import the required libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

In [2]:
```
data = pd.read_csv('/home/tamanna/Downloads/WineQT.csv')
data
```

Out[2]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56
...	...	...	...	...	...	...	...	...	...	..
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71

1143 rows × 13 columns

data understanding

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

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	al
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	
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	

In [4]:
```
# number or rows
data.shape[0]
```

Out[4]:
```
1143
```

In [5]:
```
# number of columns
data.shape[1]
```

Out[5]:
```
13
```

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64  
 12  Id                    1143 non-null   int64  
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

In [7]:
```
# statistical measures
data.describe()
```

Out[7]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide
count	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000
mean	8.311111	0.531339	0.268364	2.532152	0.086933	15.615486
std	1.747595	0.179633	0.196686	1.355917	0.047267	10.250486
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000
25%	7.100000	0.392500	0.090000	1.900000	0.070000	7.000000
50%	7.900000	0.520000	0.250000	2.200000	0.079000	13.000000
75%	9.100000	0.640000	0.420000	2.600000	0.090000	21.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	68.000000

In [8]:
```
data.columns
```

Out[8]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual suga
r',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'densit
y',
       'pH', 'sulphates', 'alcohol', 'quality', 'Id'],
      dtype='object')
```

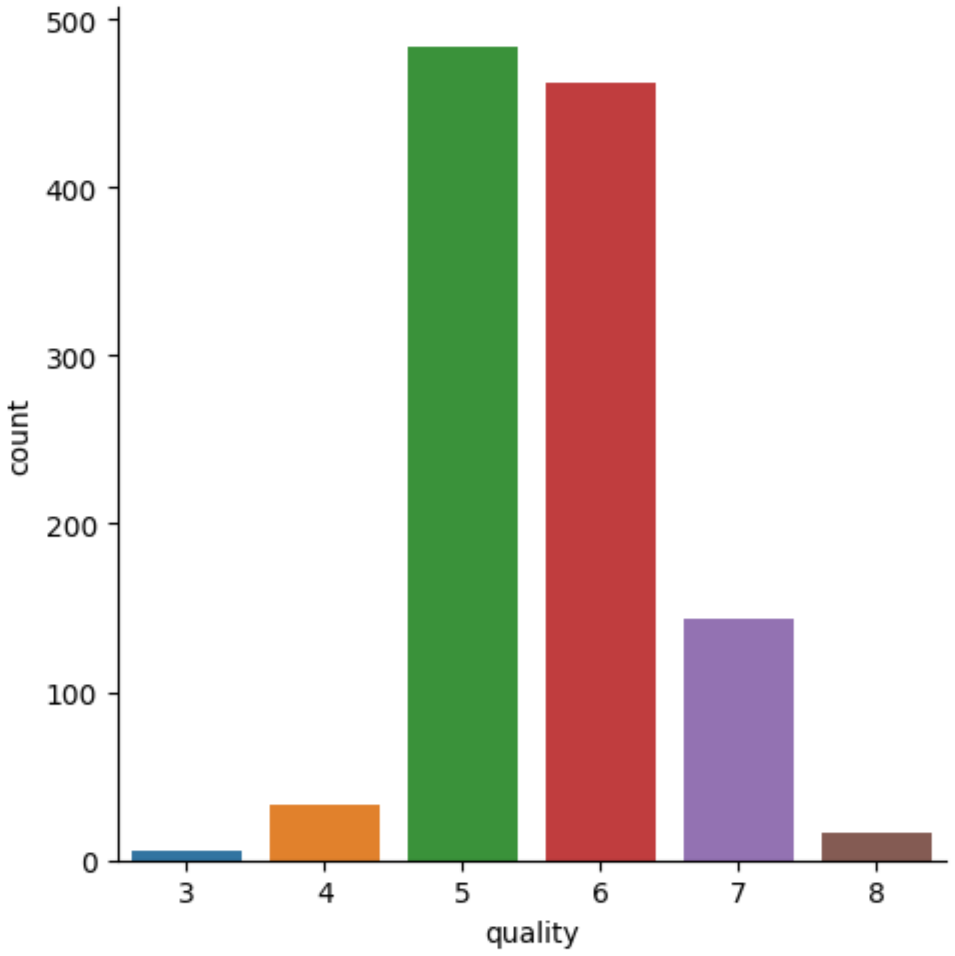
In [9]:
```
data.isnull().sum()
```

Out[9]:
```
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
Id                      0
dtype: int64
```

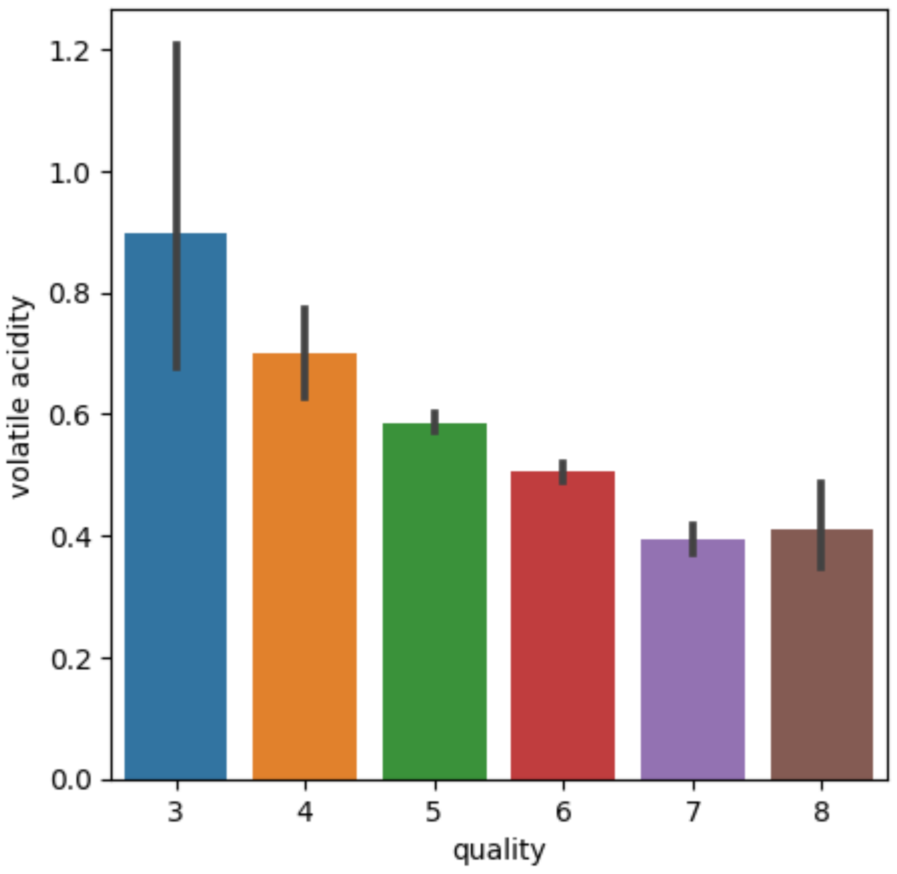
data visualization

In [10]:
```
# number of values for each quality
sns.catplot(x = 'quality', data = data, kind = 'count')
```

Out[10]:
```
<seaborn.axisgrid.FacetGrid at 0x7f440751b710>
```



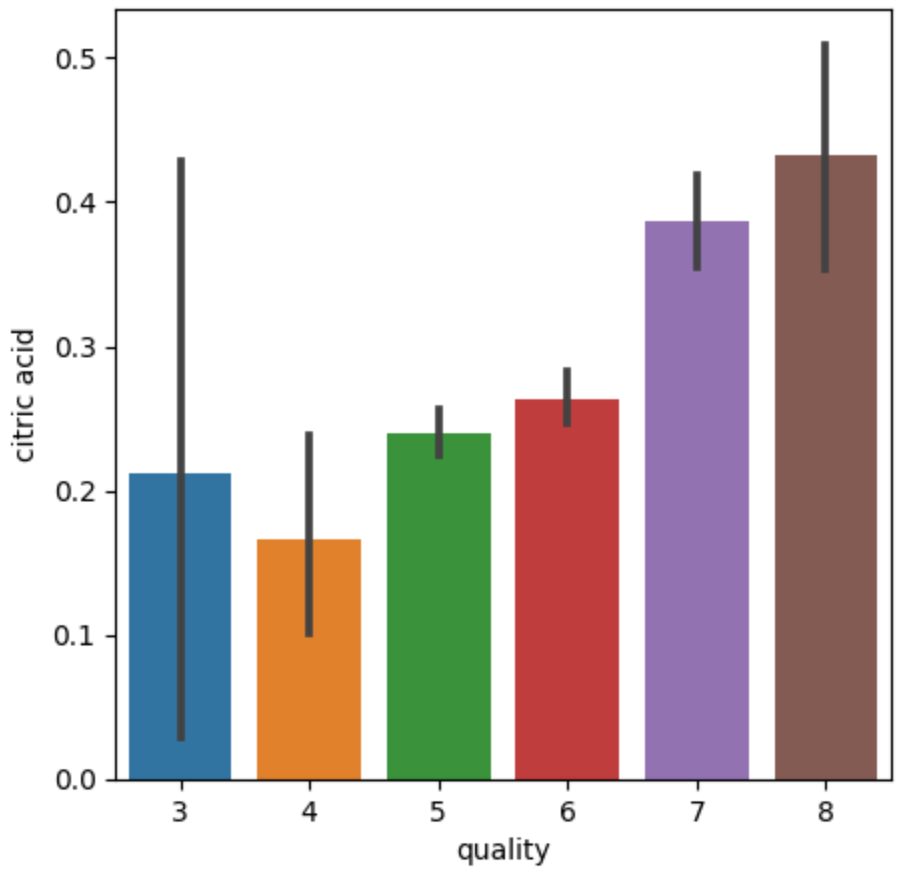
```
In [11]: #volatile aidity vs quality
         plot = plt.figure(figsize = (5,5))
         sns.barplot(x= 'quality' , y = 'volatile acidity', data = data)

Out[11]: <Axes: xlabel='quality', ylabel='volatile acidity'>
```



both are inversly proportional

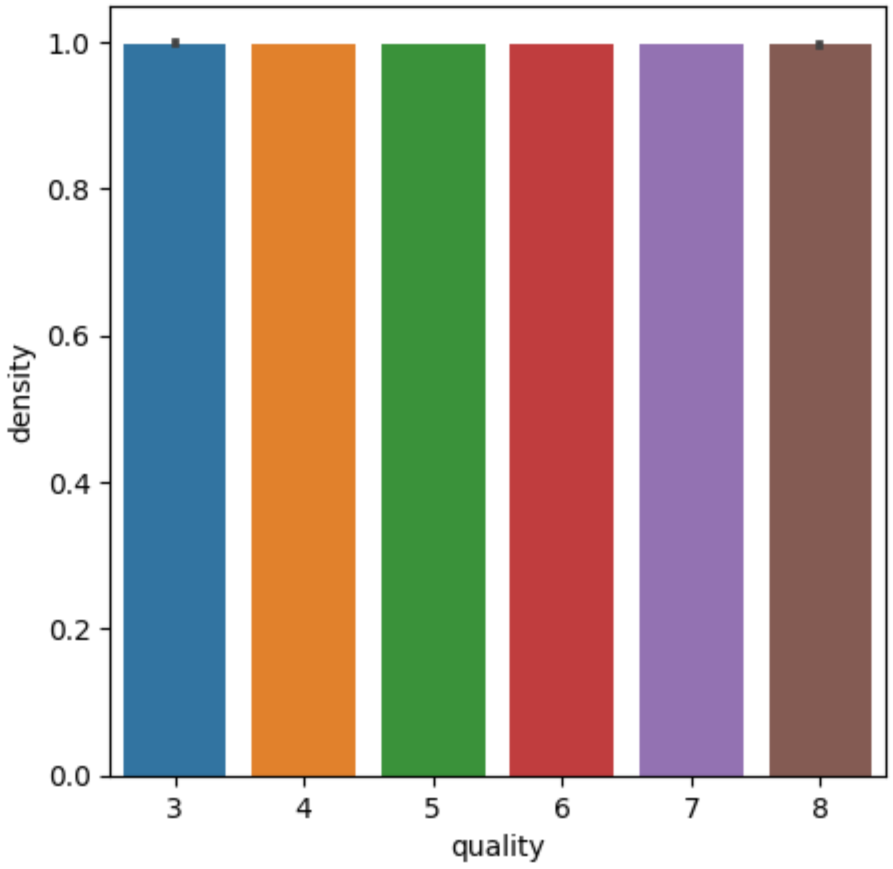
In [12]:
```
#citric acid vs quality
plot = plt.figure(figsize = (5,5))
sns.barplot(x= 'quality' , y = 'citric acid', data = data)
```

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



In [13]:
```
#density vs quality
plot = plt.figure(figsize = (5,5))
sns.barplot(x= 'quality' , y = 'density', data = data)
```

Out[13]:
```
<Axes: xlabel='quality', ylabel='density'>
```



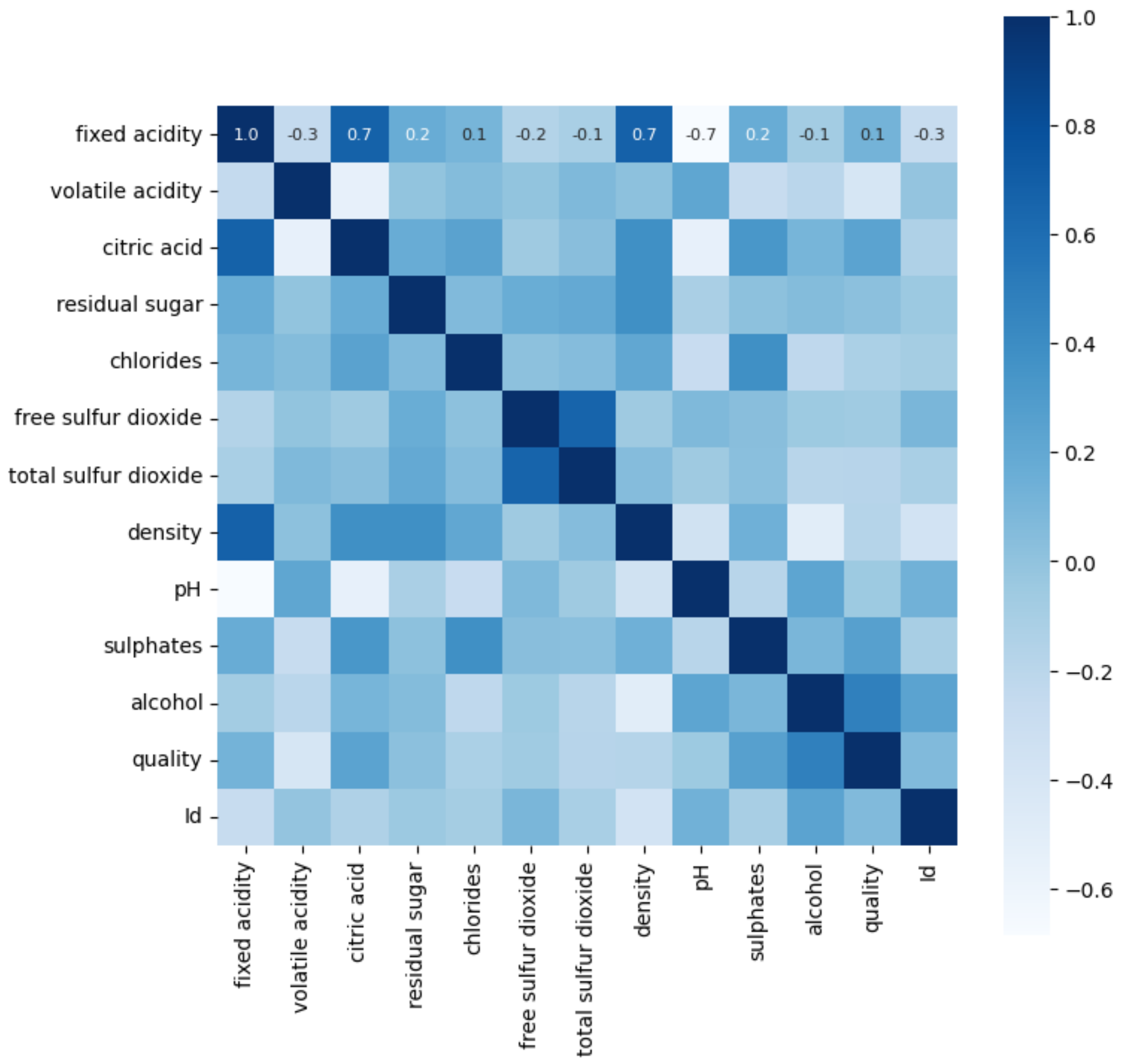
both are directly proportional

Correlation

```
In [14]: correlation = data.corr()
```

```
In [15]: # constructing a heat map to understand the correlation between the columns
         plt.figure(figsize=(8,8))
         sns.heatmap(correlation, cbar=True, square = True, fmt = '.1f', annot = True
```

```
Out[15]: <Axes: >
```



```
In [16]: # separating the data and lables
         x = data.drop('quality', axis = 1)
```

```
In [17]: print(x)
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar  chloride
s  \
0               7.4             0.700         0.00             1.9      0.07
6
1               7.8             0.880         0.00             2.6      0.09
8
2               7.8             0.760         0.04             2.3      0.09
2
3              11.2             0.280         0.56             1.9      0.07
5
4               7.4             0.700         0.00             1.9      0.07
6
...             ...               ...          ...             ...        ..
.
1138            6.3             0.510         0.13             2.3      0.07
6
1139            6.8             0.620         0.08             1.9      0.06
8
1140            6.2             0.600         0.08             2.0      0.09
0
1141            5.9             0.550         0.10             2.2      0.06
2
1142            5.9             0.645         0.12             2.0      0.07
5

      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
...                   ...                   ...      ...   ...        ...
1138                 29.0                  40.0  0.99574  3.42       0.75
1139                 28.0                  38.0  0.99651  3.42       0.82
1140                 32.0                  44.0  0.99490  3.45       0.58
1141                 39.0                  51.0  0.99512  3.52       0.76
1142                 32.0                  44.0  0.99547  3.57       0.71

      alcohol    Id
0         9.4     0
1         9.8     1
2         9.8     2
3         9.8     3
4         9.4     4
...       ...   ...
1138     11.0  1592
1139      9.5  1593
1140     10.5  1594
1141     11.2  1595
1142     10.2  1597

[1143 rows x 12 columns]
```

In [18]:
```
#label binarization
y = data['quality'].apply(lambda y_value: 1 if y_value>= 7 else 0)
```

```
In [19]: print(y)
```

```
0       0
1       0
2       0
3       0
4       0
       ..
1138    0
1139    0
1140    0
1141    0
1142    0
Name: quality, Length: 1143, dtype: int64
```

splitting the data into training and test data set

```
In [20]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, r
```

```
In [21]: print(x.shape, x_train.shape, x_test.shape)
```

```
(1143, 12) (914, 12) (229, 12)
```

Model Training

Random Forest

```
In [22]: model = RandomForestClassifier()
```

```
In [23]: model.fit(x_train, y_train)
```

```
Out[23]: ▾ RandomForestClassifier

         RandomForestClassifier()
```

Model Evaluation

Accuracy Score

```
In [24]: # accuracy on train data
         x_train_prediction = model.predict(x_train)
         train_data_accuracy = accuracy_score(x_train_prediction, y_train)
```

```
In [25]: print('Accuracy score on training data: ', train_data_accuracy)
```

```
Accuracy score on training data:  1.0
```

```
In [26]: # accuracy on test data
         x_test_prediction = model.predict(x_test)
         test_data_accuracy = accuracy_score(x_test_prediction, y_test)
```

In [27]:
```
print('Accuracy score on test data :', test_data_accuracy)
```

```
Accuracy score on test data : 0.9170305676855895
```

In [28]:
```
# checking the model accuracy on given data
input_data = (11.2,0.28,0.56,1.9,0.075,17.0,60.0,0.9980,3.16,0.58,9.8,3)

# changing the input data to a numpy array
input_data_as_numpy_arr = np.asarray(input_data)

# reshape the data as we are predicting the label for only one instance
input_data_reshaped = input_data_as_numpy_arr.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)

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

```
[0]
Bad Quality Wine
```