In [38]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
warnings.filterwarnings("ignore")
```

In [ ]:

In [8]:

```
df=pd.read_csv("https://raw.githubusercontent.com/dsrscientist/DSData/maste
r/winequality-red.csv")
df
```

Out[8]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

In [9]:

```
df.shape
```

Out[9]:

```
(1599, 12)
```

In [10]:

```
df.head()
```

Out[10]:

	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

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
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]:

```
df.isnull().sum()
```

Out[12]:

```
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 [13]:

```
df.isnull().sum().sum()
```

Out[13]:

```
0
```

In [15]:

```
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
```

In [16]:

```
df["quality"].unique()
```

Out[16]:

```
array([5, 6, 7, 4, 8, 3], dtype=int64)
```

In [17]:

```
df["quality"].nunique()
```

Out[17]:

```
6
```

In [18]:

```
for i in df.columns:
    print(df[i].value_counts())
    print('\n')
```

```
7.2     67
7.1     57
7.8     53
7.5     52
7.0     50
        ..
13.5     1
13.8     1
13.4     1
4.7      1
5.5      1
Name: fixed acidity, Length: 96, dtype: int64


0.600    47
0.500    46
0.430    43
0.590    39
0.360    38
         ..
1.035     1
0.565     1
1.130     1
1.115     1
1.090     1
Name: volatile acidity, Length: 143, dtype: int64


0.00    132
0.49     68
0.24     51
0.02     50
0.26     38
       ...
0.72      1
0.62      1
0.75      1
1.00      1
0.78      1
Name: citric acid, Length: 80, dtype: int64
```

```
2.00     156
2.20     131
1.80     129
2.10     128
1.90     117
        ...
4.25       1
2.85       1
3.45       1
2.35       1
13.90      1
Name: residual sugar, Length: 91, dtype: int64


0.080    66
0.074    55
0.076    51
0.078    51
0.084    49
         ..
0.108     1
0.148     1
0.143     1
0.222     1
0.230     1
Name: chlorides, Length: 153, dtype: int64


6.0     138
5.0     104
10.0     79
15.0     78
12.0     75
7.0      71
9.0      62
16.0     61
17.0     60
11.0     59
13.0     57
8.0      56
14.0     50
3.0      49
18.0     46
4.0      41
21.0     41
19.0     39
24.0     34
26.0     32
23.0     32
```

```
20.0     30
27.0     29
25.0     24
29.0     23
28.0     23
22.0     22
32.0     22
31.0     20
34.0     18
30.0     16
35.0     15
36.0     11
33.0     11
38.0      9
41.0      7
40.0      6
39.0      5
48.0      4
51.0      4
45.0      3
1.0       3
43.0      3
42.0      3
52.0      3
37.0      3
68.0      2
55.0      2
50.0      2
37.5      2
53.0      1
72.0      1
57.0      1
47.0      1
5.5       1
2.0       1
46.0      1
54.0      1
40.5      1
66.0      1
Name: free sulfur dioxide, dtype: int64


28.0     43
24.0     36
15.0     35
18.0     35
23.0     34
         ..
139.0     1
```

```
149.0     1
152.0     1
155.0     1
165.0     1
Name: total sulfur dioxide, Length: 144, dtype: int64


0.99720    36
0.99680    35
0.99760    35
0.99800    29
0.99620    28
           ..
0.99684     1
0.99764     1
0.99473     1
0.99320     1
0.99651     1
Name: density, Length: 436, dtype: int64


3.30    57
3.36    56
3.26    53
3.38    48
3.39    48
        ..
3.75     1
2.74     1
3.70     1
3.85     1
2.90     1
Name: pH, Length: 89, dtype: int64


0.60    69
0.58    68
0.54    68
0.62    61
0.56    60
        ..
1.00     1
1.59     1
0.33     1
1.26     1
1.01     1
Name: sulphates, Length: 96, dtype: int64
```

```
9.500000     139
9.400000     103
9.800000      78
9.200000      72
10.000000     67
            ...
9.950000       1
9.233333       1
9.250000       1
9.050000       1
10.750000      1
Name: alcohol, Length: 65, dtype: int64
```

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

In [21]:

```
cat=[]
num=[]

for i in df.dtypes.index:
    if df.dtypes[i]=="object":
        cat.append(i)
print ("cat:",cat)
print('\n')

for i in df.dtypes.index:
    if df.dtypes[i]!="object":
        num.append(i)
print ("num:",num)
print('\n')
cat: []
```

```
num: ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
'pH', 'sulphates', 'alcohol', 'quality']
```

In [22]:

```
df.nunique().to_frame("Unique values count")
```

Out[22]:

	Unique values count
fixed acidity	96
volatile acidity	143
citric acid	80
residual sugar	91
chlorides	153
free sulfur dioxide	60
total sulfur dioxide	144
density	436
pH	89
sulphates	96
alcohol	65
quality	6

In [23]:

```
df.describe()
```

Out[23]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	de
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.00
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.9967
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.0018
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.9900
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.9956
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.9967
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.9978
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.0036

In [37]:

```
sns.catplot(x='quality',data=df,kind='count')
```

Out[37]:

```
<seaborn.axisgrid.FacetGrid at 0x1ec03d69160>
```

In [ ]:

In [ ]:

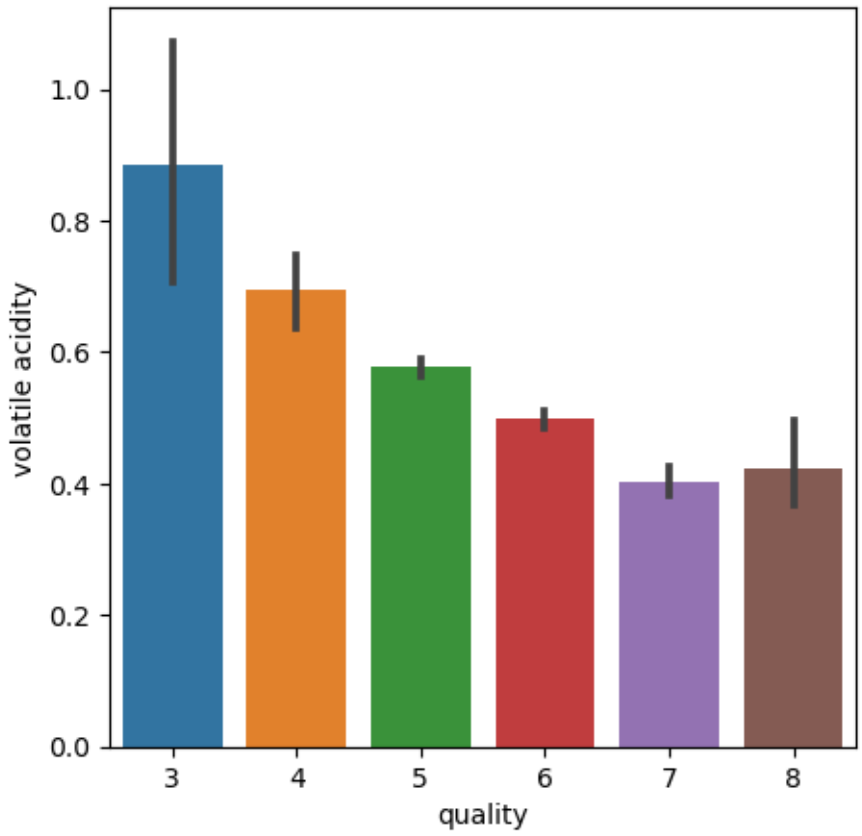
```
5 & 6 quality of wine is highest
```

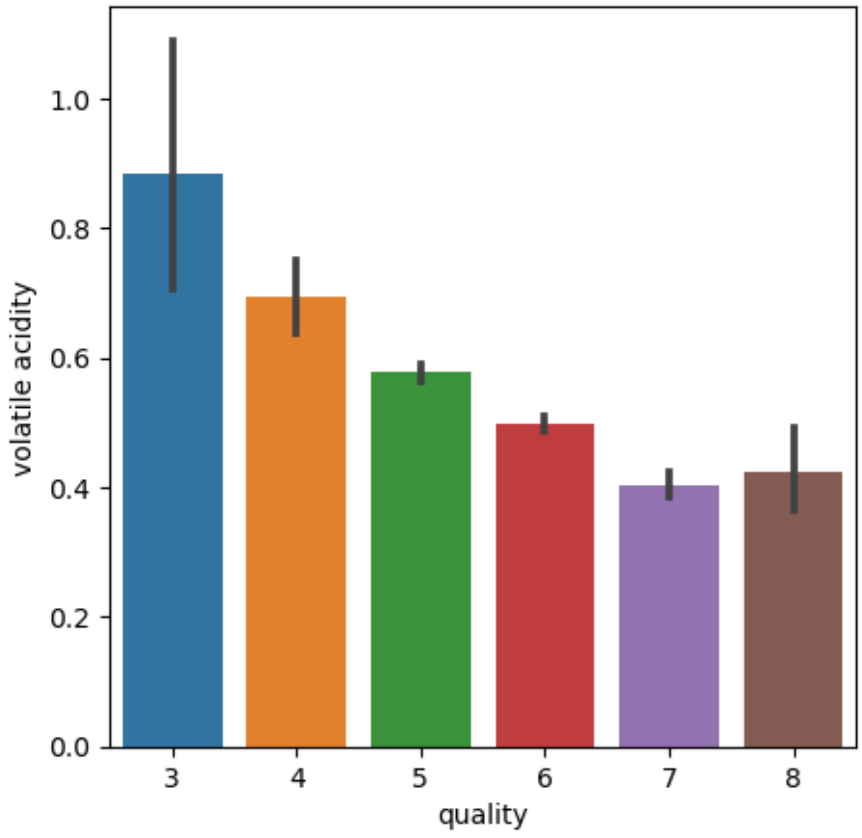
In [ ]:

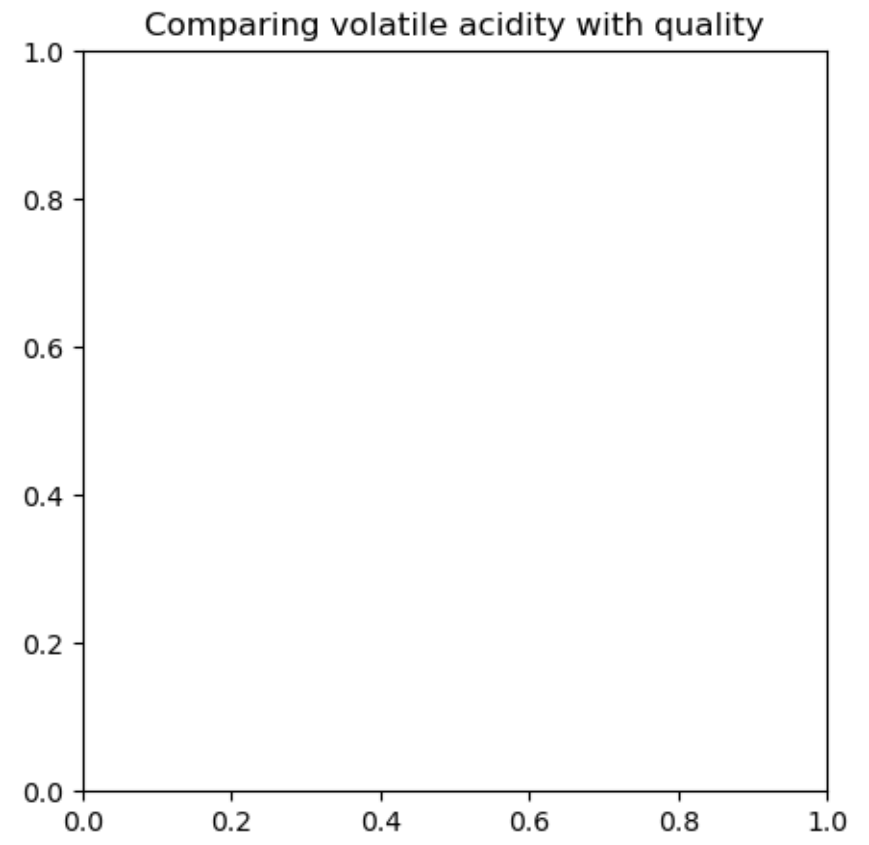
```
Comparing volatile acidity with quality
```

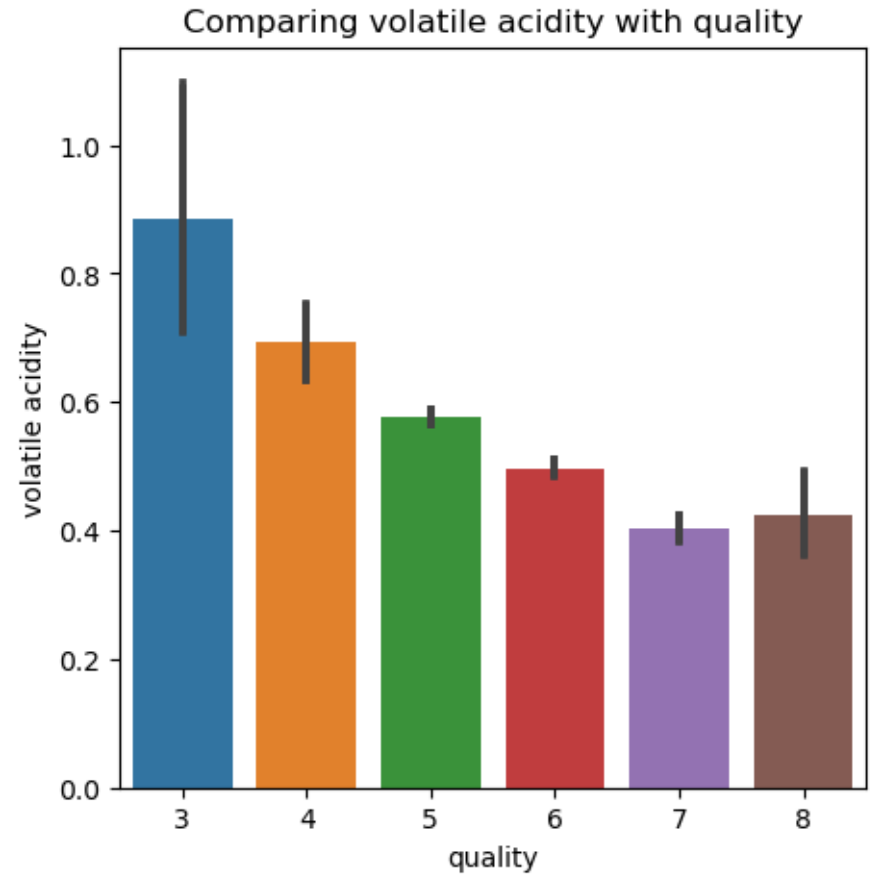
In [56]:

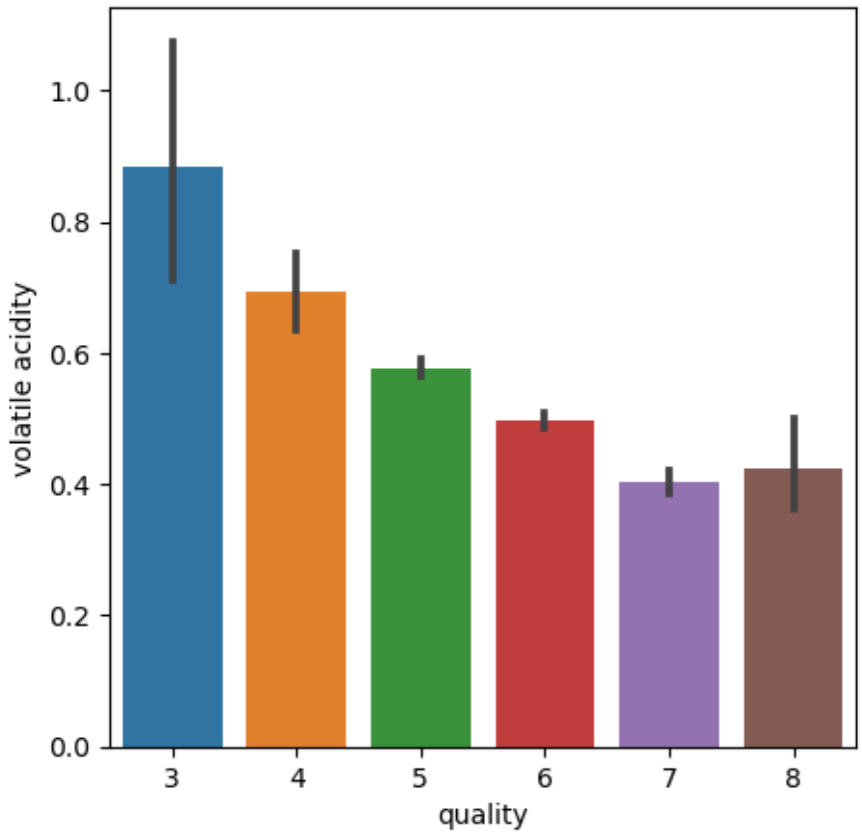
```
plt.title("Comparing volatile acidity with quality")
sns.barplot(x="quality",y="volatile acidity",data=df,palette="winter_r")
plt.show()
```

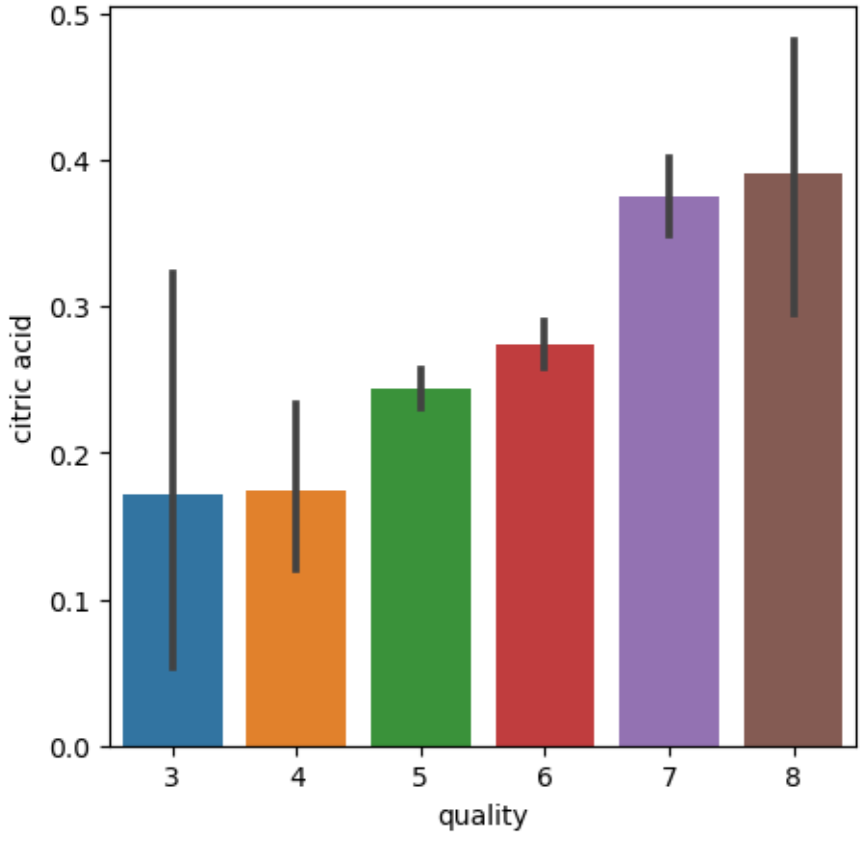

1.0
0.8
0.6
0.4
0.2
0.0
volatile acidity
3
4
5
6
7
8
quality

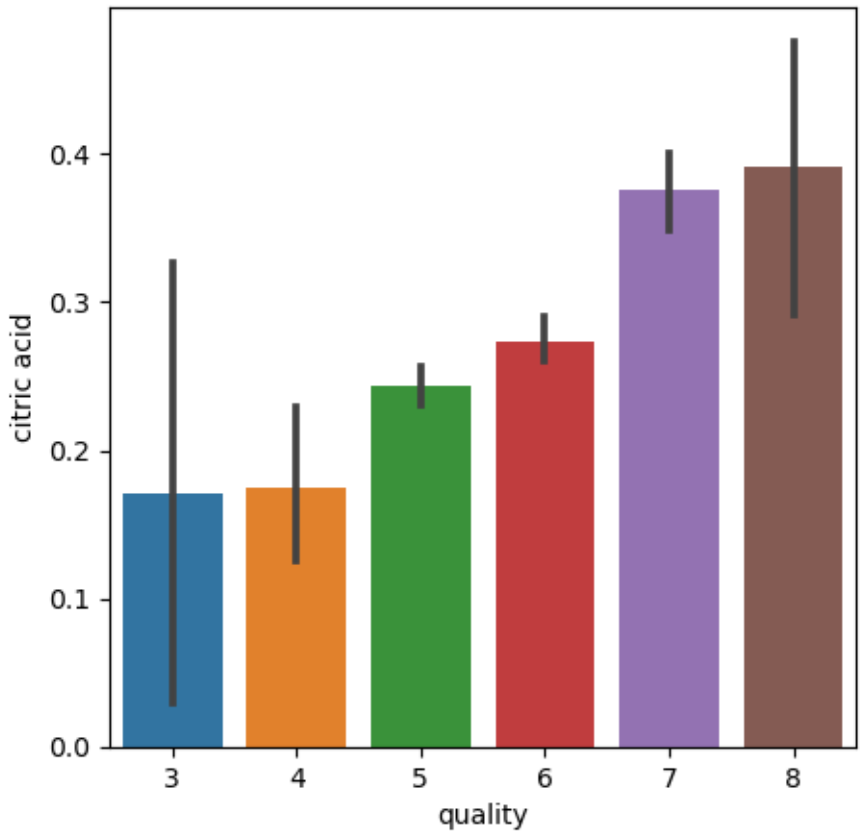

1.0
0.8
0.6
0.4
0.2
0.0
volatile acidity
3
4
5
6
7
8
quality

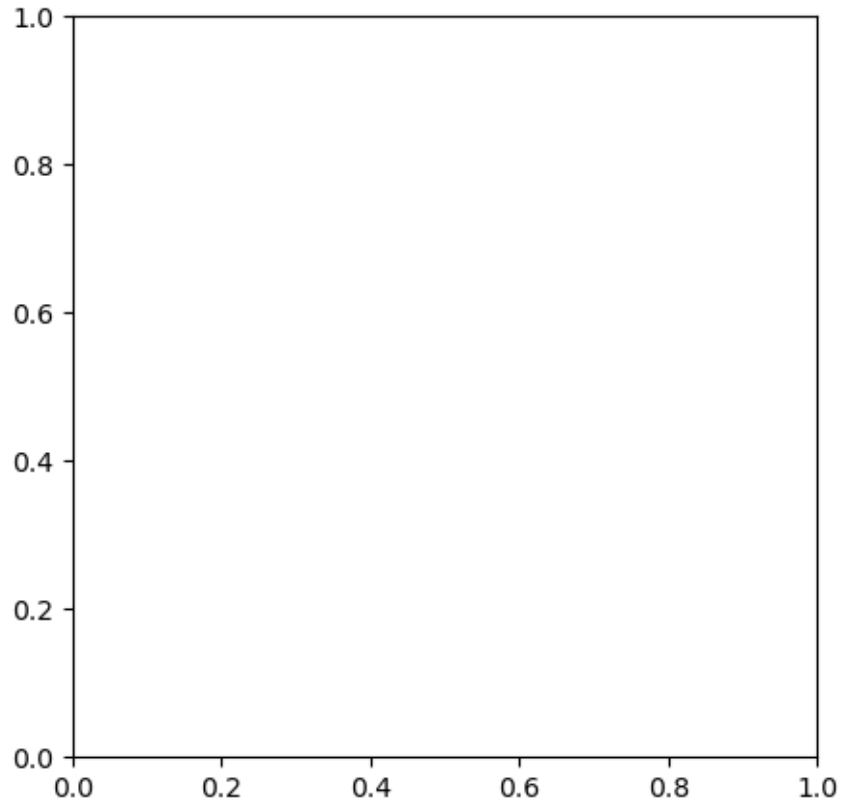

Comparing volatile acidity with quality
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

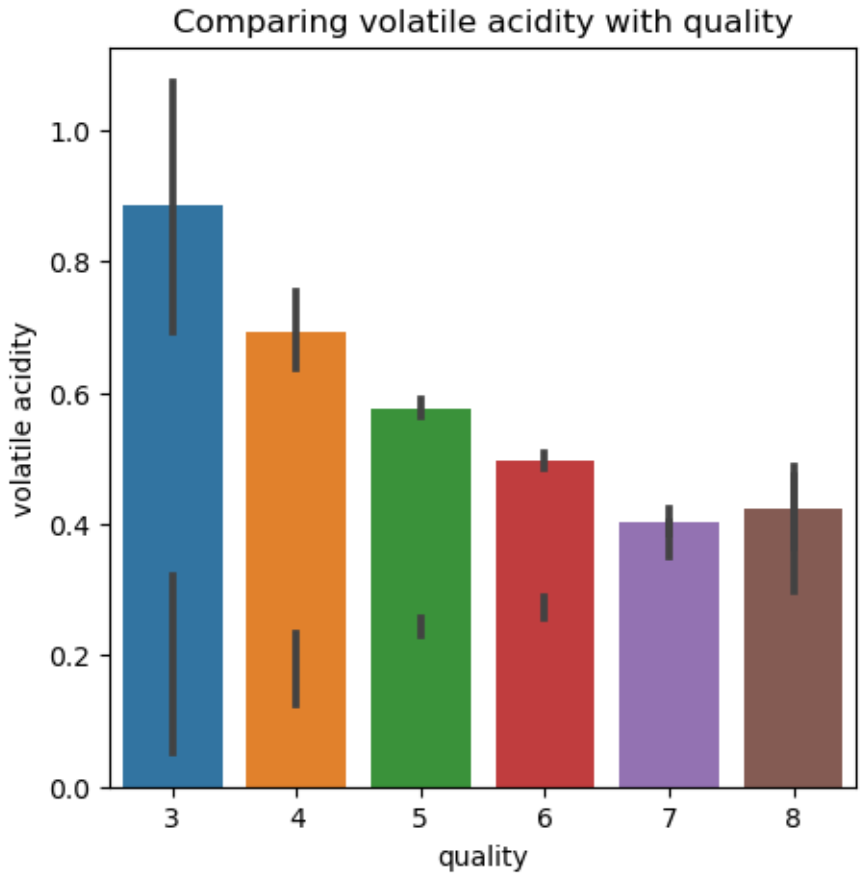

Comparing volatile acidity with quality
volatile acidity
1.0
0.8
0.6
0.4
0.2
0.0
3
4
5
6
7
8
quality

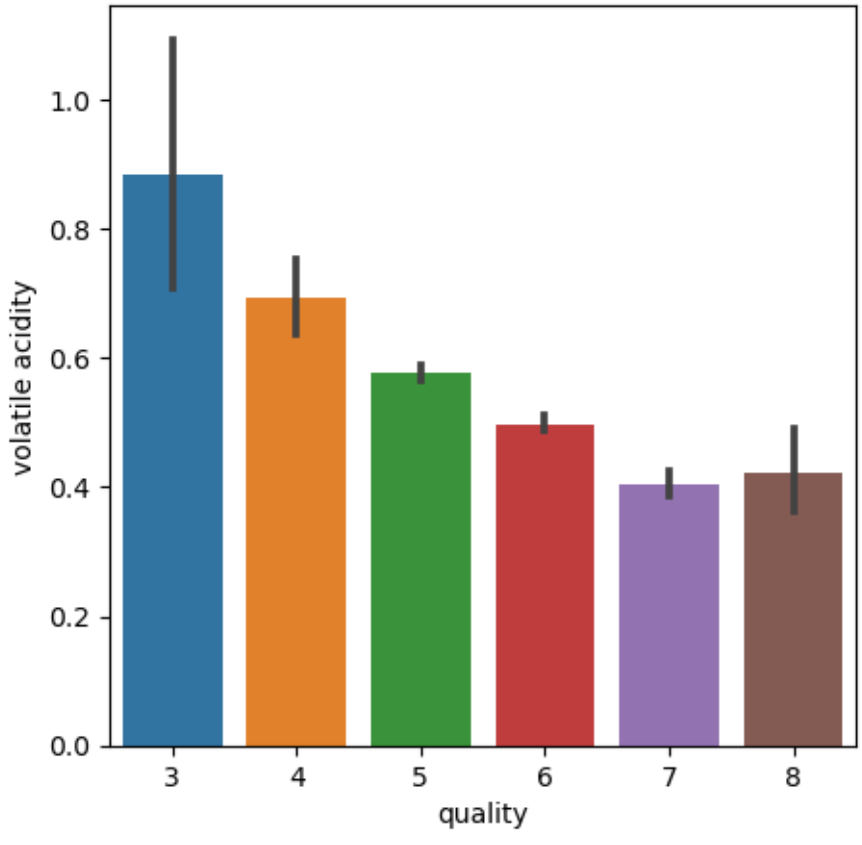

volatile acidity
1.0
0.8
0.6
0.4
0.2
0.0
3
4
5
6
7
8
quality

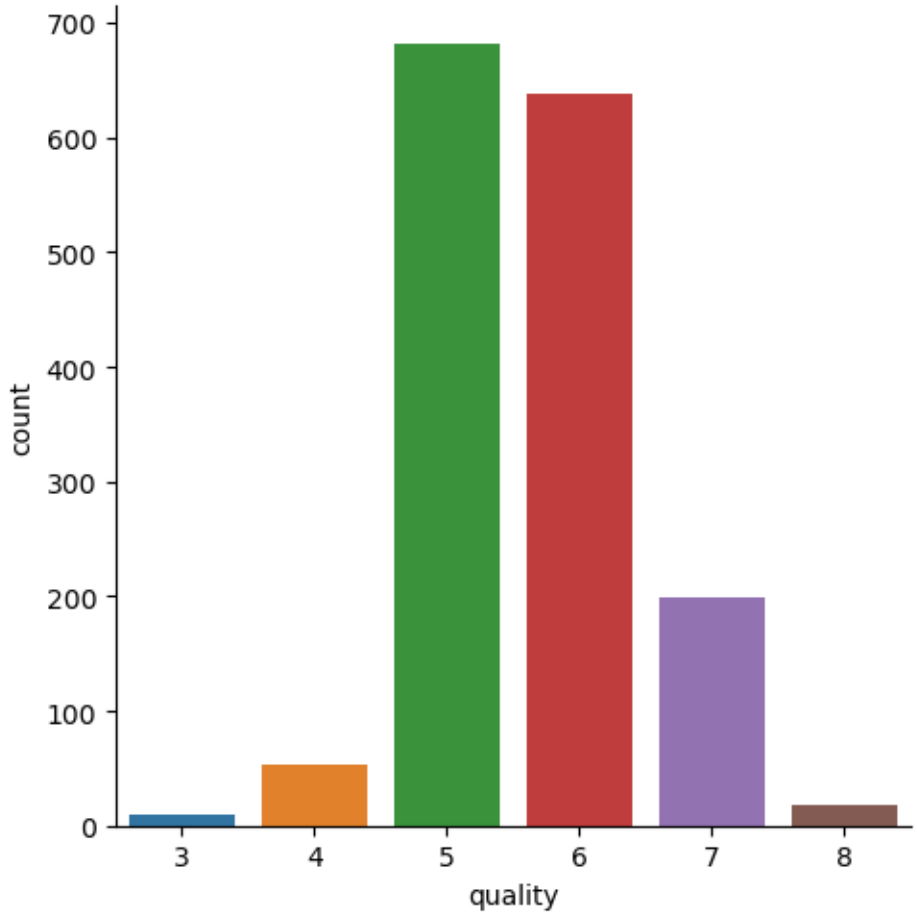

citric acid
0.5
0.4
0.3
0.2
0.1
0.0
3
4
5
6
7
8
quality

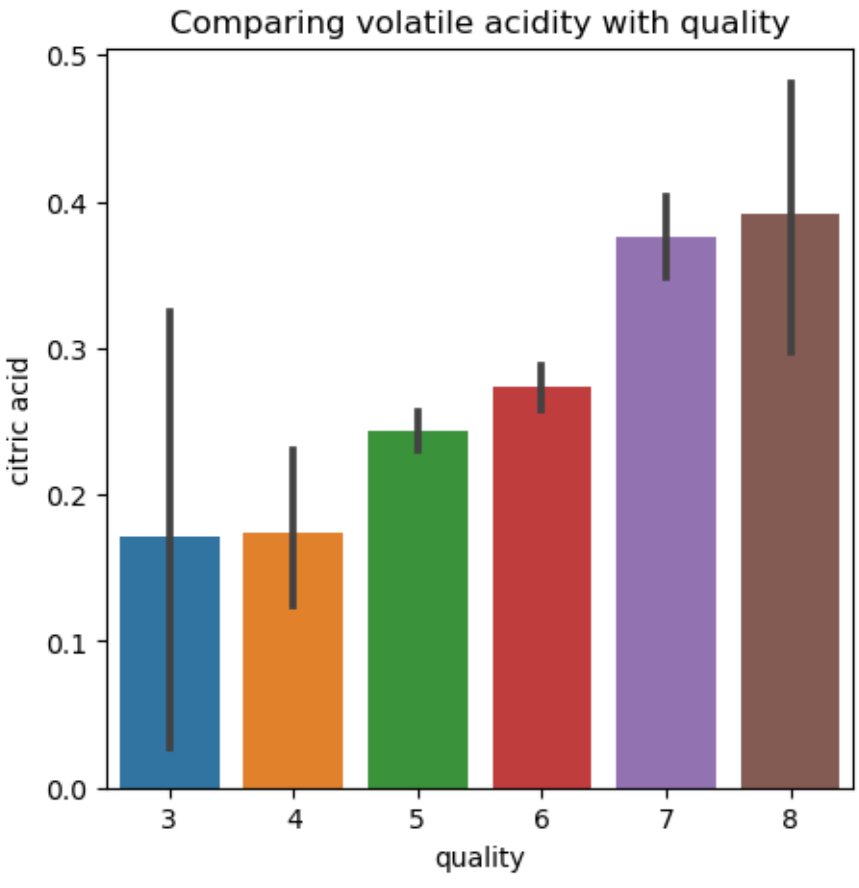

citric acid
0.0
0.1
0.2
0.3
0.4
3
4
5
6
7
8
quality

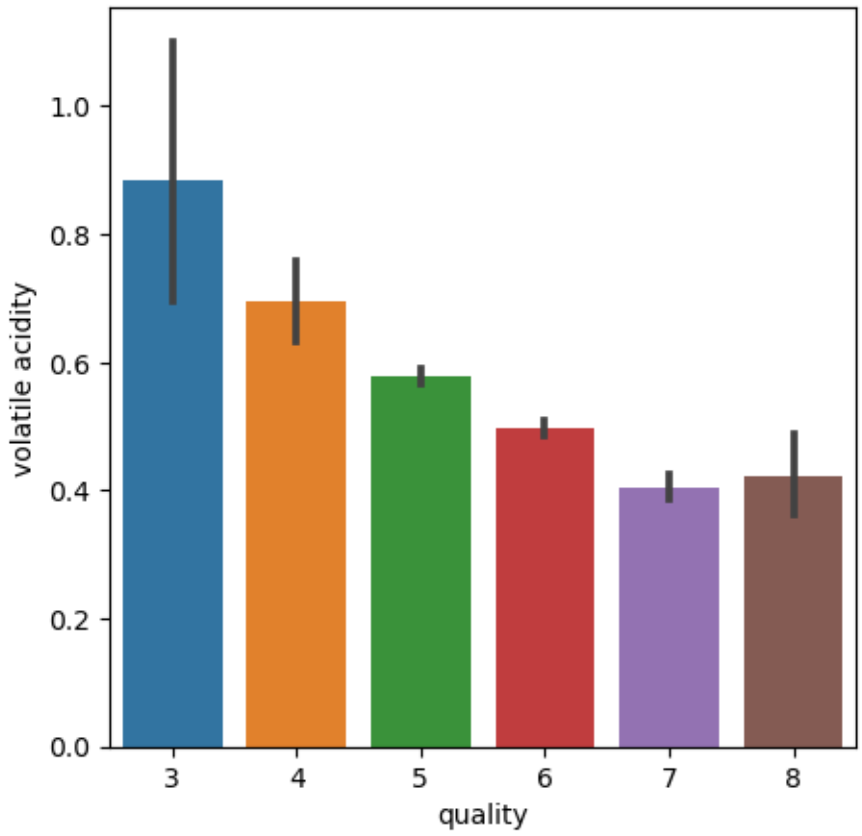

0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

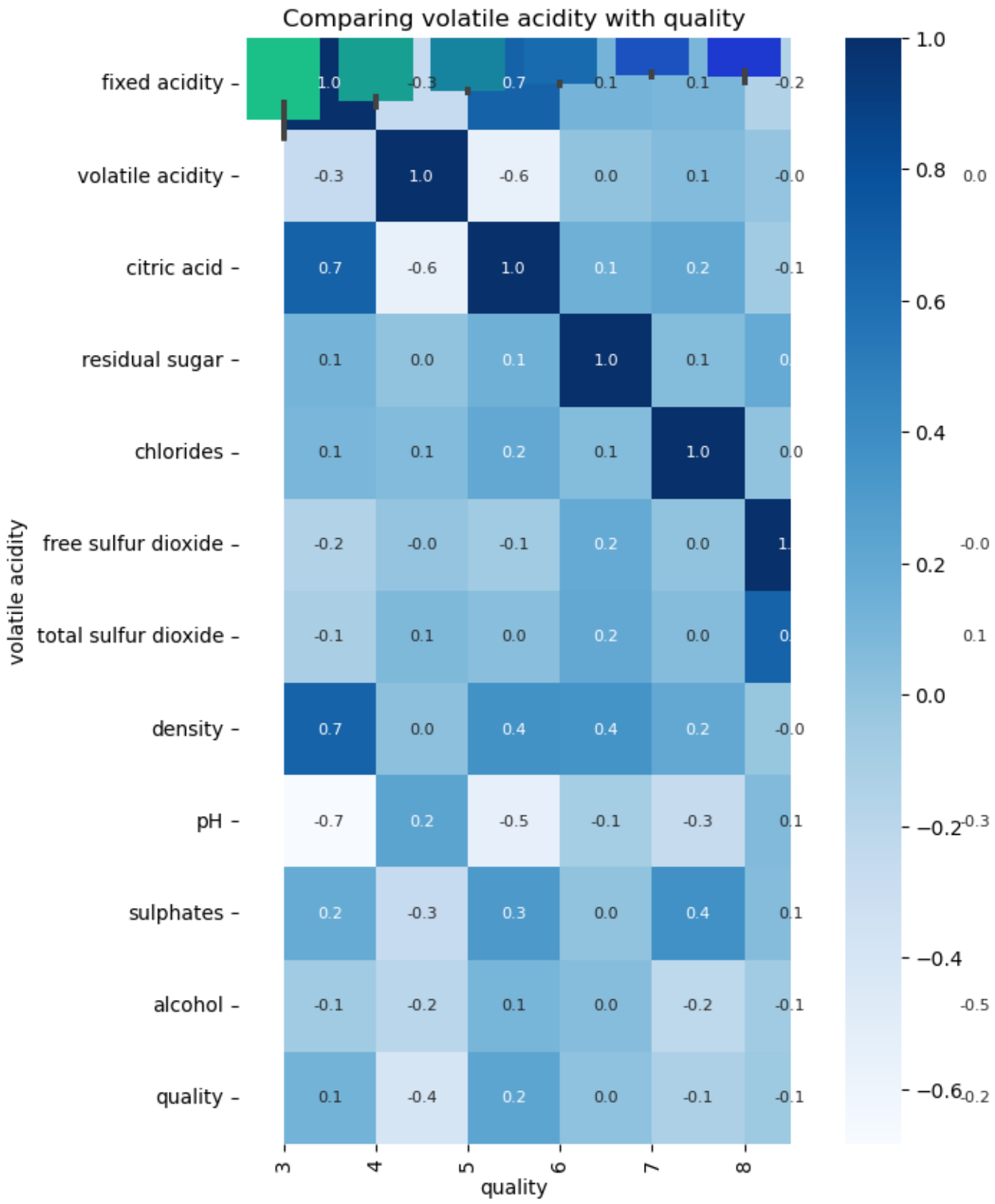

Comparing volatile acidity with quality
volatile acidity
1.0
0.8
0.6
0.4
0.2
0.0
3
4
5
6
7
8
quality


volatile acidity
1.0
0.8
0.6
0.4
0.2
0.0
3
4
5
6
7
8
quality


700
600
500
400
300
200
100
0
count
3
4
5
6
7
8
quality


Comparing volatile acidity with quality
0.5
0.4
0.3
0.2
0.1
0.0
citric acid
3
4
5
6
7
8
quality


volatile acidity
1.0
0.8
0.6
0.4
0.2
0.0
3
4
5
6
7
8
quality

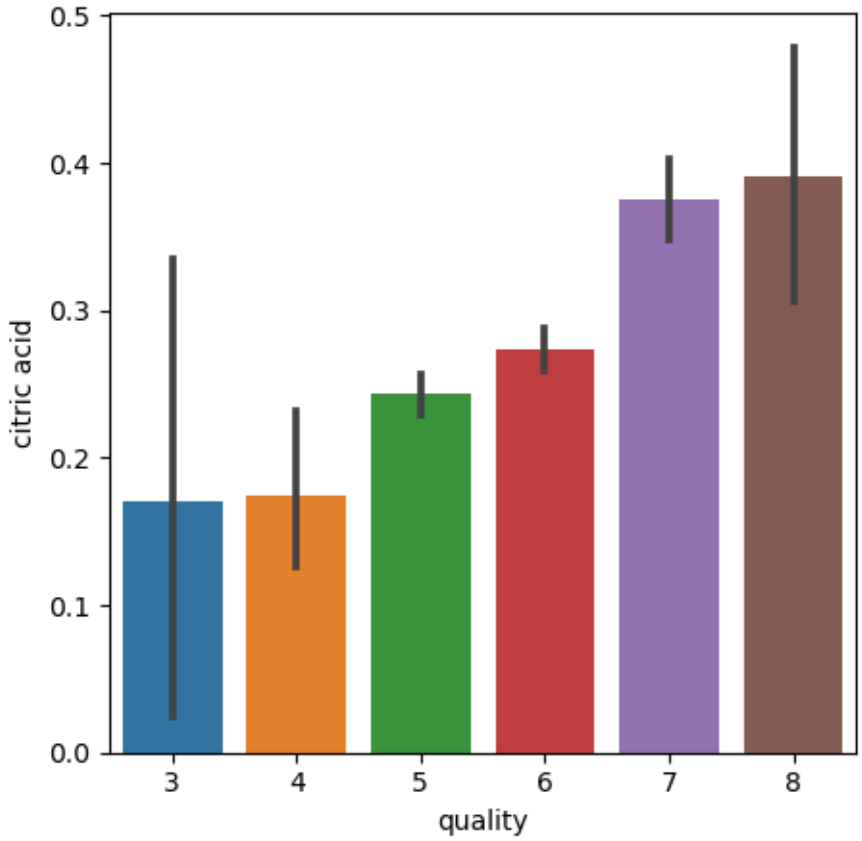


In [ ]:

```
#If volatite acidity is high than quality is worst
```

In [57]:

```
plot=plt.figure(figsize=(5,5))
sns.barplot(x="quality",y="citric acid",data = df)
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
#If Citric acid content is more than quality is also good
```

In [ ]:

```
#find correlation between data
```

In [41]:

```
correlation=df.corr()
```

In [ ]:

```
#using heatmap to understand correlation between columns
```

In [42]:

```
plt.figure(figsize=(10,10))
sns.heatmap(correlation,cbar=True,
square=True,fmt=".1f",annot=True,annot_kws={"size":8},cmap="Blues")
```

Out[42]:

```
<AxesSubplot:>
```

In [ ]:

```
#separate data and lable- Data processing
```

In [43]:

```
X=df.drop("quality",axis=1)
print(X)
```

	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]
```

In [ ]:
```
#Label Binarization
```

In [44]:
```
Y=df["quality"].apply(lambda y_value:1 if y_value>=7 else 0)
print(Y)
```

```
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
```

In [ ]:
```
#train test and split
```

In [46]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_sta
te=3)
```

In [47]:
```
print(Y.shape,Y_train.shape,Y_test.shape)
```

```
(1599,) (1279,) (320,)
```

In [ ]:
```
#Model training
#Random forest Classifier
```

In [48]:
```
model=RandomForestClassifier()
```

In [50]:
```
model.fit(X_train,Y_train)
```

Out[50]:
```
RandomForestClassifier()
```

In [ ]:
```
#Model evaluation-Accuracy score
#Accuracy on test data
```

In [52]:
```
X_test_prediction=model.predict(X_test)
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)
print("Accuracy:",test_data_accuracy)
```

```
Accuracy: 0.93125
```

In [ ]:
```
#Building a predictive system
```

In [53]:
```
input_data=(7.3,0.65,0.0,1.2,0.065,15.0,21.0,0.9946,3.39,0.47,10.0)
input_data_as_numpy_array=np.asarray(input_data)
```

```
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)
prediction=model.predict(input_data_reshaped)
print(prediction)
```

```
[1]
```

In [54]:

```
if (prediction[0]==1):
    print("Good Quality Wine")
else:
    print("Bad Quality Wine")
```

```
Good Quality Wine
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: