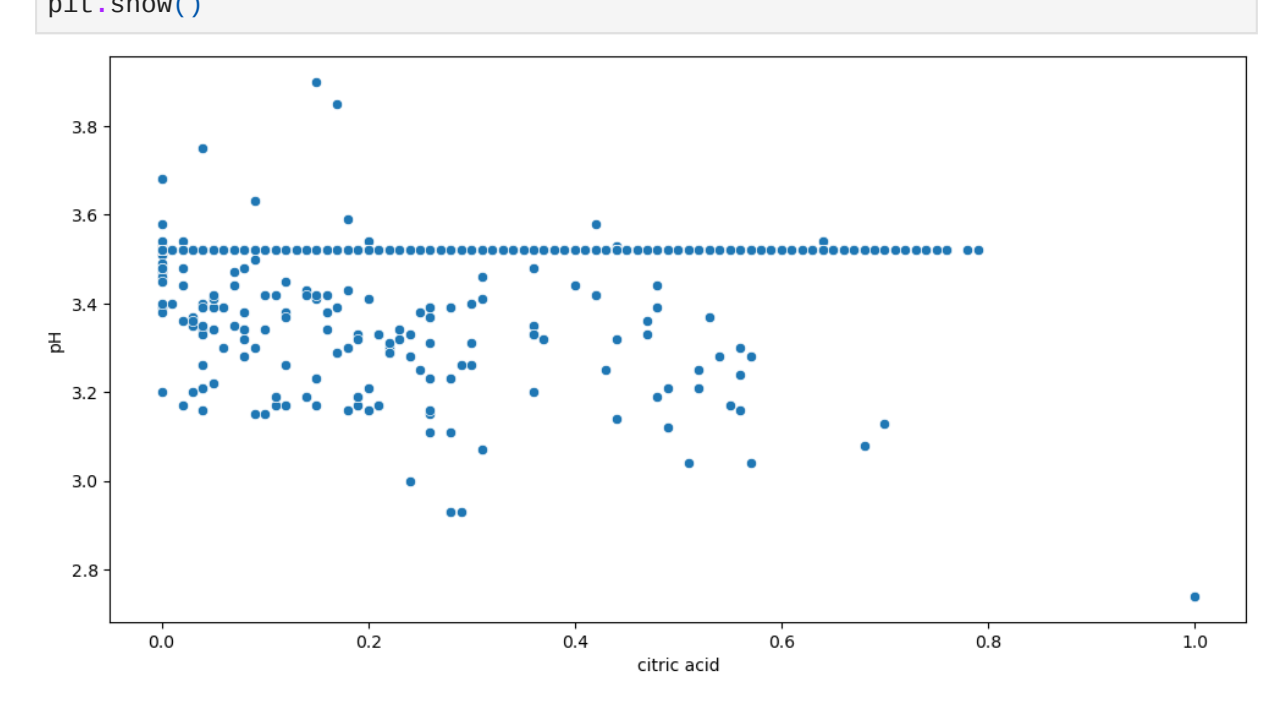
In [124]:
```python
# QUALITY VS pH

plt.figure(figsize=(12,6))
sns.barplot(x='quality',y='pH',data=wine)
plt.show()
```



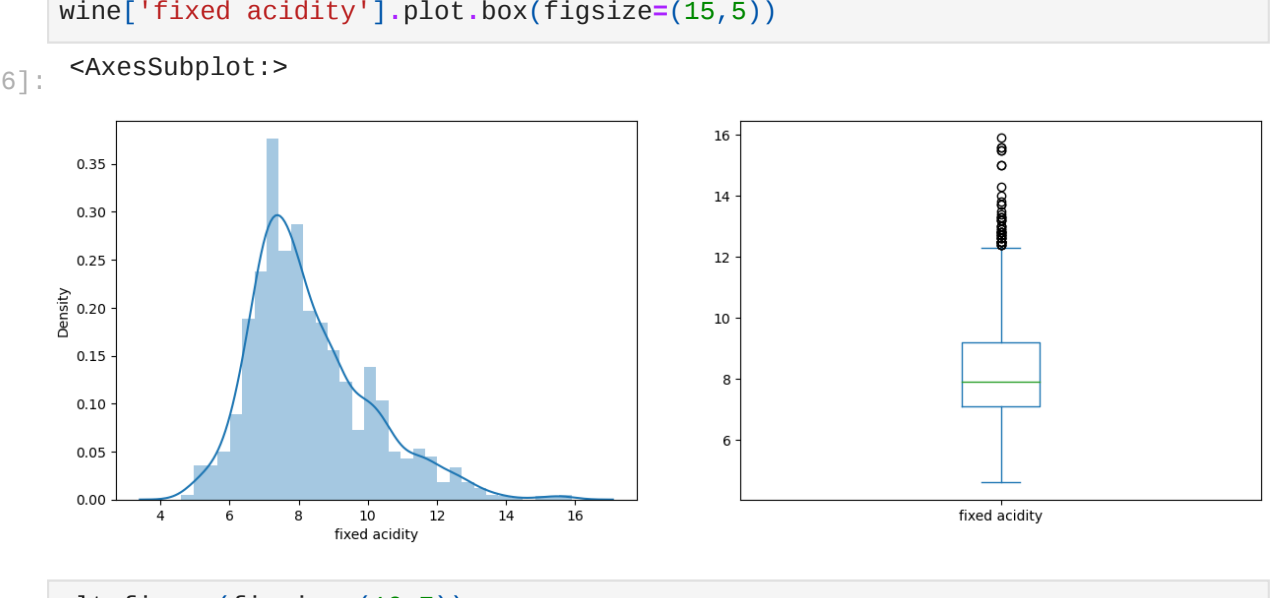
In [125]:
```python
# CITRIC ACID VS pH

plt.figure(figsize=(12,6))
sns.scatterplot(x='citric acid',y='pH',data=wine)
plt.show()
```



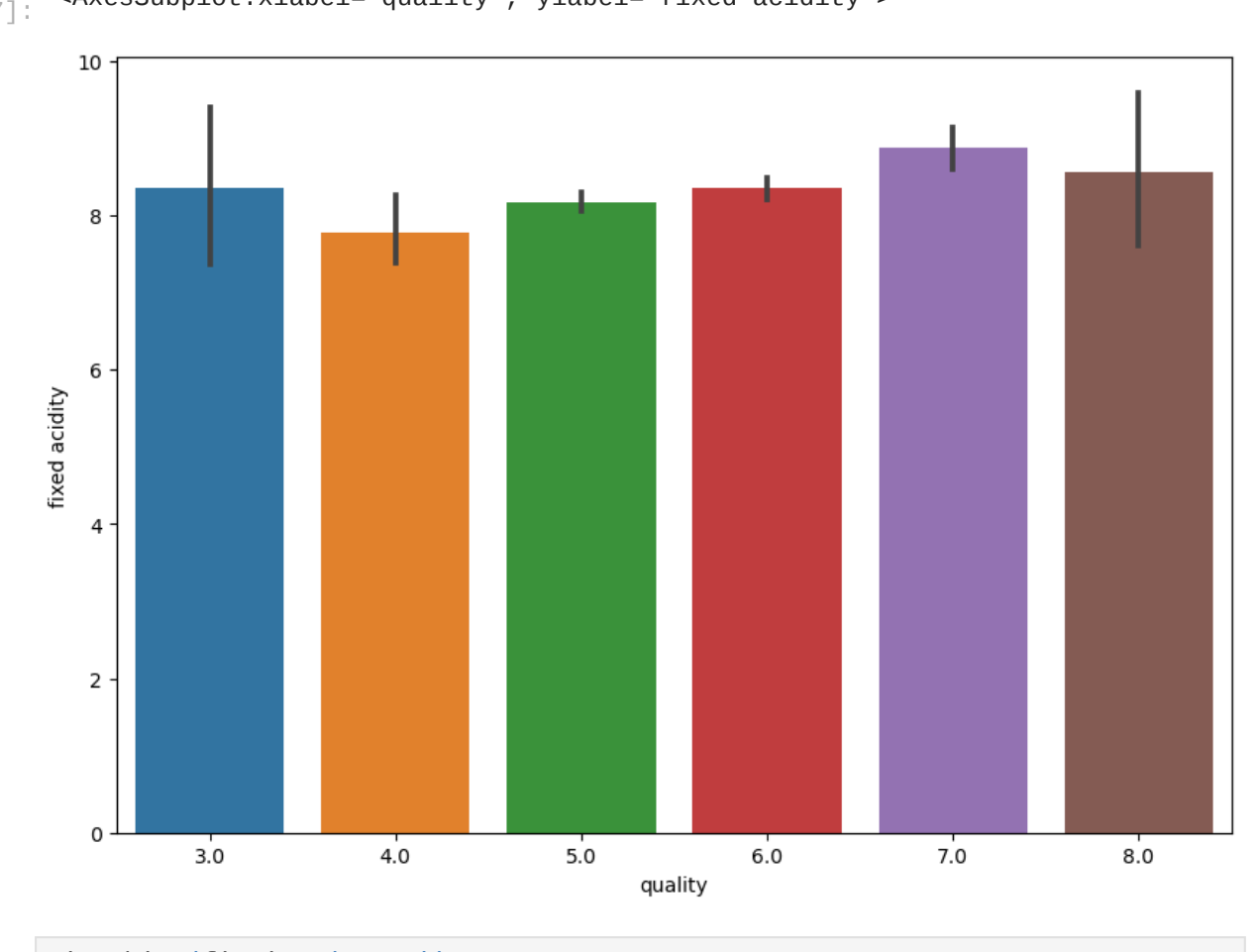
In [126]:
```python
plt.figure(2)
plt.subplot(121)        # 1 row 2 column 1 fig (121)
sns.distplot(wine['fixed acidity'])
plt.subplot(122)
wine['fixed acidity'].plot.box(figsize=(15,5))
```

Out[126]: <AxesSubplot:>

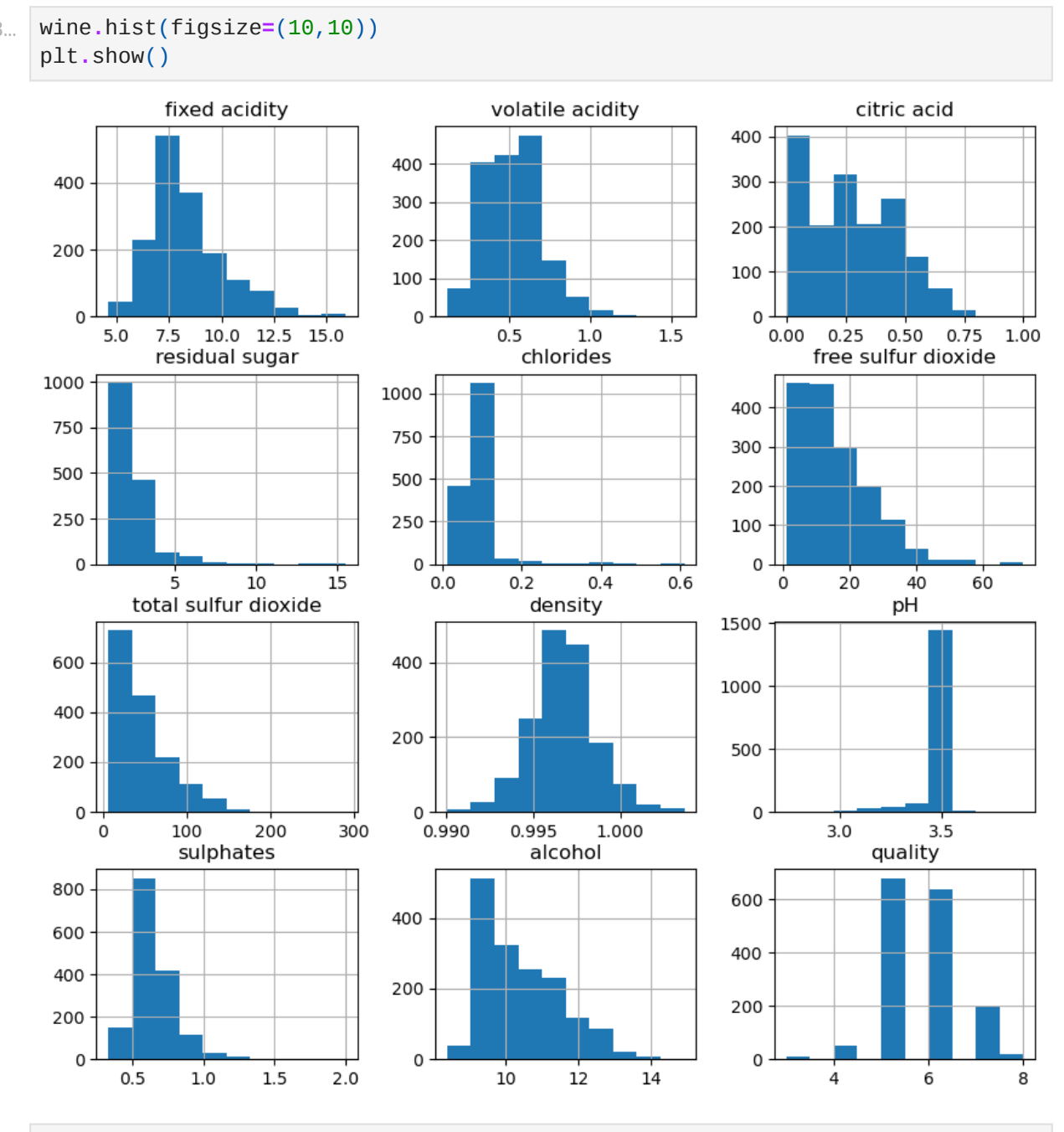


In [127]:
```python
plt.figure(figsize=(10,7))
sns.barplot(x="quality",y="fixed acidity",data=wine)
```

Out[127]: <AxesSubplot:xlabel='quality', ylabel='fixed acidity'>

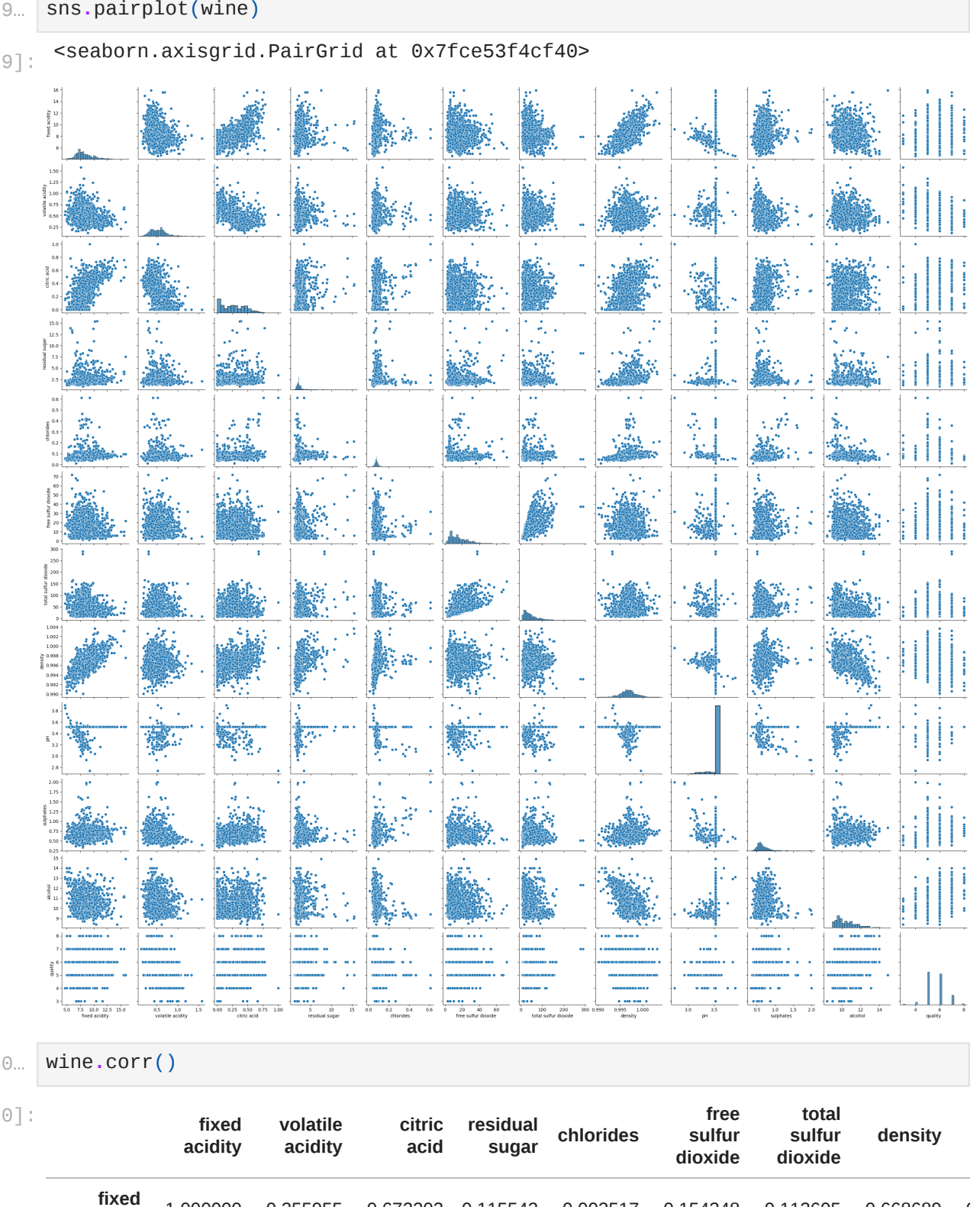


In [128]:
```python
wine.hist(figsize=(10,10))
plt.show()
```



In [129]:
```python
sns.pairplot(wine)
```

Out[129]: <seaborn.axisgrid.PairGrid at 0x7fce53f4cf40>



In [130]:
```python
wine.corr()
```

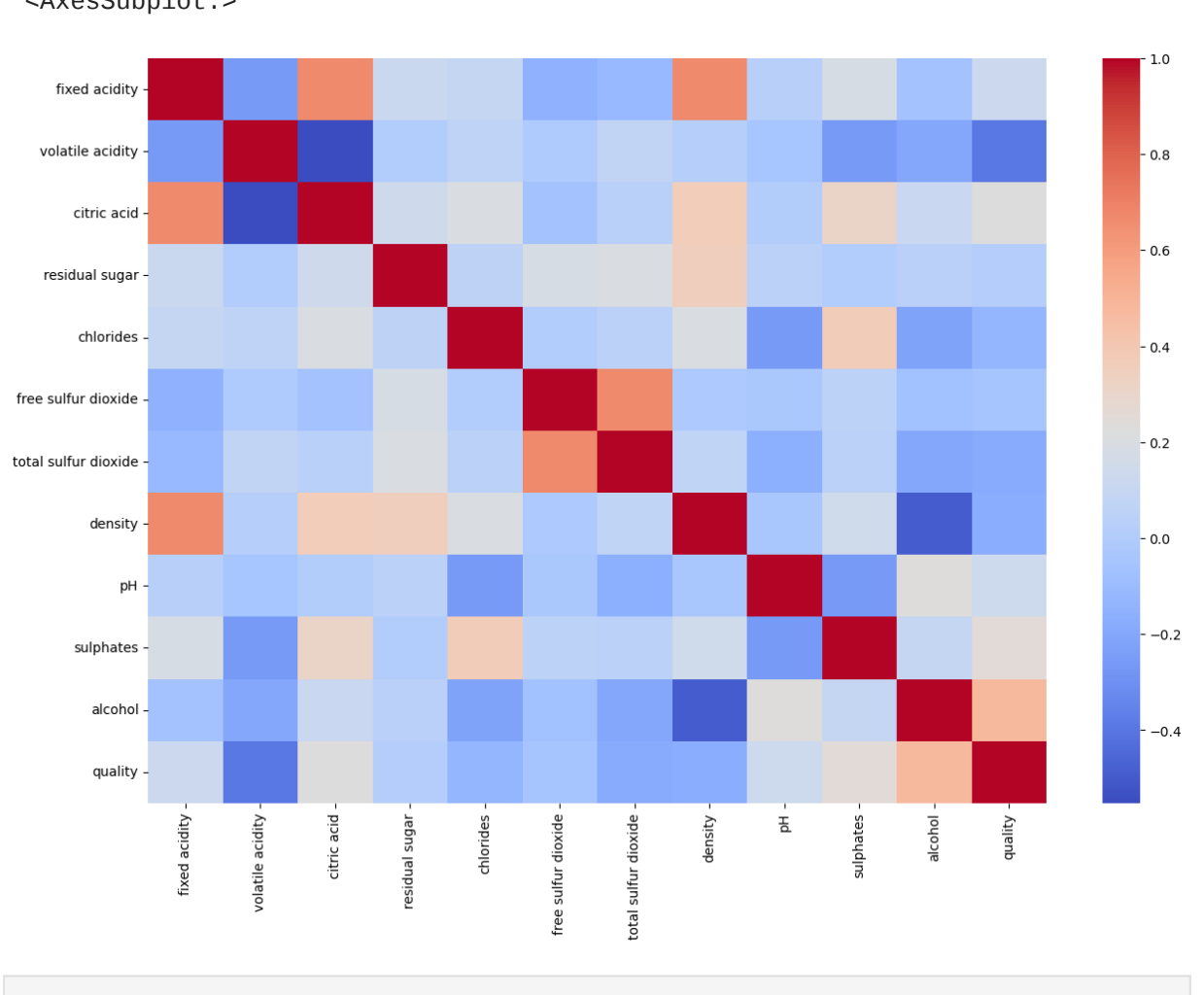
Out[130]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density
fixed acidity	1.000000	-0.255955	0.672203	0.115542	0.093517	-0.154348	-0.112605	0.668689
volatile acidity	-0.255955	1.000000	-0.552080	0.002286	0.061452	-0.009960	0.076832	0.021761
citric acid	0.672203	-0.552080	1.000000	0.143201	0.203904	-0.061621	0.034873	0.365526
residual sugar	0.115542	0.002286	0.143201	1.000000	0.056236	0.187112	0.201120	0.355219
chlorides	0.093517	0.061452	0.203904	0.056236	1.000000	0.005456	0.047837	0.200860
free sulfur dioxide	-0.154348	-0.009960	-0.061621	0.187112	0.005456	1.000000	0.668505	-0.021797
total sulfur dioxide	-0.112605	0.076832	0.034873	0.201120	0.047837	0.668505	1.000000	0.070830
density	0.668689	0.021761	0.365526	0.355219	0.200860	-0.021797	0.070830	1.000000
pH	0.026958	-0.046243	0.008324	0.043099	-0.259456	-0.025512	-0.162984	-0.037144
sulphates	0.183046	-0.260504	0.312146	0.004105	0.371510	0.051200	0.042082	0.148561
alcohol	-0.062419	-0.201591	0.109062	0.041798	-0.221472	-0.070222	-0.205951	-0.496244
quality	0.124090	-0.390616	0.226261	0.015165	-0.129172	-0.050656	-0.184857	-0.174567

In [131]:
```python
# COORELATION GRAPH

plt.figure(figsize=(15,10))
sns.heatmap(wine.corr(),cmap='coolwarm')          #blue is neg corelation and r
```

Out[131]: <AxesSubplot:>



FEATURE SELECTION

In [132]:
```python
wine.sample(5)
```

Out[132]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
459	11.6	0.580	0.66	2.2	0.074	10.0	47.0	1.0008	3.52	0.57
568	9.8	0.500	0.49	2.6	0.250	5.0	20.0	0.9990	3.52	0.79
653	9.4	0.330	0.59	2.8	0.079	9.0	30.0	0.9976	3.52	0.54
275	7.9	0.545	0.06	4.0	0.087	27.0	61.0	0.9965	3.52	0.67
168	6.8	0.630	0.07	2.1	0.089	11.0	44.0	0.9953	3.47	0.55

In [133]:
```python
wine['quality'].unique()
```

Out[133]: array([5., 6., 7., 4., 8., 3.])

In [134]:
```python
# QUALITY OF WINE IS 6 OR ABOVE THEN GOOD ELSE VICEVERSA

wine['goodquality']=[1 if x>=6 else 0 for x in wine['quality']]
wine.sample(5)
```

Out[134]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
1237	7.1	0.750	0.01	2.2	0.059	11.0	18.0	0.99242	3.52	0.40
627	8.8	0.600	0.29	2.2	0.098	5.0	15.0	0.99880	3.52	0.49
283	8.9	0.400	0.32	5.6	0.087	10.0	47.0	0.99910	3.52	0.77
351	9.1	0.795	0.00	2.6	0.096	11.0	26.0	0.99940	3.52	0.63
1144	7.3	0.320	0.23	2.3	0.066	35.0	70.0	0.99588	3.52	0.62

In [135]:
```python
# SEPARATE DEPENDENT AND INDEPENDENT VARIABLES

X = wine.drop(['quality','goodquality'],axis = 1)
Y = wine['goodquality']
```

In [136]:
```python
wine['goodquality'].value_counts()
```

Out[136]:
```
1    854
0    742
Name: goodquality, dtype: int64
```

In [137]:
```python
X
```

Out[137]:

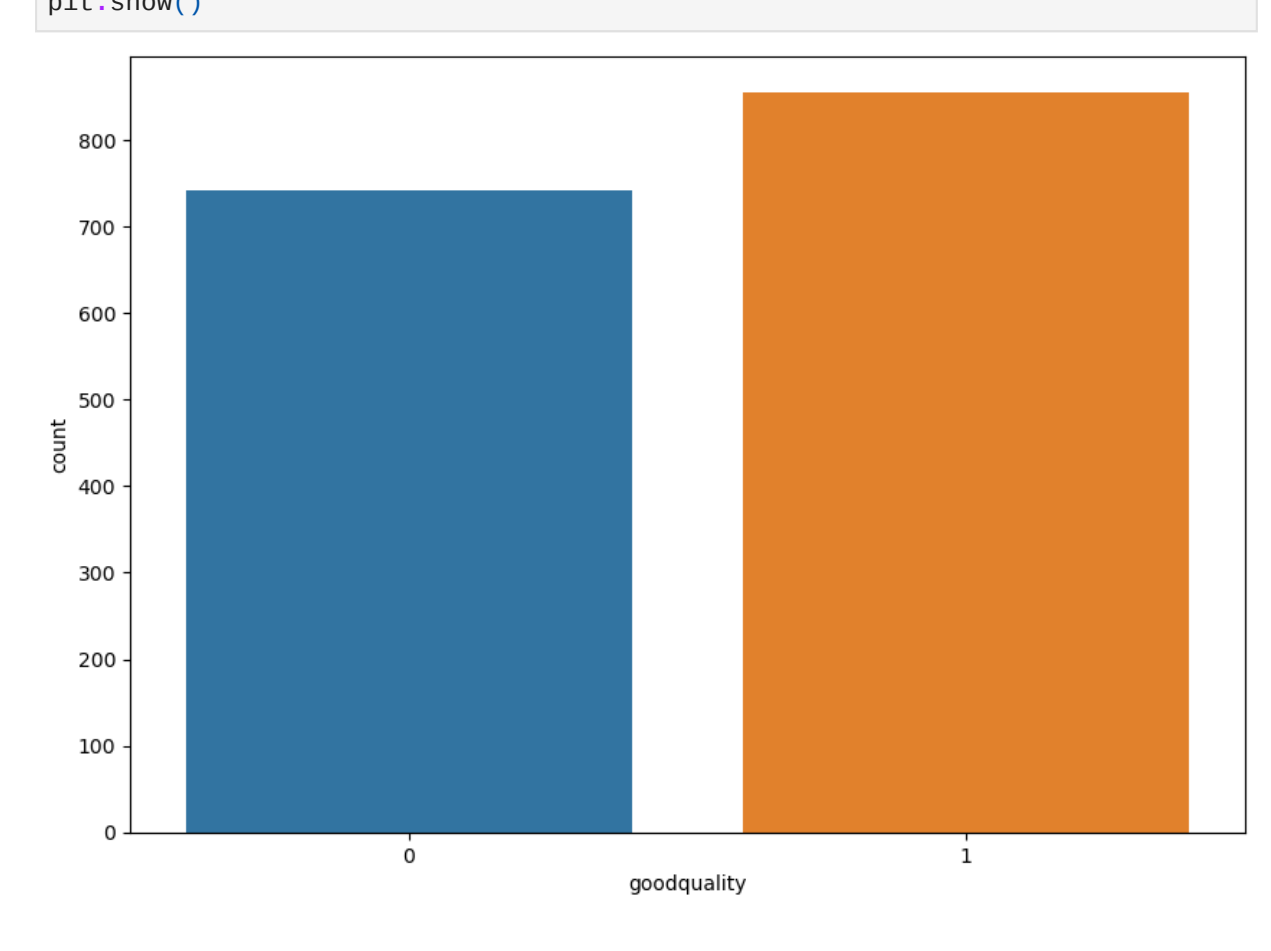
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56
...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.52	0.58
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.52	0.75
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.52	0.71
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.52	0.66

1596 rows × 11 columns

In [138]:
```python
print(Y)
```

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

In [139]:
```python
plt.figure(figsize=(10,7))
sns.countplot(x='goodquality',data=wine)
plt.show()
```



FEATURE IMPORTANCE

In [140]:
```python
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)
```

```
[0.07353085 0.10334186 0.07767972 0.07059736 0.07626066 0.07010207
 0.10232024 0.08101192 0.03139841 0.11779015 0.19596676]
```

TRAIN AND TEST DATA

In [144]:
```python
# Import package for splitting data set
from sklearn.model_selection import train_test_split
```

In [145]:
```python
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_s
```

LOGISTIC REGRESSION

In [143]:
```python
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: 0.7640918580375783

In [146]:
```python
confusion_mat = confusion_matrix(Y_test,Y_pred)
print(confusion_mat)
```

```
[[166  57]
 [ 56 200]]
```

Using KNN:

In [ ]:
```python
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))
```

Using SVC:

In [147]:
```python
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: 0.6325678496868476

Using Decision Tree:

In [148]:
```python
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: 0.7432150313152401

Using GaussianNB:

In [149]:
```python
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: 0.7306889352818372

Using Random Forest:

In [150]:
```python
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: 0.8079331941544885

RESULT

In [152]:
```python
results = pd.DataFrame({
    'Model': ['Logistic Regression','KNN', 'SVC','Decision Tree' ,'GaussianNB
    'Score': [0.870,0.872,0.868,0.864,0.833,0.893]})
result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
result_df
```

Out[152]:

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

In [153]:
```python
# NOW CAN CHOOSE ANY OF THESE MODEL
```