red-wine-quality-prediction

January 16, 2024

[1]:
```
#importing required packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split,GridSearchCV,␣
 ↪cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
```

[2]:
```
#loading dataset
df=pd.read_csv(r'C:\Users\RAVI\Downloads\cognorise\cognorise\red wine␣
 ↪quality\winequality-red.csv')
```

[3]:
```
df.head()
```

[3]:

	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	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58		
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56		

```
3      9.8        6
4      9.4        5
```

[4]:
```
df.shape
```

[4]:
```
(1599, 12)
```

[5]:
```
df.describe()
```

[5]:
```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1599.000000       1599.000000  1599.000000     1599.000000
mean        8.319637          0.527821     0.270976        2.538806
std         1.741096          0.179060     0.194801        1.409928
min         4.600000          0.120000     0.000000        0.900000
25%         7.100000          0.390000     0.090000        1.900000
50%         7.900000          0.520000     0.260000        2.200000
75%         9.200000          0.640000     0.420000        2.600000
max        15.900000          1.580000     1.000000       15.500000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  1599.000000          1599.000000           1599.000000  1599.000000
mean      0.087467            15.874922             46.467792     0.996747
std       0.047065            10.460157             32.895324     0.001887
min       0.012000             1.000000              6.000000     0.990070
25%       0.070000             7.000000             22.000000     0.995600
50%       0.079000            14.000000             38.000000     0.996750
75%       0.090000            21.000000             62.000000     0.997835
max       0.611000            72.000000            289.000000     1.003690

                pH    sulphates      alcohol      quality
count  1599.000000  1599.000000  1599.000000  1599.000000
mean      3.311113     0.658149    10.422983     5.636023
std       0.154386     0.169507     1.065668     0.807569
min       2.740000     0.330000     8.400000     3.000000
25%       3.210000     0.550000     9.500000     5.000000
50%       3.310000     0.620000    10.200000     6.000000
75%       3.400000     0.730000    11.100000     6.000000
max       4.010000     2.000000    14.900000     8.000000
```

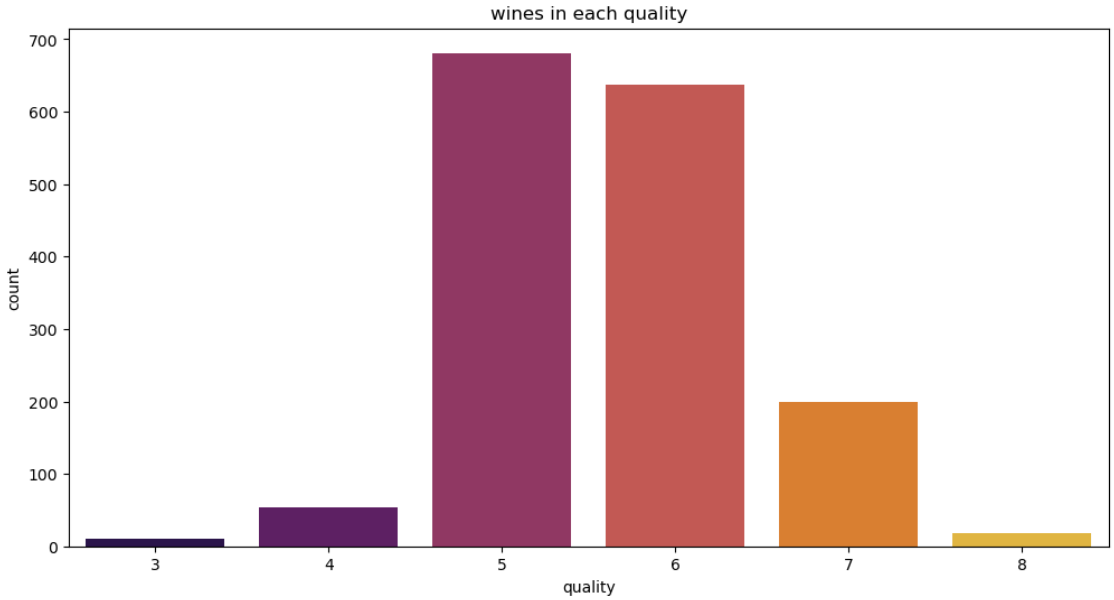
[6]:
```
df.info()
```

```
<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
```

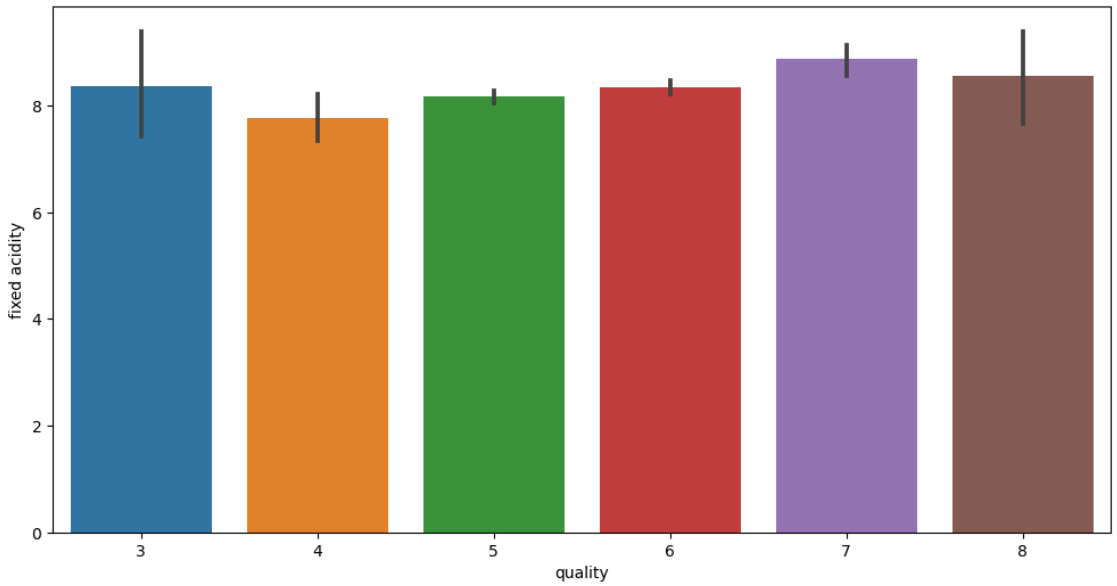
```
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   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
```

[7]:
```
plt.figure(figsize=(12,6))
sns.countplot(x='quality',data=df,palette='inferno')
plt.title('wines in each quality')
```

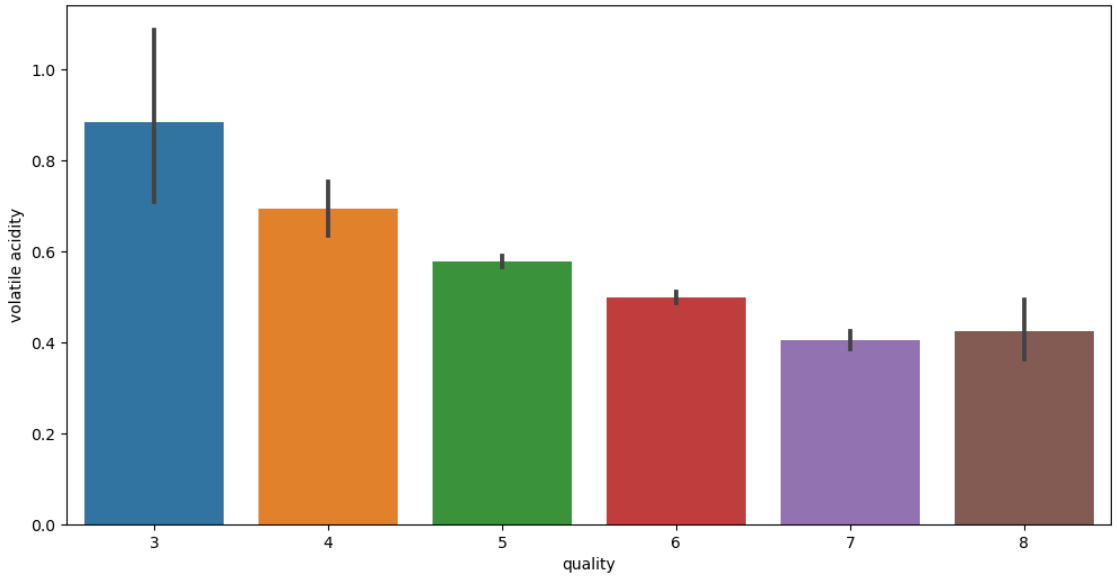
[7]:
```
Text(0.5, 1.0, 'wines in each quality')
```



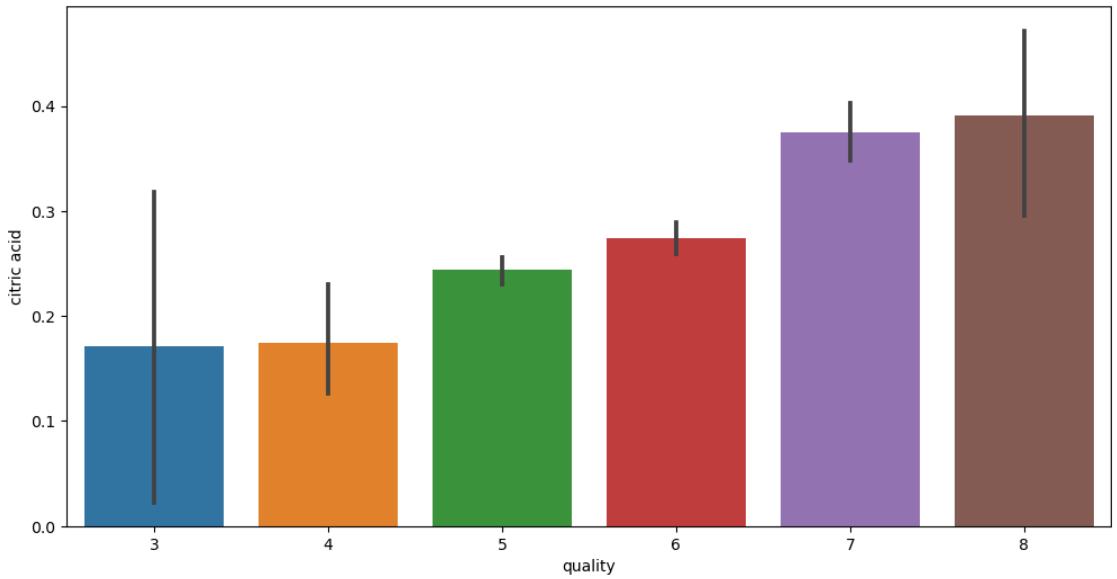
[8]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='quality',y='fixed acidity',data=df)
plt.show()
```



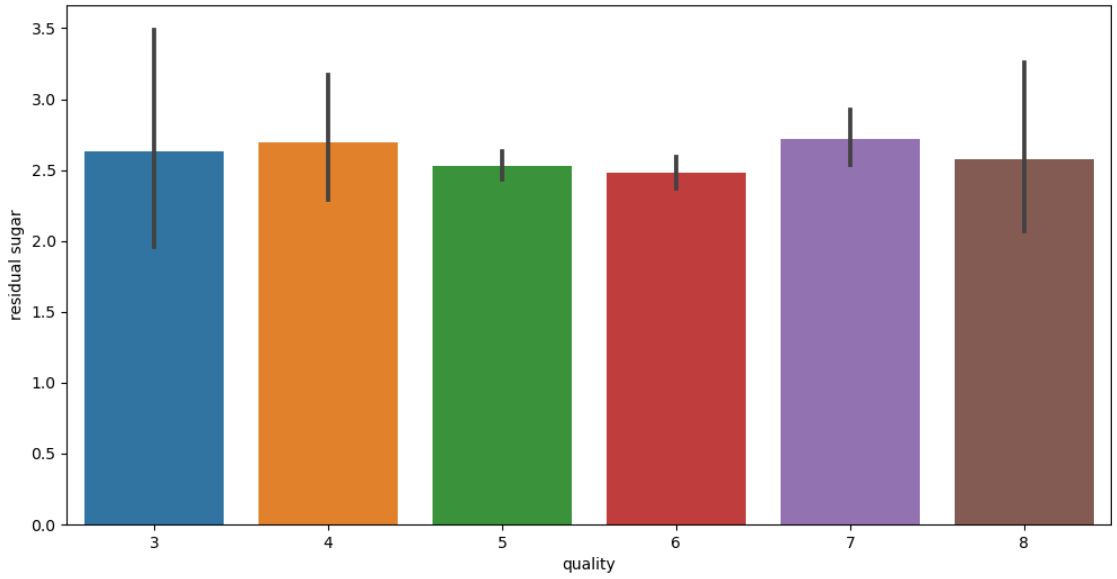
[9]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='quality',y='volatile acidity',data=df)
plt.show()
```



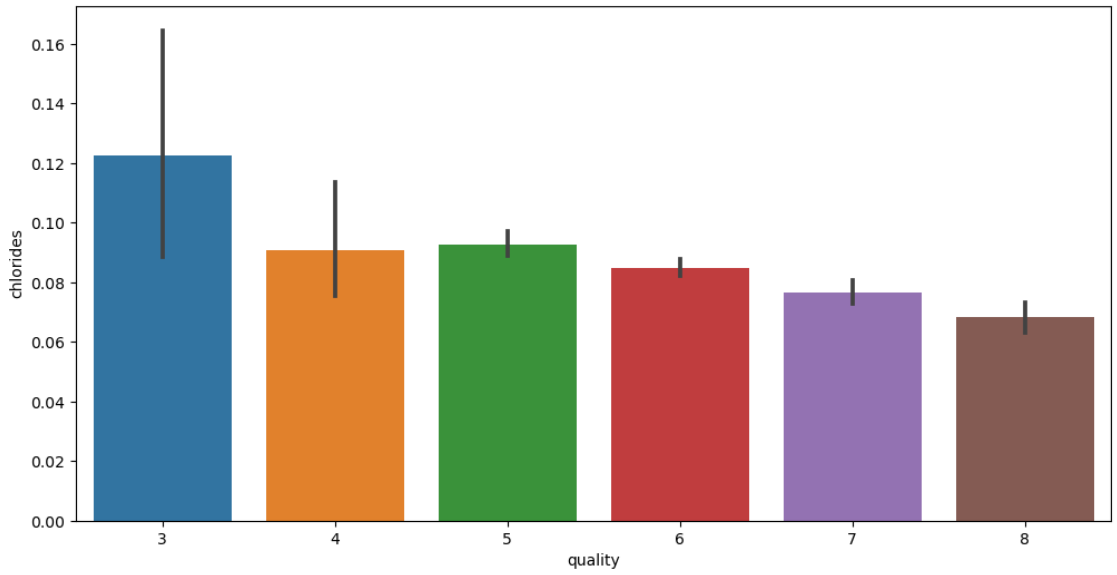
[10]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='quality',y='citric acid',data=df)
plt.show()
```



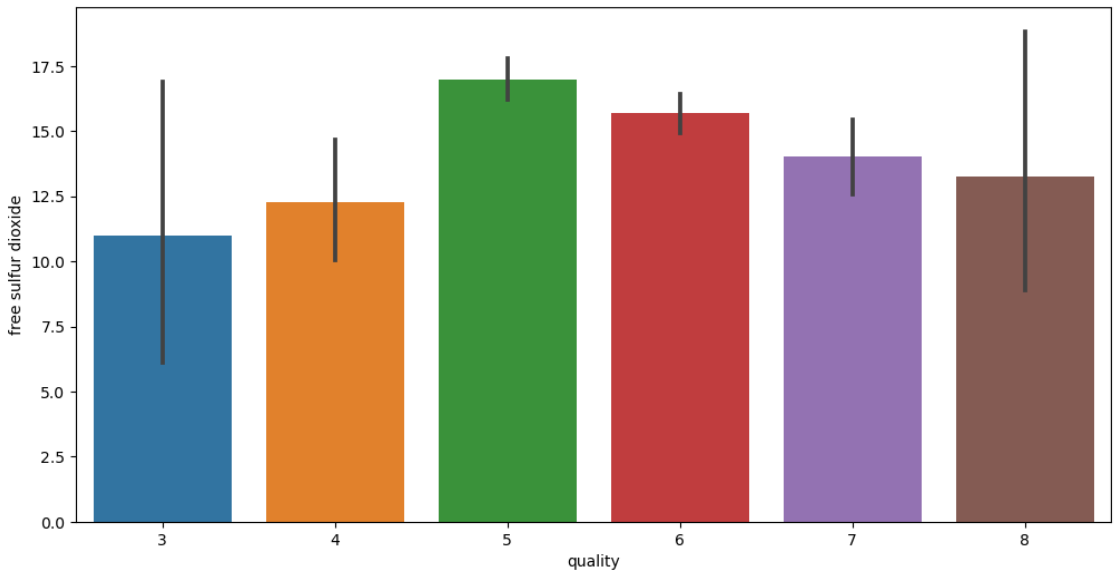
[11]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='quality',y='residual sugar',data=df)
plt.show()
```



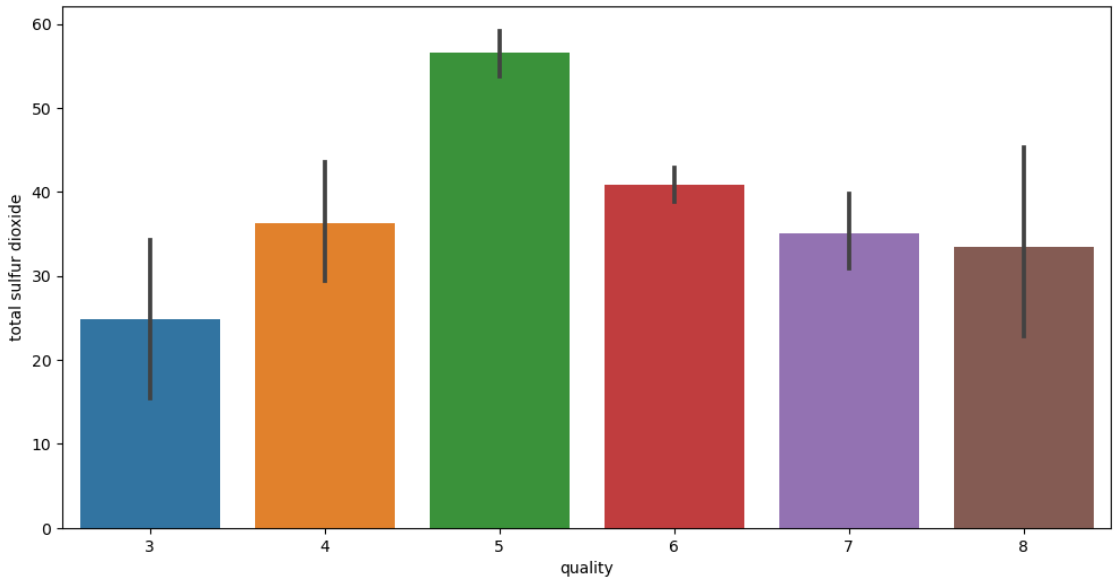
[12]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='quality',y='chlorides',data=df)
plt.show()
```



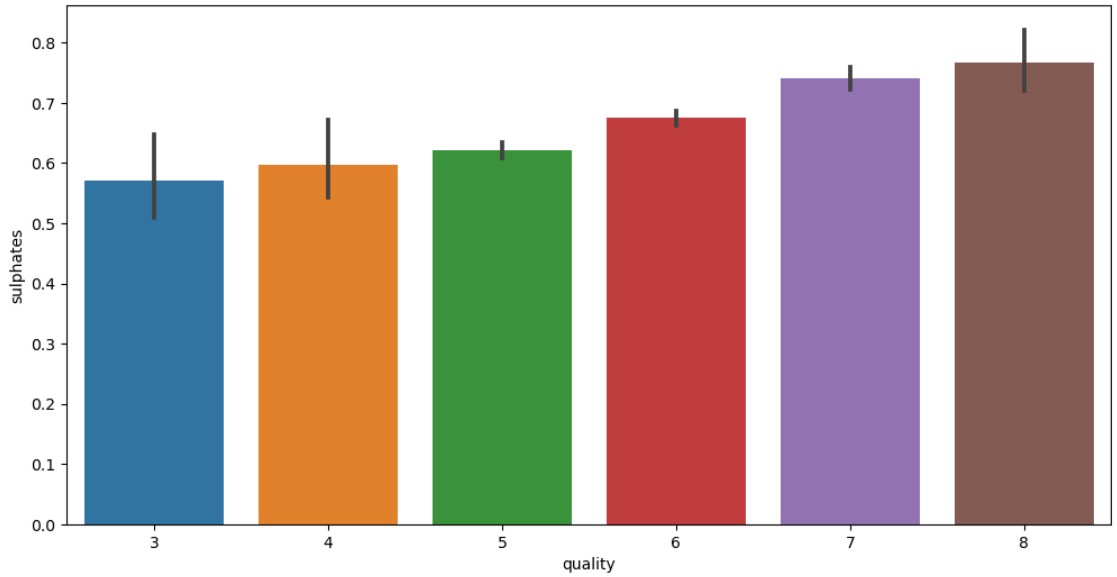
[13]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='quality',y='free sulfur dioxide',data=df)
plt.show()
```



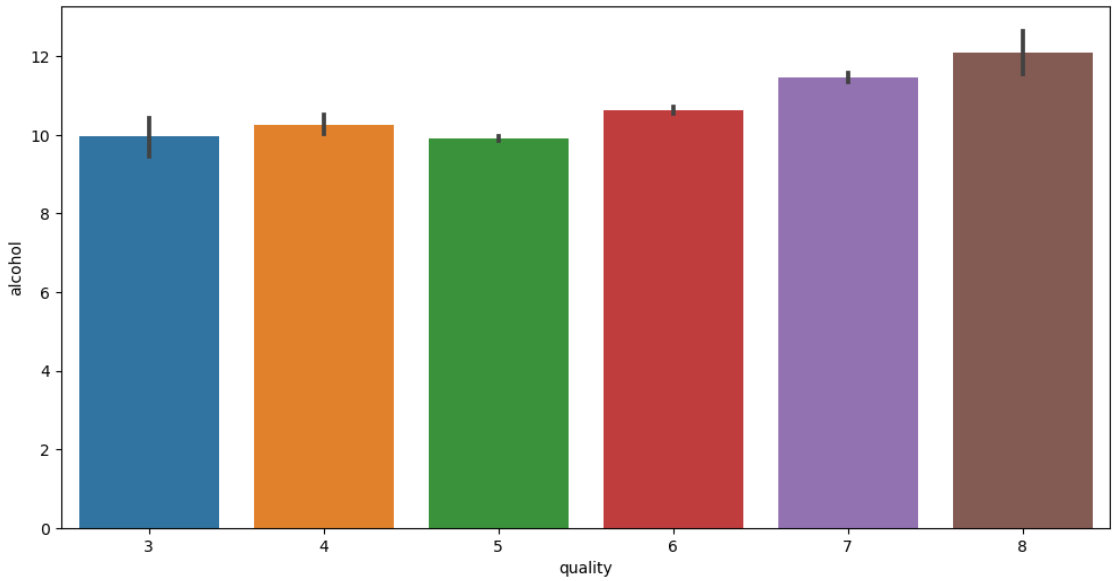
[14]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='quality',y='total sulfur dioxide',data=df)
plt.show()
```



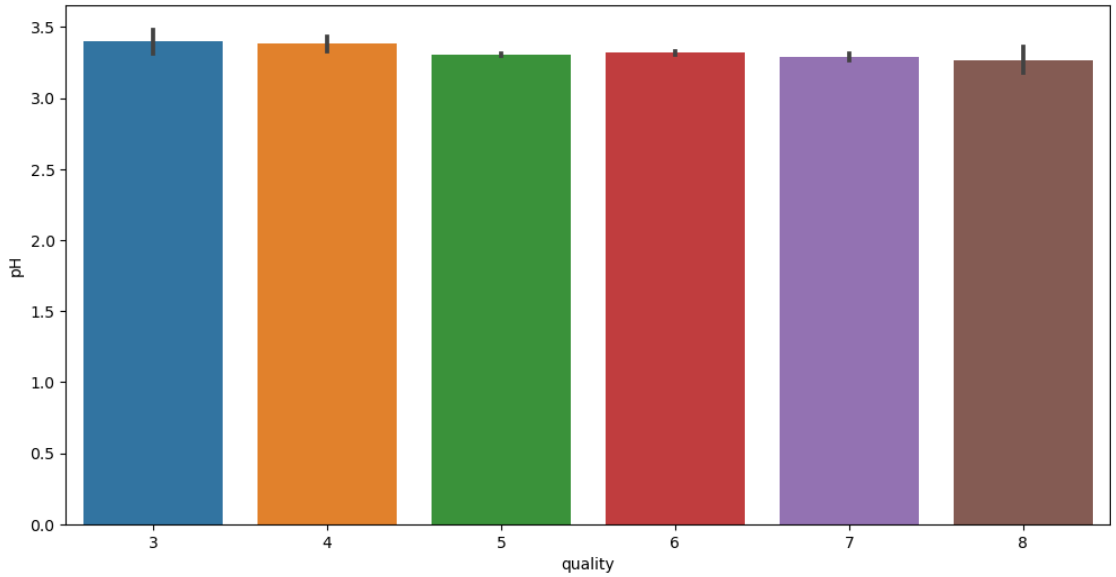
[15]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='quality',y='sulphates',data=df)
plt.show()
```



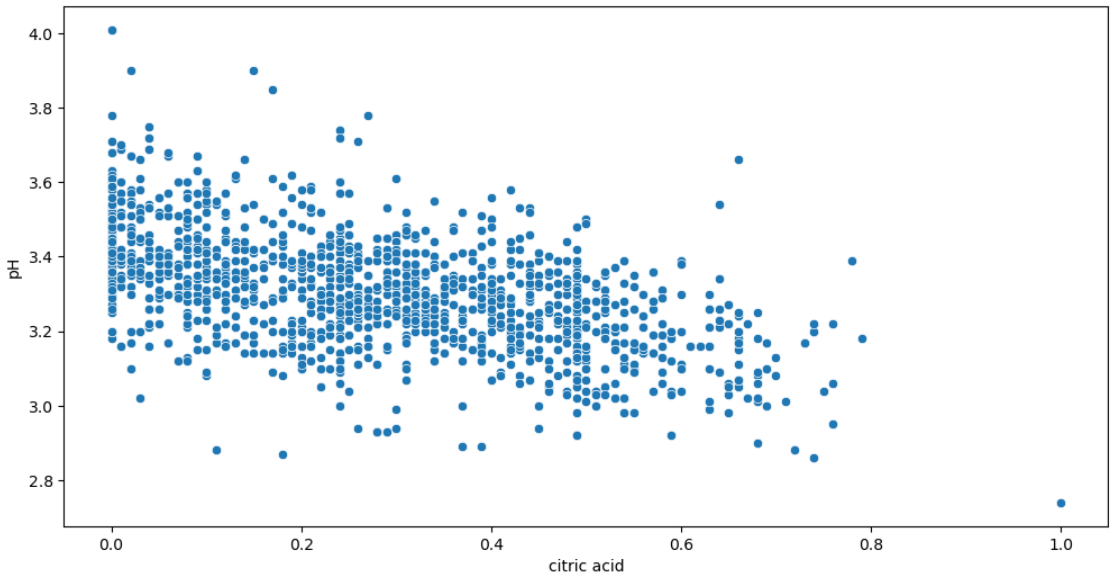
[16]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='quality',y='alcohol',data=df)
plt.show()
```



[17]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='quality',y='pH',data=df)
plt.show()
```



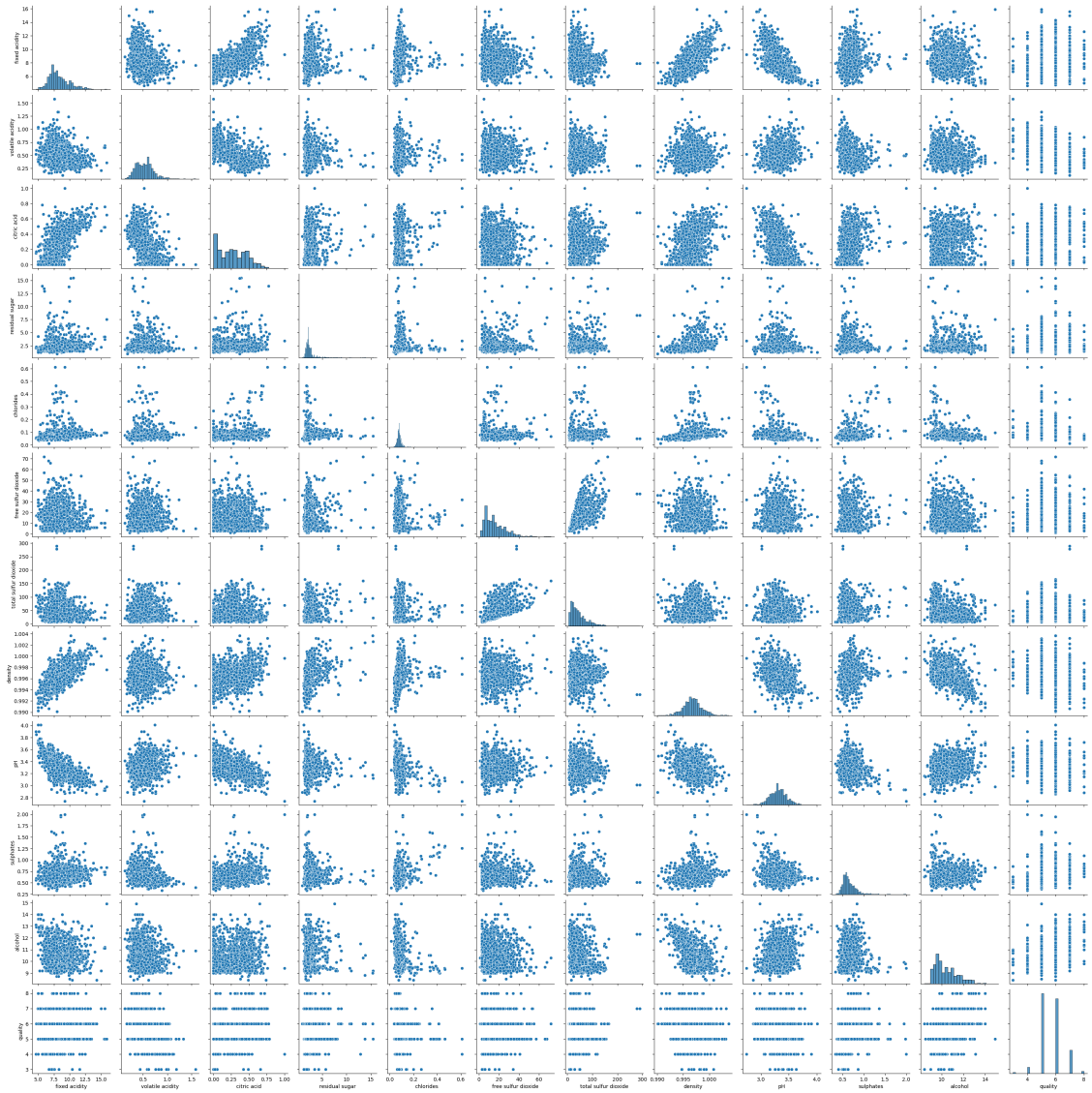
[18]:
```
plt.figure(figsize=(12,6))
sns.scatterplot(x='citric acid',y='pH',data=df)
plt.show()
```



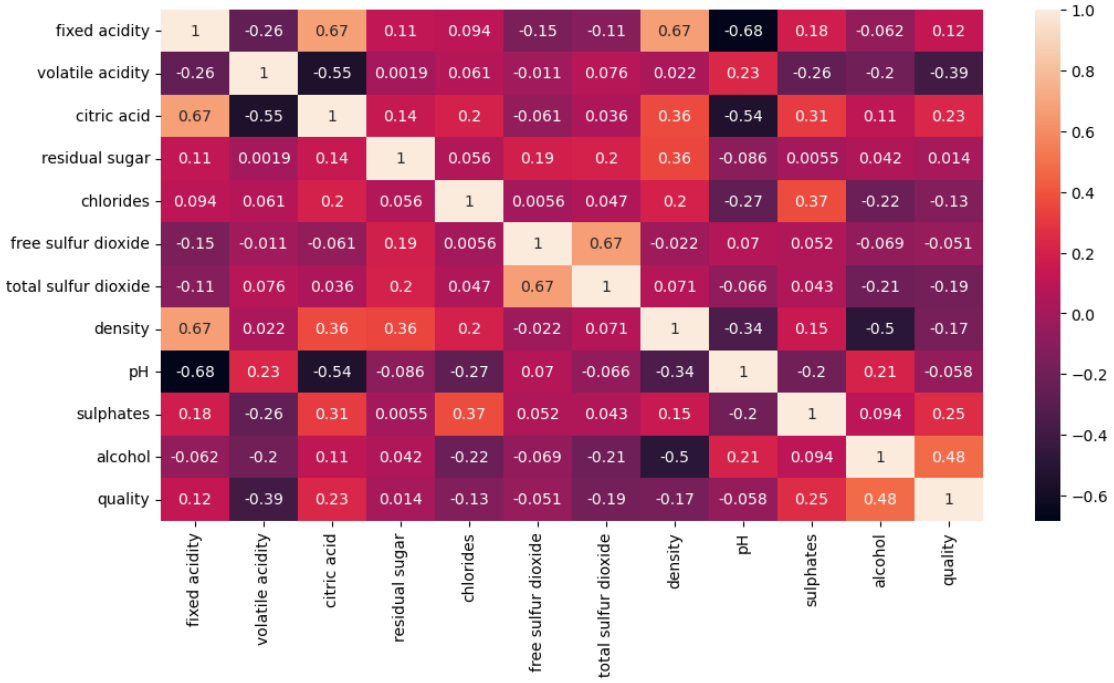
[19]:
```
plt.figure(figsize=(12,6))
sns.pairplot(df)
plt.show()
```

```
D:\Users\RAVI\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

```
<Figure size 1200x600 with 0 Axes>
```



[20]:
```
plt.figure(figsize=(12,6))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



[21]:
```
#preprocessing the data
```

[22]:
```
df['quality'].value_counts()
```

[22]:
```
quality
5    681
6    638
7    199
4     53
8     18
3     10
Name: count, dtype: int64
```

[23]:
```
#classifying the wine quality as good or bad based on its quality
#bad or 0 if quality lies in (3,6)
#good or 1 if quality lies in (7,8)

df['quality']=df['quality'].apply(lambda x:1 if x>6.5 else 0)
df.head()
```

[23]:

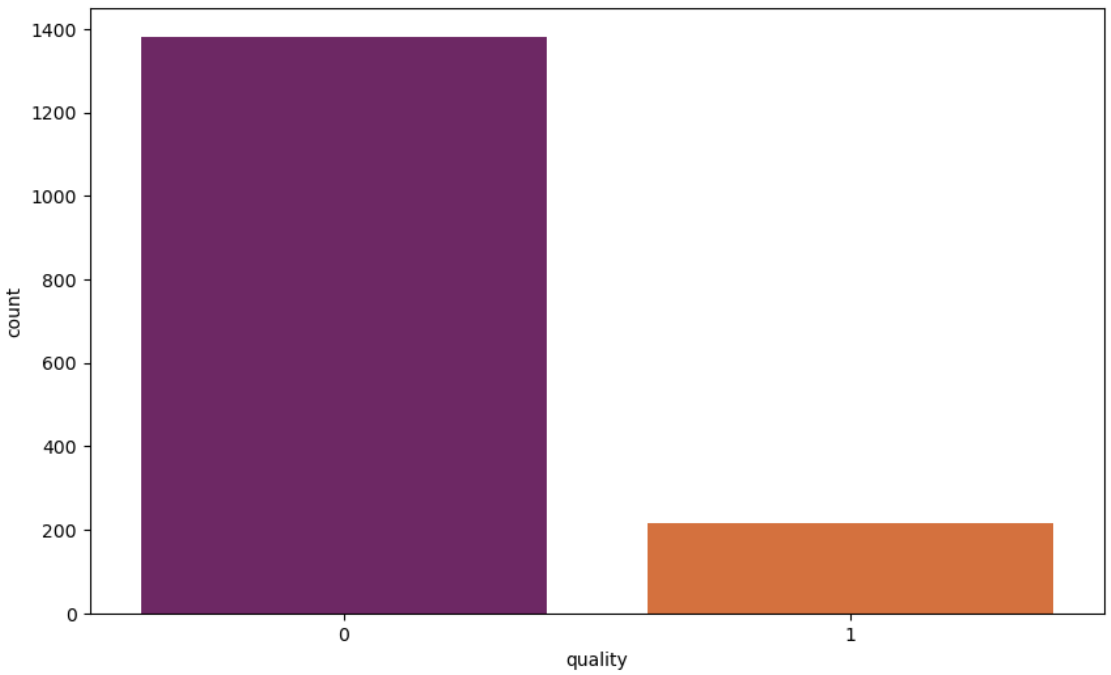
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	\
0	7.4	0.70	0.00	1.9	0.076	
1	7.8	0.88	0.00	2.6	0.098	
2	7.8	0.76	0.04	2.3	0.092	
3	11.2	0.28	0.56	1.9	0.075	

```
4            7.4              0.70         0.00             1.9       0.076

   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

   alcohol  quality
0      9.4        0
1      9.8        0
2      9.8        0
3      9.8        0
4      9.4        0
```

[24]:
```
print(df['quality'].value_counts())
fig=plt.figure(figsize=(10,6))
sns.countplot(x='quality',data=df,palette='inferno')
```

```
quality
0    1382
1     217
Name: count, dtype: int64
```

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



[25]:
```
#separating to dependent and independent variables

x=df.drop(['quality'],axis=1)
y=df['quality']
```

[26]:
```
x
```

[26]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	\
0	7.4	0.700	0.00	1.9	0.076	
1	7.8	0.880	0.00	2.6	0.098	
2	7.8	0.760	0.04	2.3	0.092	
3	11.2	0.280	0.56	1.9	0.075	
4	7.4	0.700	0.00	1.9	0.076	
…	…	…	…	…	…	
1594	6.2	0.600	0.08	2.0	0.090	
1595	5.9	0.550	0.10	2.2	0.062	
1596	6.3	0.510	0.13	2.3	0.076	
1597	5.9	0.645	0.12	2.0	0.075	
1598	6.0	0.310	0.47	3.6	0.067	

	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	\
0	11.0	34.0	0.99780	3.51	0.56	
1	25.0	67.0	0.99680	3.20	0.68	
2	15.0	54.0	0.99700	3.26	0.65	
3	17.0	60.0	0.99800	3.16	0.58	
4	11.0	34.0	0.99780	3.51	0.56	
…	…	…	…	…	…	
1594	32.0	44.0	0.99490	3.45	0.58	
1595	39.0	51.0	0.99512	3.52	0.76	
1596	29.0	40.0	0.99574	3.42	0.75	
1597	32.0	44.0	0.99547	3.57	0.71	
1598	18.0	42.0	0.99549	3.39	0.66	

	alcohol
0	9.4
1	9.8
2	9.8
3	9.8
4	9.4
…	…
1594	10.5
1595	11.2
1596	11.0
1597	10.2
1598	11.0

```
[1599 rows x 11 columns]
```

[27]:
```
y
```

[27]:
```
0       0
1       0
2       0
3       0
4       0
       ..
1594    0
1595    0
1596    0
1597    0
1598    0
Name: quality, Length: 1599, dtype: int64
```

[28]:
```
#splitting into train and test sets
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.
 ↪2,random_state=42)
```

[29]:
```
print("x-train shape:",x_train.shape)
print("x-test shape:",x_test.shape)
print("y-train shape:",y_train.shape)
print("y-test shape:",y_test.shape)
```

```
x-train shape: (1279, 11)
x-test shape: (320, 11)
y-train shape: (1279,)
y-test shape: (320,)
```

[30]:
```
#applying standard scaling to the dataset to scale all the field values to same␣
 ↪scale(approx.)
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

[31]:
```
#logisticregression

lr=LogisticRegression()
lr.fit(x_train,y_train)
predictions = lr.predict(x_test)
accuracy_score(y_test,predictions)
```

[31]:
```
0.875
```

[32]:
```
#decisiontreeclassifier

dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
accuracy_score(y_test,dt.predict(x_test))
```

[32]: 0.846875

[33]:
```
#randomforestclassifier

rf=RandomForestClassifier(random_state=42)
rf.fit(x_train,y_train)
accuracy_score(y_test,rf.predict(x_test))
```

[33]: 0.875

[ ]:
```

```