In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
wine_dataset = pd.read_csv(r"C:\Users\s323\Desktop\Gatherings\Data Science\Datasets
```

In [3]:
```
wine_dataset.head()
```

Out[3]:

	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

In [4]:
```
wine_dataset.shape
```

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

In [6]:
```
wine_dataset.columns
```

Out[6]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

In [8]:
```
wine_dataset.describe()
```

Out[8]:

	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	15.874922	46.467792
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324
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

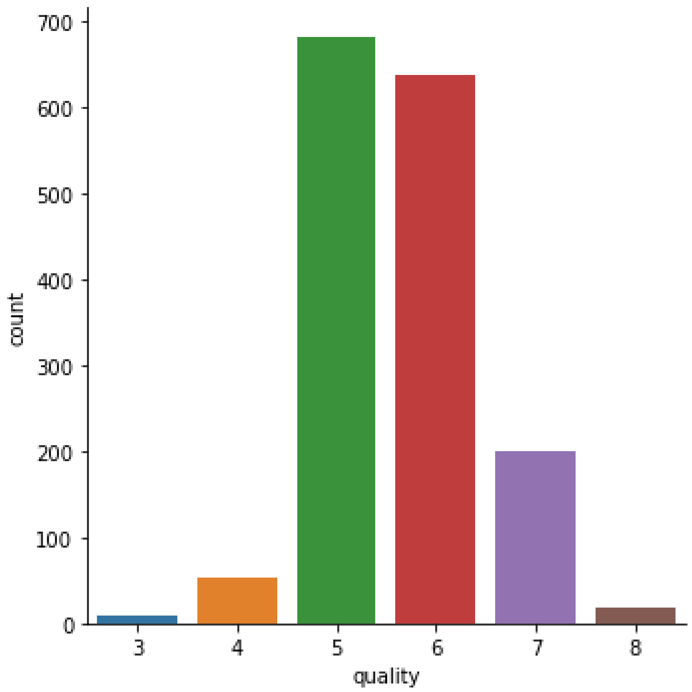
Data Analysis and Visualization

In [14]:
```
wine_dataset.isnull().sum()
```

Out[14]:
```
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
```

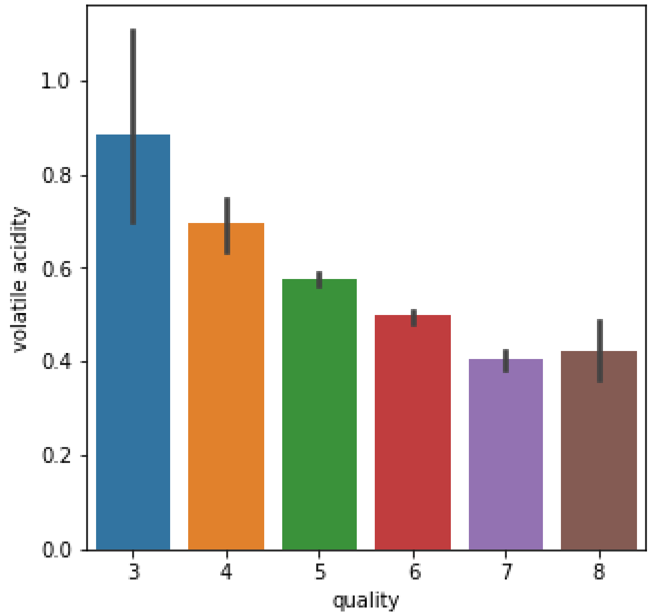
In [17]:
```
# No of values for each quality
sns.catplot(x='quality', data = wine_dataset, kind = 'count')
```

Out[17]: <seaborn.axisgrid.FacetGrid at 0x20817ccf0d0>



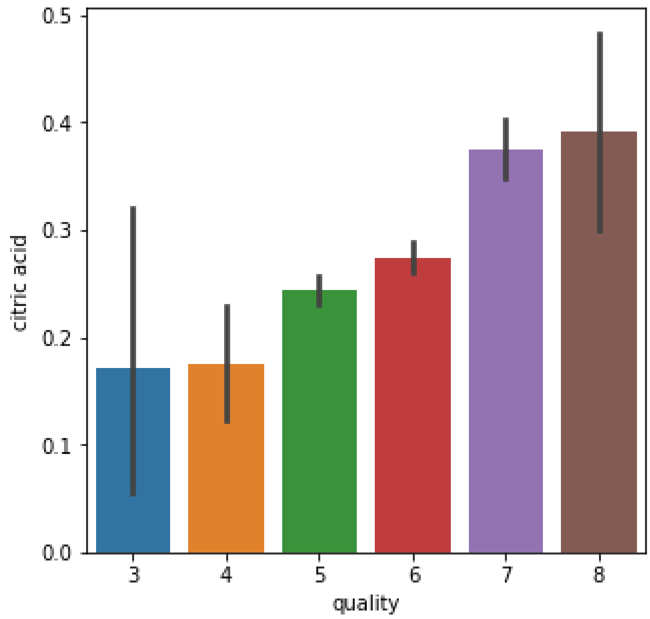
In [18]:
```
# volatile acidity vs quality
plot = plt.figure(figsize=(5,5))
sns.barplot(x="quality",y="volatile acidity",data=wine_dataset)
```

Out[18]: <AxesSubplot:xlabel='quality', ylabel='volatile acidity'>



In [19]:
```
# citric acid vs quality
plot = plt.figure(figsize=(5,5))
sns.barplot(x="quality",y="citric acid",data=wine_dataset)
```

Out[19]: <AxesSubplot:xlabel='quality', ylabel='citric acid'>



Co-relation

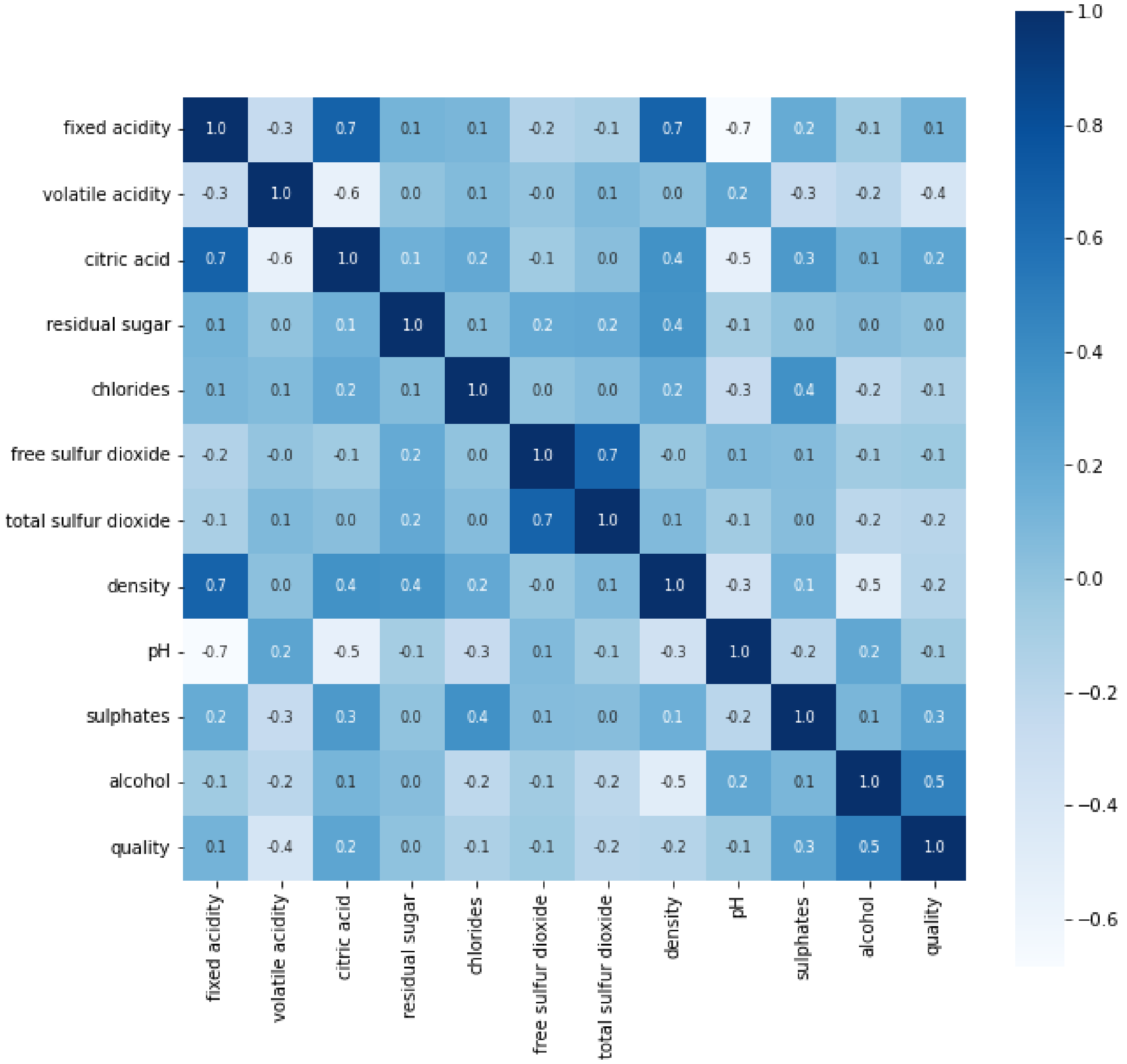
- Two types:

1. Positive- When they are directly prpotional
2. Negative- When they are inversely propotional

In [20]:
```
correlation = wine_dataset.corr()
```

In [22]:
```
plt.figure(figsize=(10,10))
sns.heatmap(correlation, cbar=True, square =True, fmt =".1f", annot=True, annot_kw
```

Out[22]: <AxesSubplot:>



- 12 columns vertical as well as horizontal- more dark means postively and highly corelated and less dark vice versa

Data Preprocessing

In [23]:
```
X = wine_dataset.drop("quality",axis =1)
```

In [24]:
```
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]
```

Label Binarization or Label Encoding

- Quality = 7 and 8 - Good - 1
- Quality = 6 or less than 6- Bad- 0

In [27]:
```
Y = wine_dataset["quality"].apply(lambda y_value:1 if y_value>=7 else 0)
```

In [28]:
```
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
```

Train and Test split

In [47]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.2, random_state=3
```

In [48]:
```
print(X.shape,x_train.shape,x_test.shape)
```

```
(1599, 11) (1279, 11) (320, 11)
```

Model Training

In [49]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [50]:
```
model= RandomForestClassifier()
```

In [51]:
```
model.fit(x_train,y_train)
```

Out[51]:
```
RandomForestClassifier()
```

Predictions

In [52]:
```
from sklearn.metrics import accuracy_score
```

In [53]:
```
train_model_predictions = model.predict(x_train)
train_data_accuracy = accuracy_score(train_model_predictions,y_train)
```

In [54]:
```
train_data_accuracy
```

Out[54]:
```
1.0
```

In [55]:
```
test_model_predictions = model.predict(x_test)
test_data_accuracy = accuracy_score(test_model_predictions,y_test)
```

In [56]:
```
test_data_accuracy
```

Out[56]:
```
0.921875
```

Building a new predictive system

In [58]:
```
input_data = (7.5,0.5,0.36,6.1,0.071,17.0,102.0,0.9978,3.35,0.8,10.5)
# changing input data into a numpy array
# since original is tuple
input_data_as_numpy_array = np.asarray(input_data)
```

In [59]:
```
# now we need to reshape the array as we only want value for one instance not whole
input_data_as_numpy_array_reshpe = input_data_as_numpy_array.reshape(1,-1)
```

In [60]:
```
prediction = model.predict(input_data_as_numpy_array_reshpe)
```

In [63]:
```
print(prediction)
```

```
[0]
```

In [64]:
```
if (prediction[0]==1):
  print('Good Quality Wine')
else:
  print('Bad Quality Wine')
```

```
Bad Quality Wine
```