In [9]:
```
#Importing required packages.
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
%matplotlib inline
```

In [10]:
```
#Loading dataset
wine = pd.read_csv('C:/Users/computer world/OneDrive/Desktop/winequality-red.csv')
```

In [11]:
```
#Let's check how the data is distributed
wine.head()
```

Out[11]:

	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	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

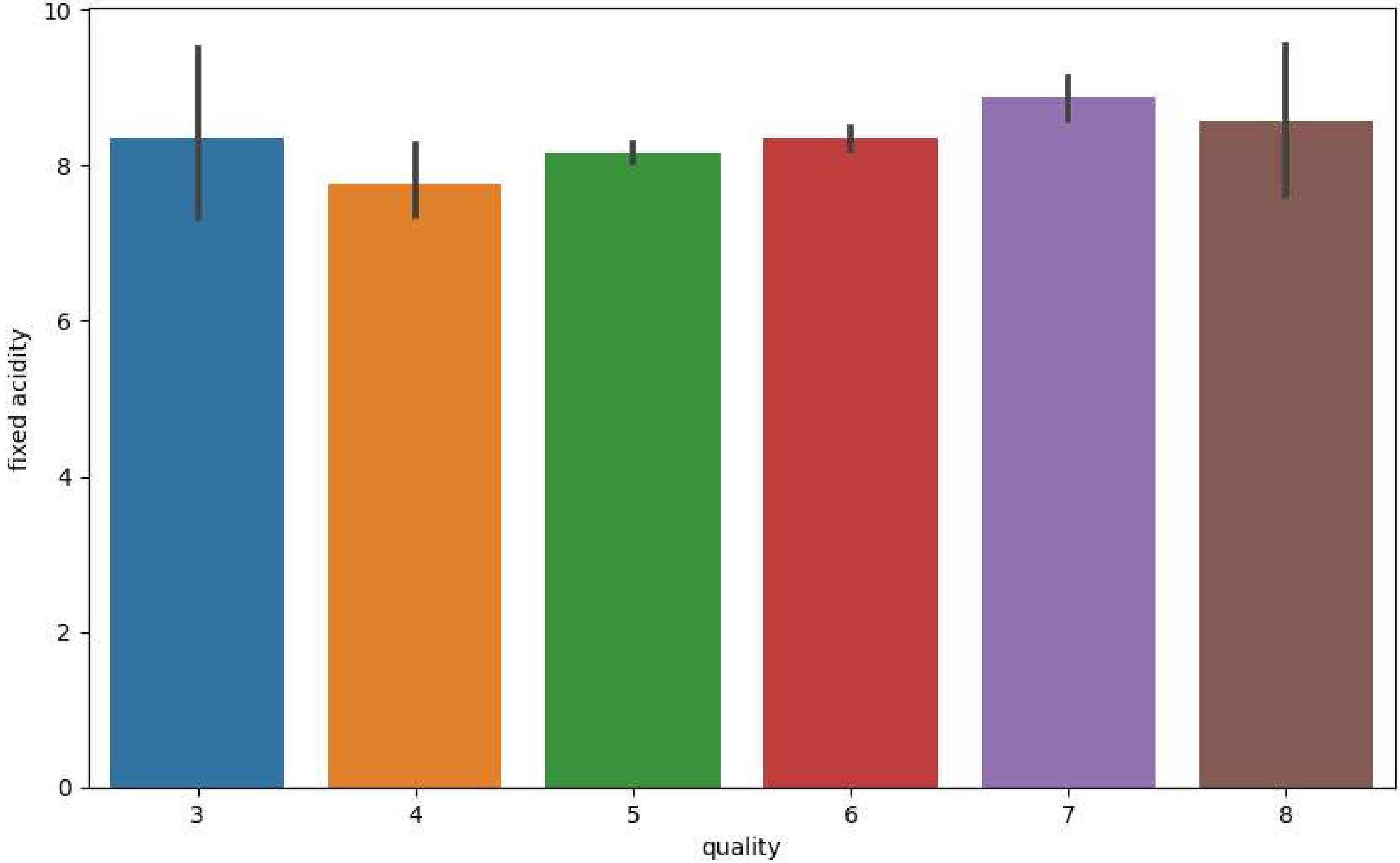
In [12]:
```python
#Information about the data columns
wine.info()
```

```
<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
```

Let's do some plotting to know how the data columns are distributed in the dataset

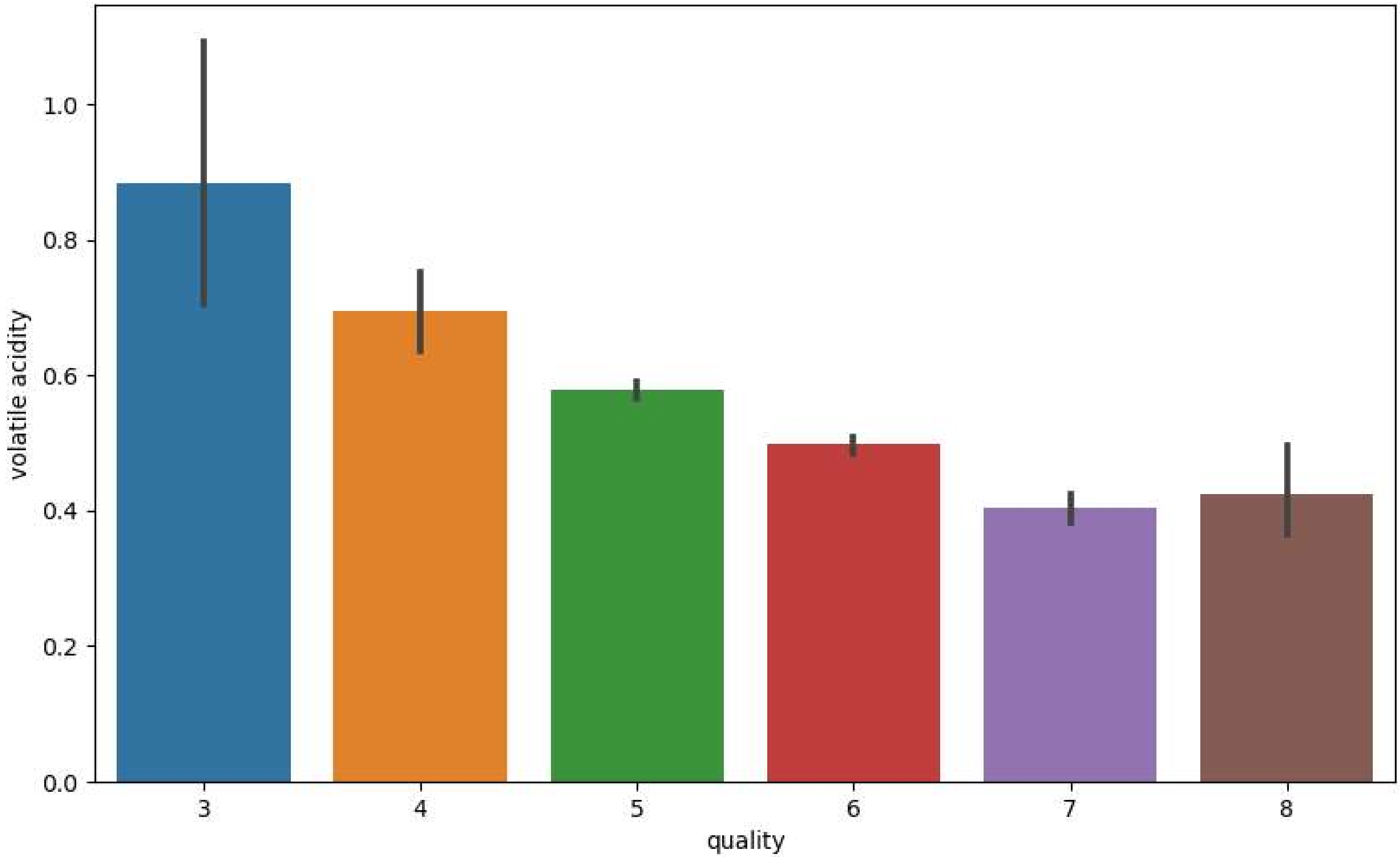
In [13]:
```python
#Here we see that fixed acidity does not give any specification to classify the quality.
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'fixed acidity', data = wine)
```

Out[13]: <Axes: xlabel='quality', ylabel='fixed acidity'>



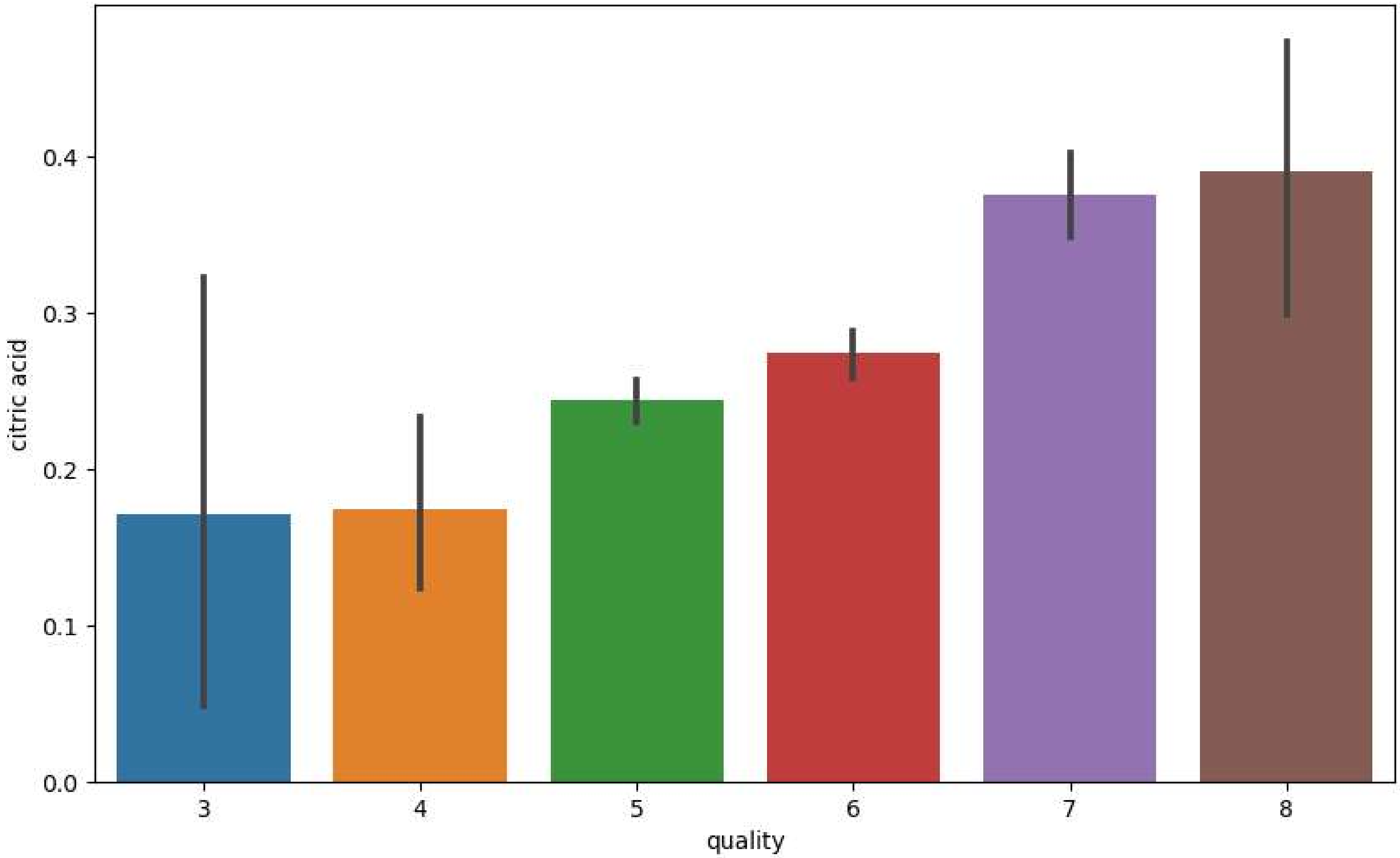
In [14]:
```
#Here we see that its quite a downing trend in the volatile acidity as we go higher the quality
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'volatile acidity', data = wine)
```

Out[14]: <Axes: xlabel='quality', ylabel='volatile acidity'>



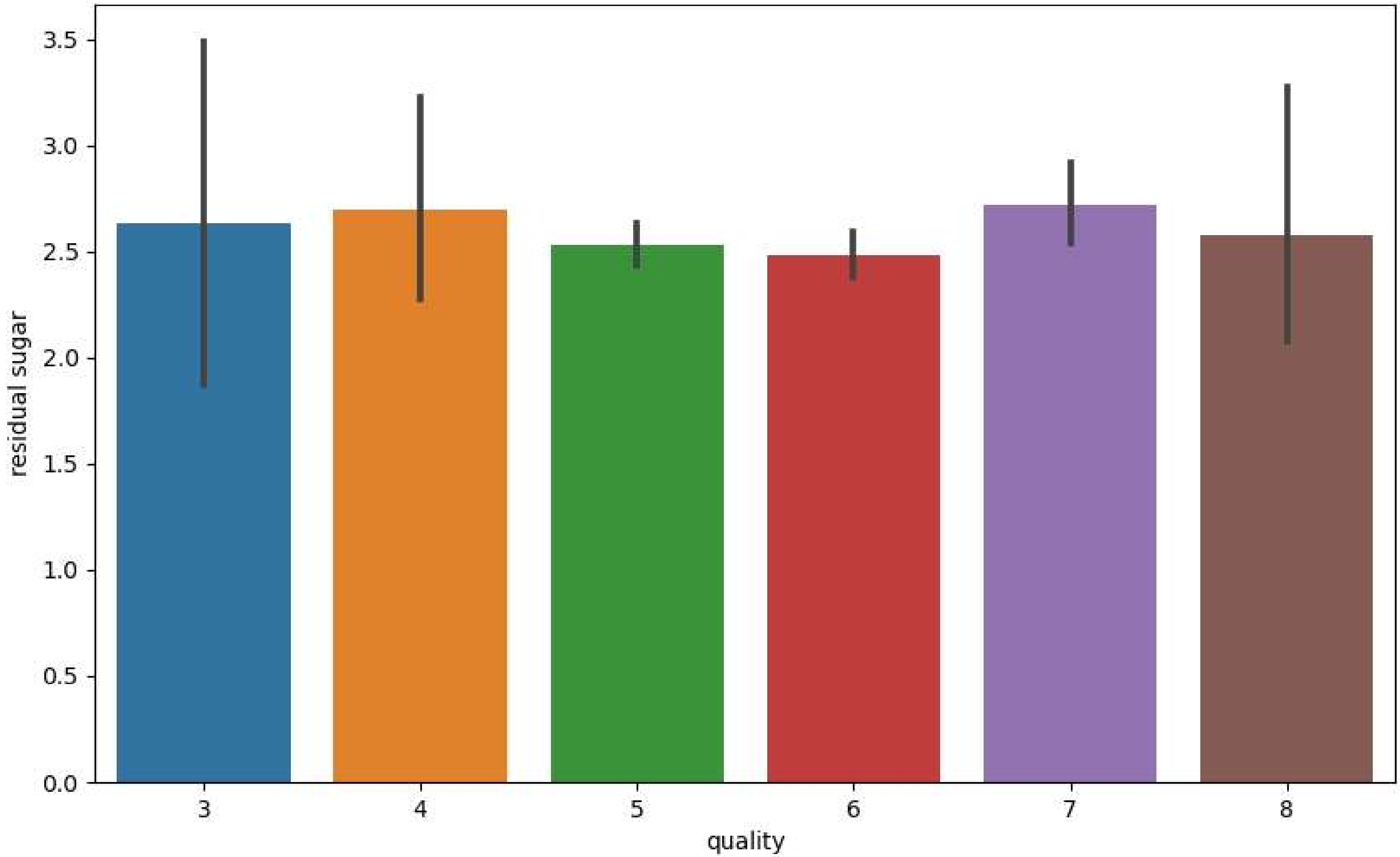
In [15]:
```
#Composition of citric acid go higher as we go higher in the quality of the wine
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'citric acid', data = wine)
```

Out[15]: <Axes: xlabel='quality', ylabel='citric acid'>



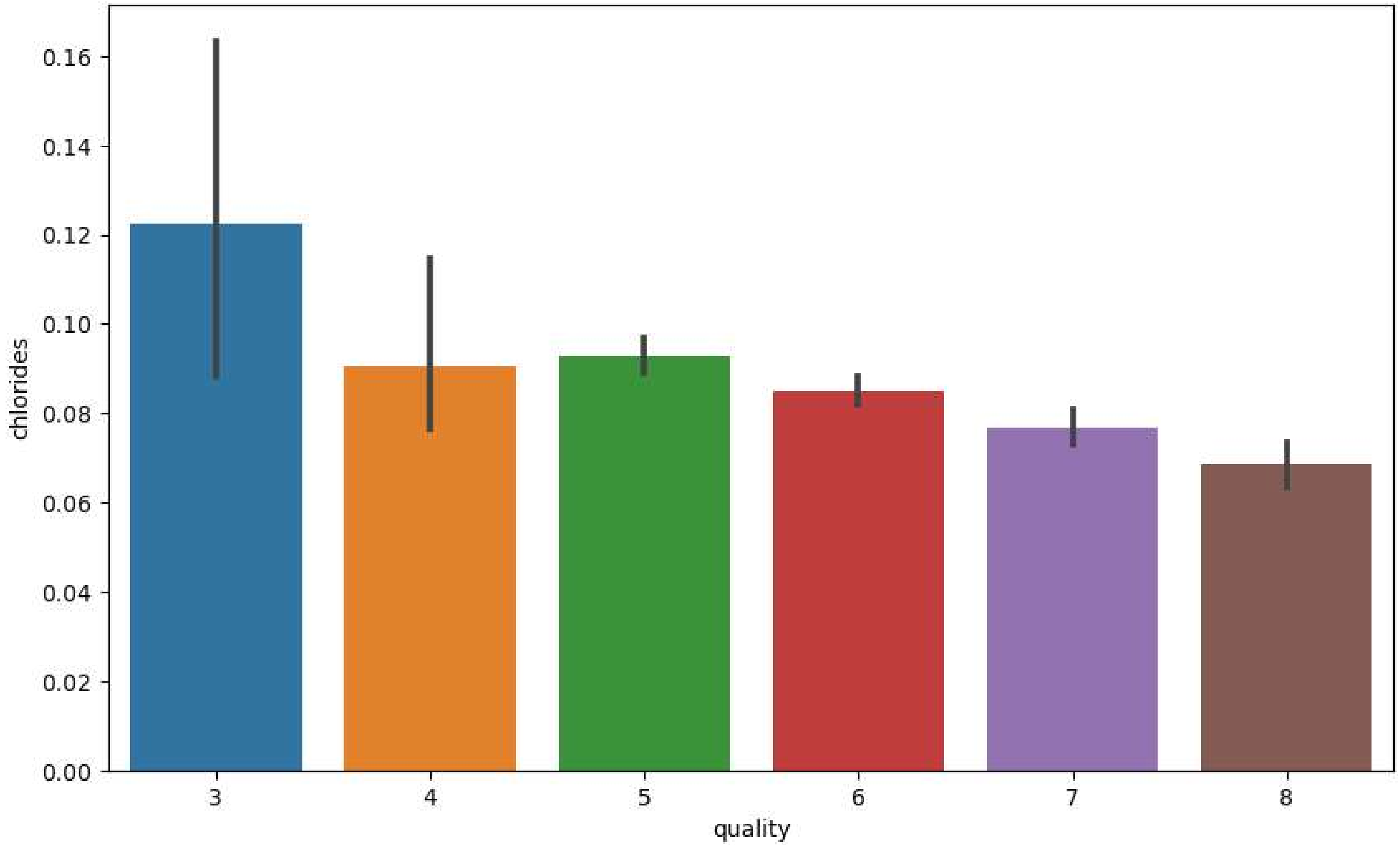
In [16]:
```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'residual sugar', data = wine)
```

Out[16]: <Axes: xlabel='quality', ylabel='residual sugar'>



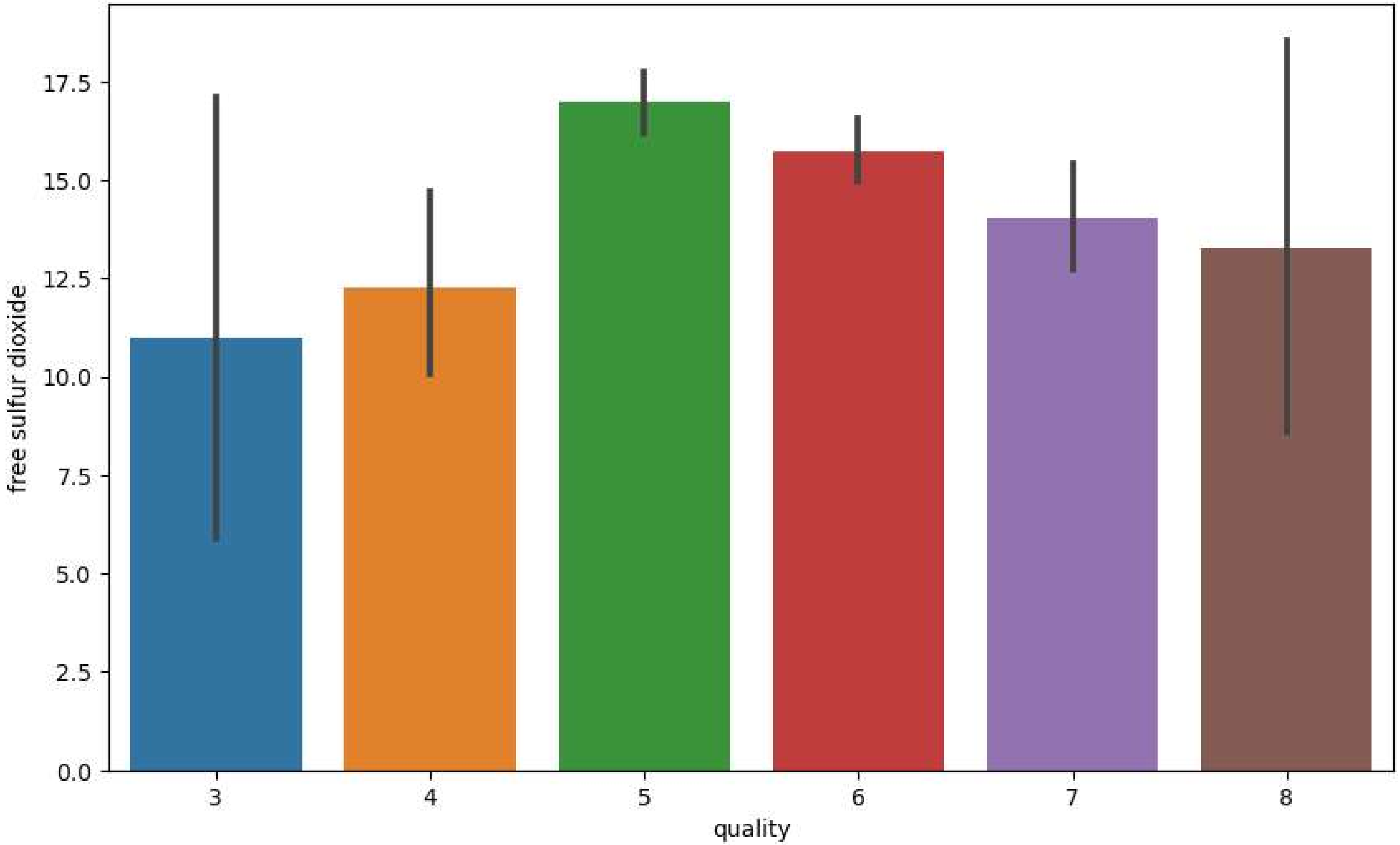
In [17]:
```python
#Composition of chloride also go down as we go higher in the quality of the wine
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'chlorides', data = wine)
```

Out[17]: <Axes: xlabel='quality', ylabel='chlorides'>



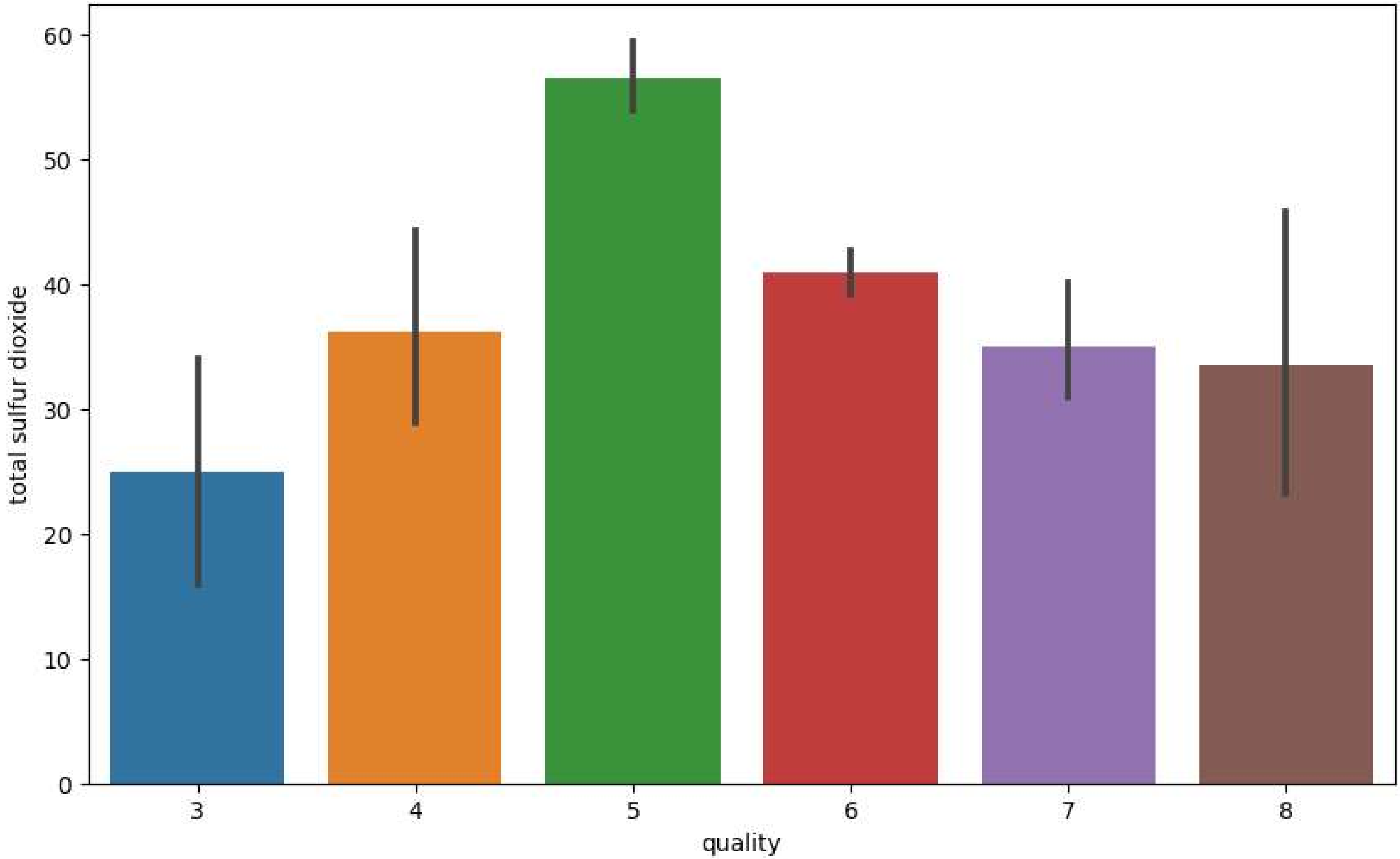
In [18]:
```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'free sulfur dioxide', data = wine)
```

Out[18]: <Axes: xlabel='quality', ylabel='free sulfur dioxide'>



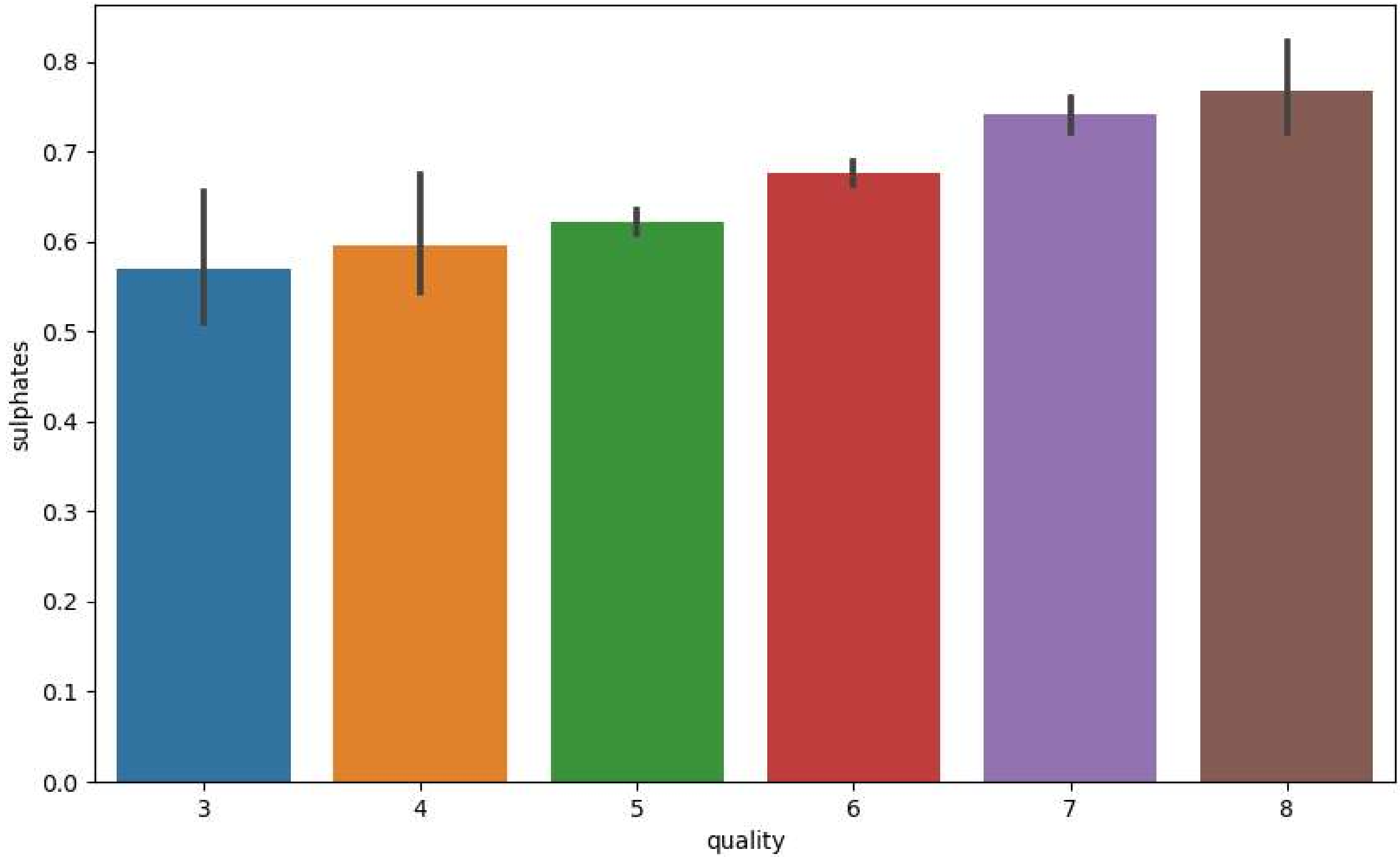
In [19]:
```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'total sulfur dioxide', data = wine)
```

Out[19]: <Axes: xlabel='quality', ylabel='total sulfur dioxide'>



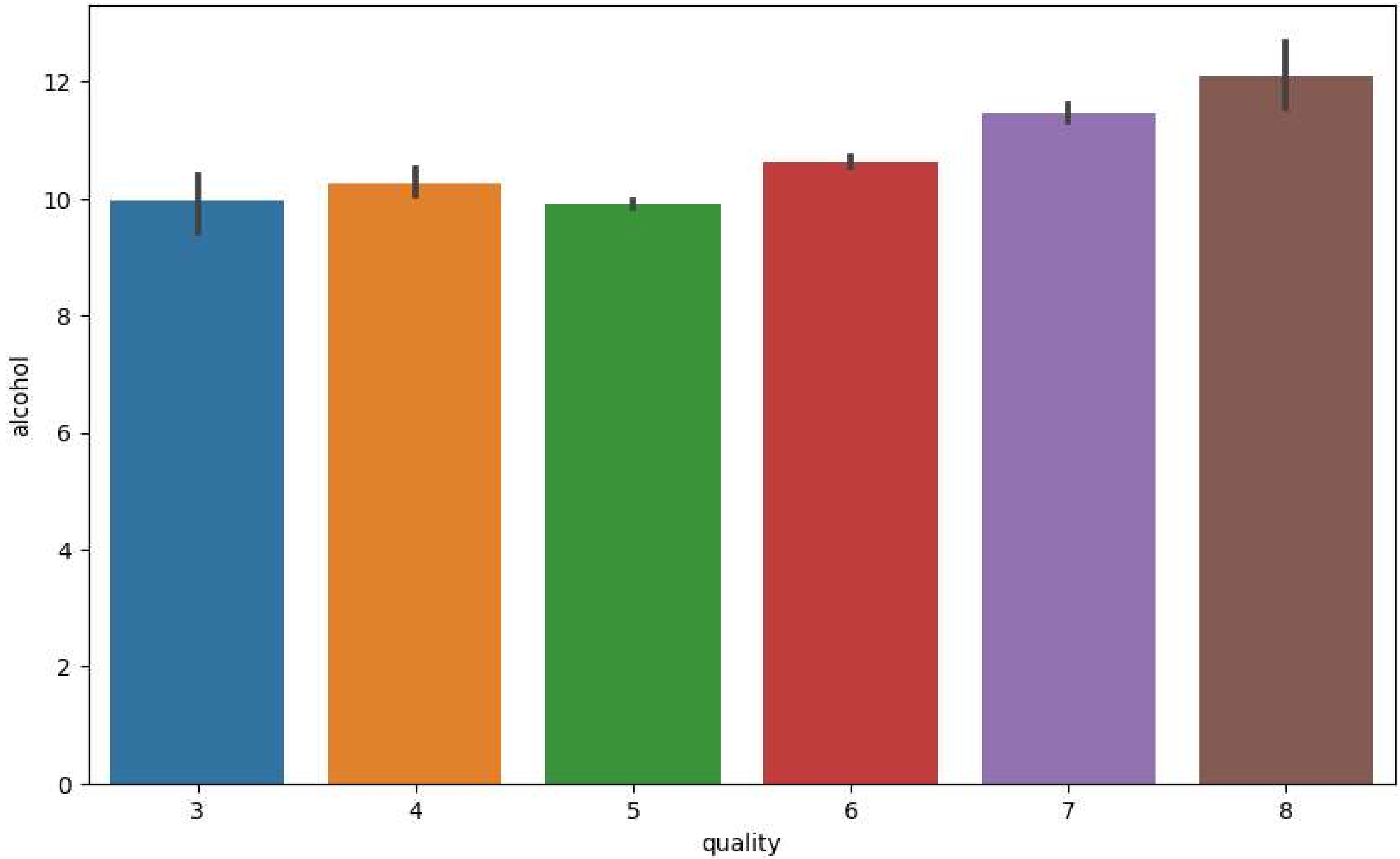
In [20]:
```
#Sulphates level goes higher with the quality of wine
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'sulphates', data = wine)
```

Out[20]: <Axes: xlabel='quality', ylabel='sulphates'>



In [21]:
```
#Alcohol level also goes higher as te quality of wine increases
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'alcohol', data = wine)
```

Out[21]: <Axes: xlabel='quality', ylabel='alcohol'>



Preprocessing Data for performing Machine learning algorithms

In [22]:
```
#Making binary classificaion for the response variable.
#Dividing wine as good and bad by giving the limit for the quality
bins = (2, 6.5, 8)
group_names = ['bad', 'good']
wine['quality'] = pd.cut(wine['quality'], bins = bins, labels = group_names)
```

In [23]:
```
#Now lets assign a labels to our quality variable
label_quality = LabelEncoder()
```

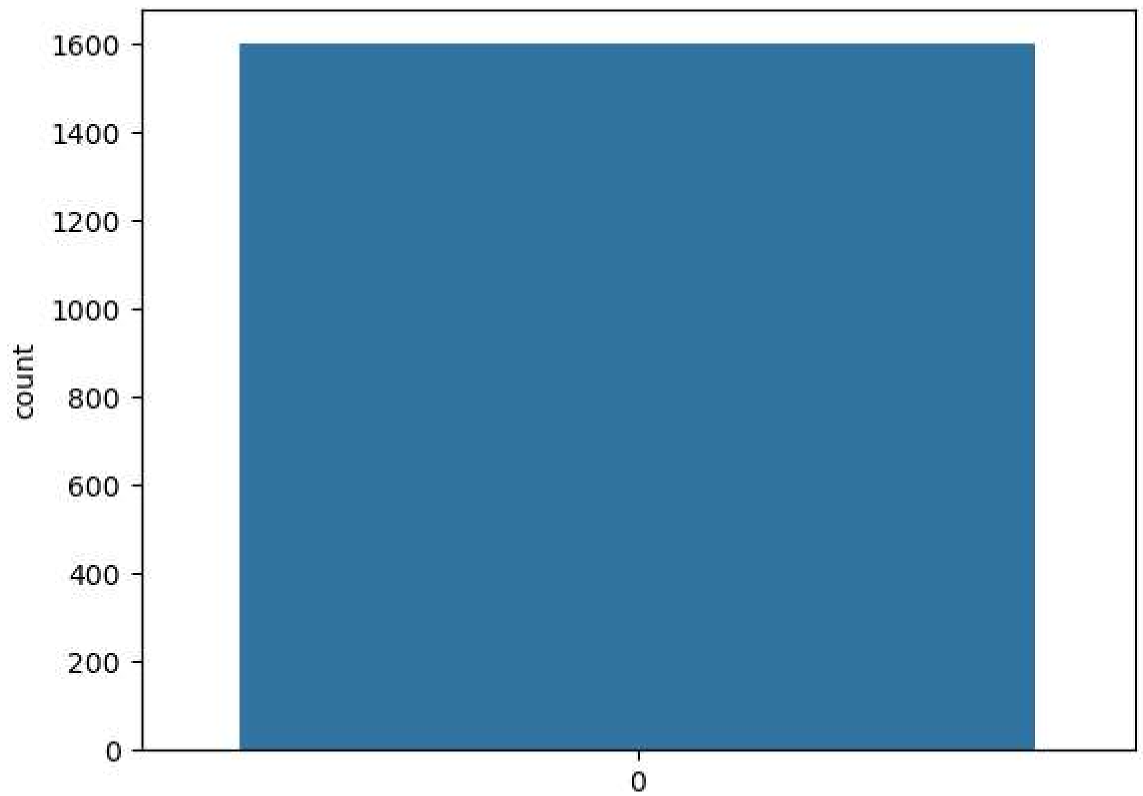
In [24]:
```
#Bad becomes 0 and good becomes 1
wine['quality'] = label_quality.fit_transform(wine['quality'])
```

In [25]:
```
wine['quality'].value_counts()
```

Out[25]:
```
0    1382
1     217
Name: quality, dtype: int64
```

In [26]:
```
sns.countplot(wine['quality'])
```

Out[26]: <Axes: ylabel='count'>



In [27]:
```
#Now seperate the dataset as response variable and feature variabes
X = wine.drop('quality', axis = 1)
y = wine['quality']
```

In [28]:
```
#Train and Test splitting of data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

In [29]:
```
#Applying Standard scaling to get optimized result
sc = StandardScaler()
```

In [30]:
```
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

Our training and testing data is ready now to perform machine learning algorithm

In [31]:
```
rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train, y_train)
pred_rfc = rfc.predict(X_test)
```

In [32]:
```
#Let's see how our model performed
print(classification_report(y_test, pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.90      0.97      0.93       273
           1       0.68      0.40      0.51        47

    accuracy                           0.88       320
   macro avg       0.79      0.69      0.72       320
weighted avg       0.87      0.88      0.87       320
```

Random forest gives the accuracy of 87%

In [33]:
```
#Confusion matrix for the random forest classification
print(confusion_matrix(y_test, pred_rfc))
```

```
[[264   9]
 [ 28  19]]
```

Stochastic Gradient Decent Classifier

In [34]:
```
sgd = SGDClassifier(penalty=None)
sgd.fit(X_train, y_train)
pred_sgd = sgd.predict(X_test)
```

In [35]:
```
print(classification_report(y_test, pred_sgd))
```

```
              precision    recall  f1-score   support

           0       0.95      0.86      0.90       273
           1       0.47      0.74      0.58        47

    accuracy                           0.84       320
   macro avg       0.71      0.80      0.74       320
weighted avg       0.88      0.84      0.85       320
```

84% accuracy using stochastic gradient descent classifier

In [36]:
```
print(confusion_matrix(y_test, pred_sgd))
```

```
[[234  39]
 [ 12  35]]
```

Support Vector Classifier

In [37]:
```
svc = SVC()
svc.fit(X_train, y_train)
pred_svc = svc.predict(X_test)
```

In [38]:
```python
print(classification_report(y_test, pred_svc))
```

```
              precision    recall  f1-score   support

           0       0.88      0.98      0.93       273
           1       0.71      0.26      0.37        47

    accuracy                           0.88       320
   macro avg       0.80      0.62      0.65       320
weighted avg       0.86      0.88      0.85       320
```

Let's try to increase our accuracy of models

Grid Search CV

In [43]:
```python
#Finding best parameters for our SVC model
param = {
    'C': [0.1,0.8,0.9,1,1.1,1.2,1.3,1.4],
    'kernel':['linear', 'rbf'],
    'gamma' :[0.1,0.8,0.9,1,1.1,1.2,1.3,1.4]
}
grid_svc = GridSearchCV(svc, param_grid=param, scoring='accuracy', cv=10)
```

In [44]:
```python
grid_svc.fit(X_train, y_train)
```

Out[44]:
```
▸ GridSearchCV
▸ estimator: SVC
   ▸ SVC
```

In [45]:
```python
#Best parameters for our svc model
grid_svc.best_params_
```

Out[45]:
```
{'C': 1.2, 'gamma': 0.9, 'kernel': 'rbf'}
```

In [47]:
```
#Let's run our SVC again with the best parameters.
svc2 = SVC(C = 1.2, gamma =  0.9, kernel= 'rbf')
svc2.fit(X_train, y_train)
pred_svc2 = svc2.predict(X_test)
print(classification_report(y_test, pred_svc2))
```

```
              precision    recall  f1-score   support

           0       0.90      0.99      0.94       273
           1       0.89      0.34      0.49        47

    accuracy                           0.90       320
   macro avg       0.89      0.67      0.72       320
weighted avg       0.90      0.90      0.88       320
```

SVC improves from 86% to 90% using Grid Search CV

Cross Validation Score for random forest and SGD

In [48]:
```
#Now lets try to do some evaluation for random forest model using cross validation.
rfc_eval = cross_val_score(estimator = rfc, X = X_train, y = y_train, cv = 10)
rfc_eval.mean()
```

Out[48]:
```
0.913238188976378
```

In [ ]:
```
```