WINE QUALITY PREDICTION

Objective:

To predict whether a wine sample is good or not.

Importing the Dependencies

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

The Dataset

In [3]:
```
wine_dataset=pd.read_csv(r'C:\Users\HP\Downloads\archive (3)\winequality-red.csv')
```

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

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

In [5]:
```
wine_dataset.head()
```

Out[5]:

	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

Looking for null values

In [6]:
```
wine_dataset.isnull().sum()
```

Out[6]:
```
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
```

This dataset has no null values

Statistical insights of the dataset

In [7]:
```
wine_dataset.describe()
```

Out[7]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
count	1599.000000	1599.000000	1599.000000	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	0.996747	3.311113	0.658149
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000

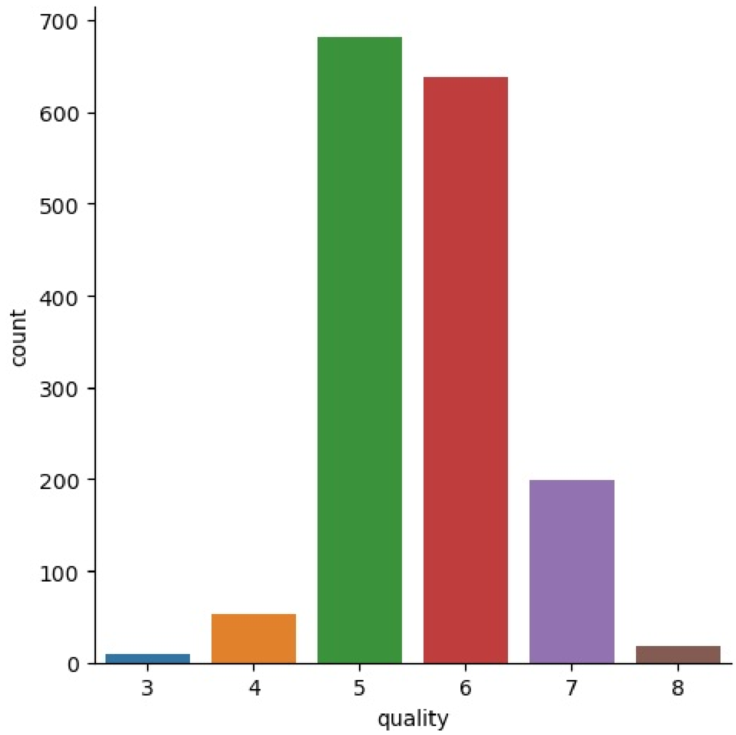
Number of values for each quality

In [10]:

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

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x1eb9fa28950>
```



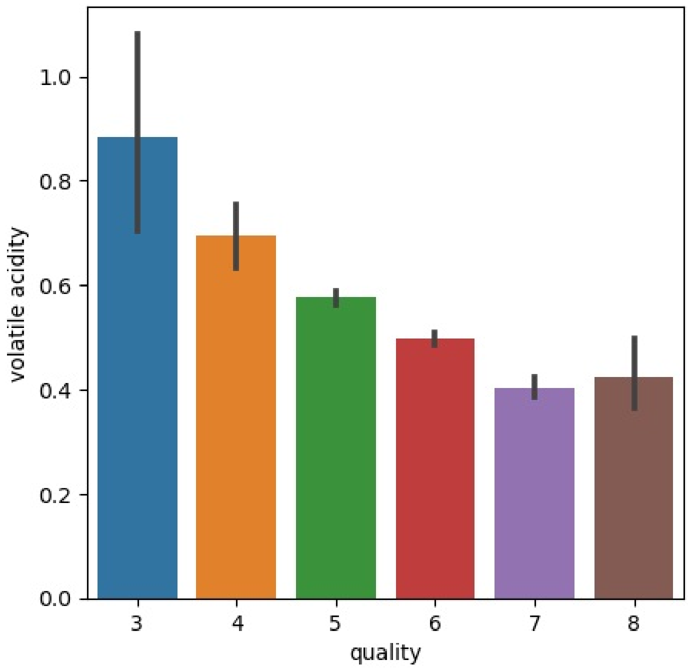
Volatile Acidity vs Quality

In [11]:

```
plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality',y='volatile acidity',data=wine_dataset)
```

Out[11]:

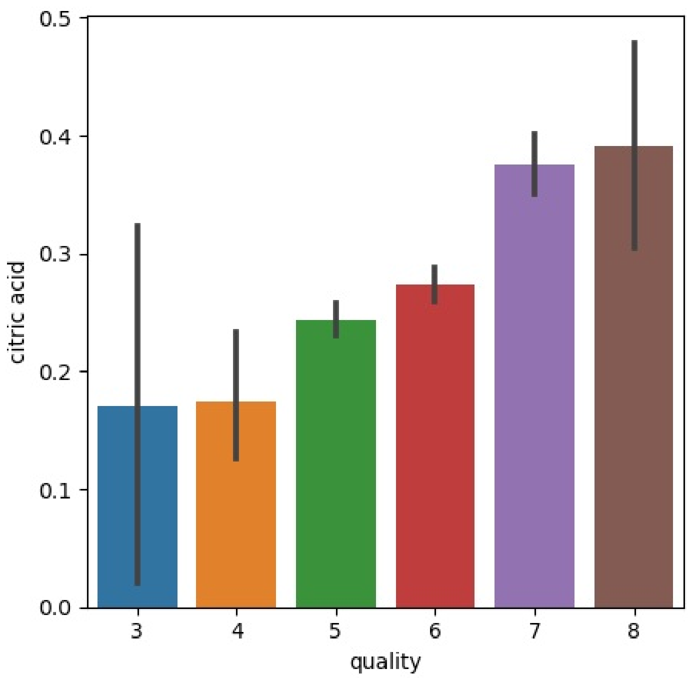
```
<Axes: xlabel='quality', ylabel='volatile acidity'>
```



Citric Acid vs Quality

In [12]:
```
plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality',y='citric acid',data=wine_dataset)
```

Out[12]: <Axes: xlabel='quality', ylabel='citric acid'>



Correlation:

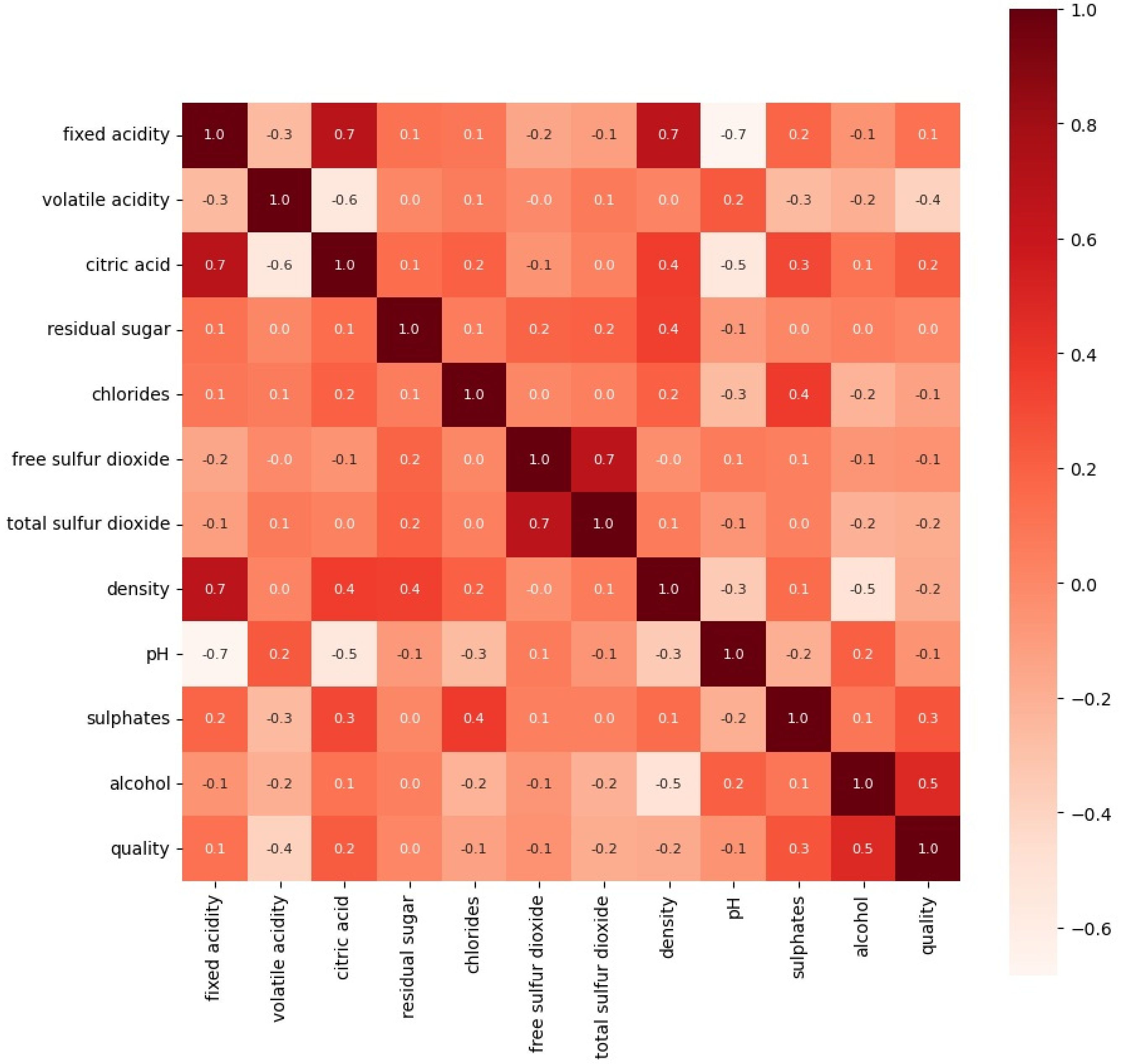
1.Positive Correlation

2. Negative Correlation

In [13]:
```
correlation=wine_dataset.corr()
```

In [17]:
```
plt.figure(figsize=(10,10))
sns.heatmap(correlation,cbar=True,square=True,fmt='.1f',annot=True,annot_kws={'size':8},cmap='Reds')
```

Out[17]: <Axes: >



Seperating the data and the label

In [18]:
```
X=wine_dataset.drop('quality',axis=1)
```

In [19]:
```
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

In [20]:
```
Y=wine_dataset['quality'].apply(lambda Y_value:1 if Y_value>=7 else 0)
```

In [21]:
```
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
```

Splitting Training data and Test data

In [22]:
```
X_train , X_test ,Y_train , Y_test = train_test_split(X,Y,test_size=0.2,random_state=2)
```

In [23]:
```
print(X.shape,X_train.shape,X_test.shape)
```

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

Model Training

Random Forest Classifier

In [24]:
```
model=RandomForestClassifier()
```

In [25]:
```
model.fit(X_train,Y_train)
```

Out[25]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

Model Evaluation

Accuracy Score

Accuracy on test data

In [26]:
```
X_test_prediction=model.predict(X_test)
```

In [27]:
```
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)
```

In [28]:
```
print('Accuracy:',test_data_accuracy)
```

```
Accuracy: 0.934375
```

Building a predictive system

In [29]:
```
input_data=(7.8,0.58,0.02,2.0,0.073,9.0,18.0,0.9968,3.36,0.57,9.5)
```

Changing the input data to a numpy array

In [30]:
```
input_data_as_numpy_array=np.asarray(input_data)
```

Reshaping the data as we are predicting the label for only one instance

In [31]:
```
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)
```

The prediction

In [32]:
```
prediction=model.predict(input_data_reshaped)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature name
s, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

In [33]:
```
print(prediction)
```

```
[1]
```

In [34]:
```
if (prediction[0]==1):
    print("It is a good quality wine.")
else:
    print("It is a poor quality wine.")
```

```
It is a good quality wine.
```

In [ ]: