In [1]:
```
import numpy as np #to create numpy arrays
import pandas as pd #to create pandas dataframe
import matplotlib.pyplot as plt #for making plots and graphs
import seaborn as sns #for data visualization
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df=pd.read_csv("winequality-red.csv")
```

In [3]:
```
df.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]:
```
df.shape
```

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

In [5]:
```
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 [6]:
```
df.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
```

As we can see there are no missing values.

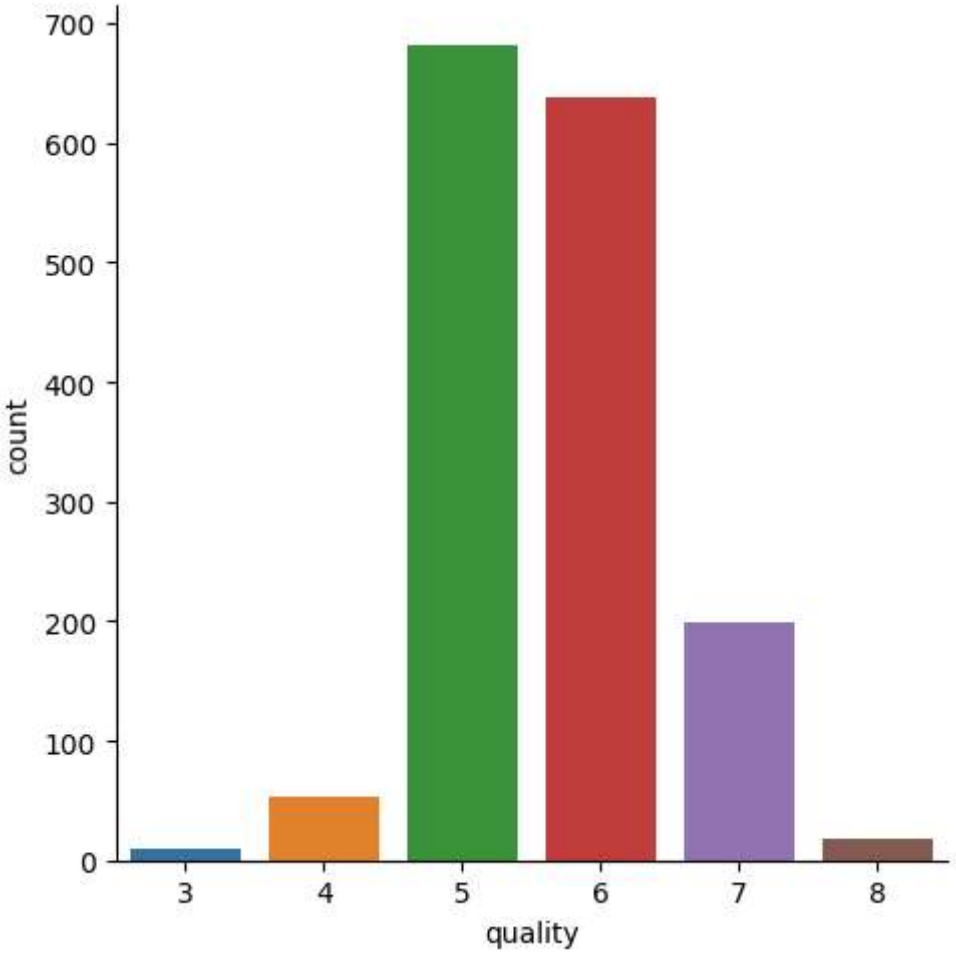
In [7]:
```
df.describe()
```

Out[7]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sul dioxi
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.0000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.4677
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.8953
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.0000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.0000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.0000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.0000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.0000

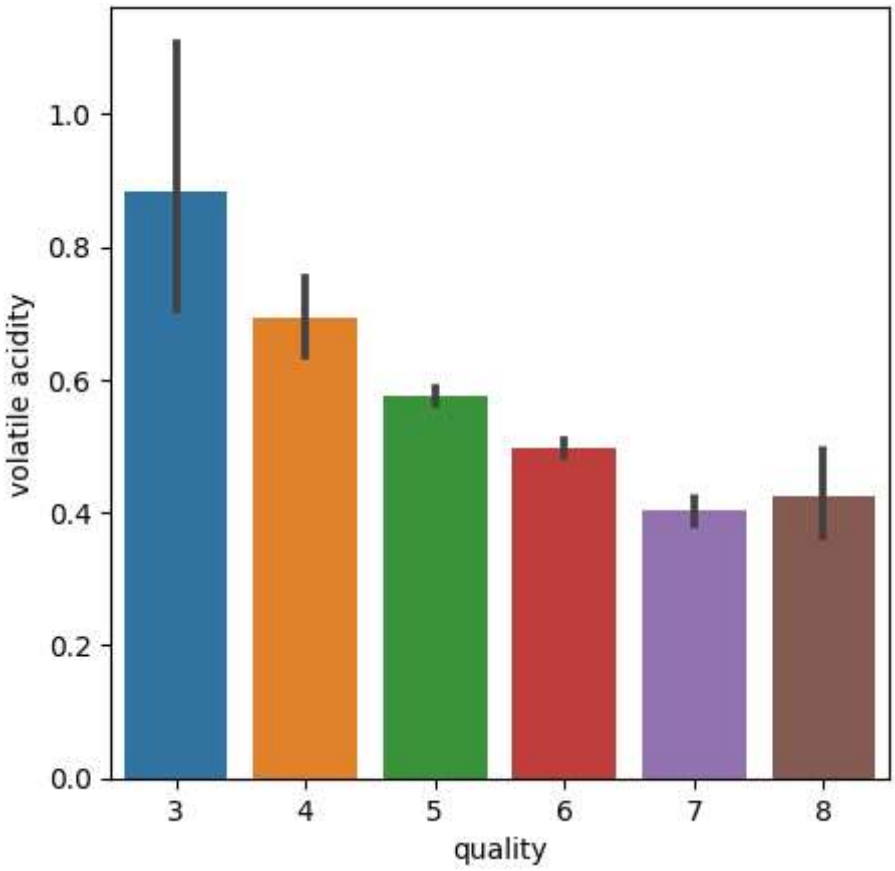
In [8]:
```
sns.catplot(x="quality", data=df,kind="count")
```

Out[8]:
```
<seaborn.axisgrid.FacetGrid at 0x250c627a110>
```



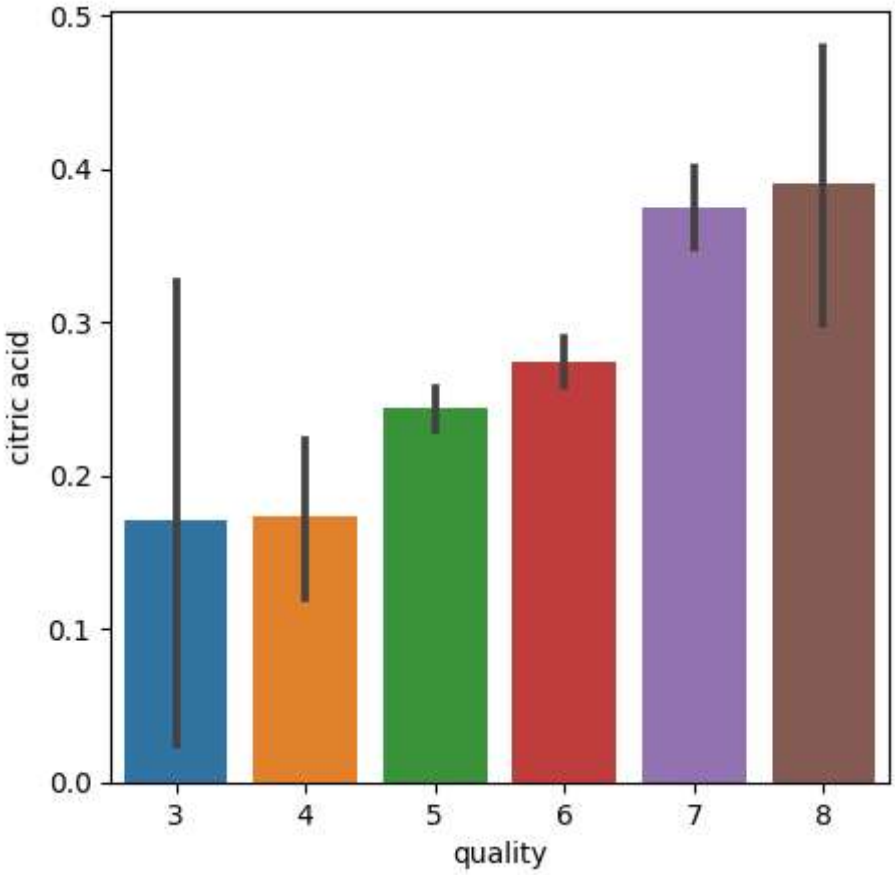
In [9]:
```
plot = plt.figure(figsize=(5,5))
sns.barplot(x="quality", y="volatile acidity", data=df)
```

Out[9]: <AxesSubplot: xlabel='quality', ylabel='volatile acidity'>

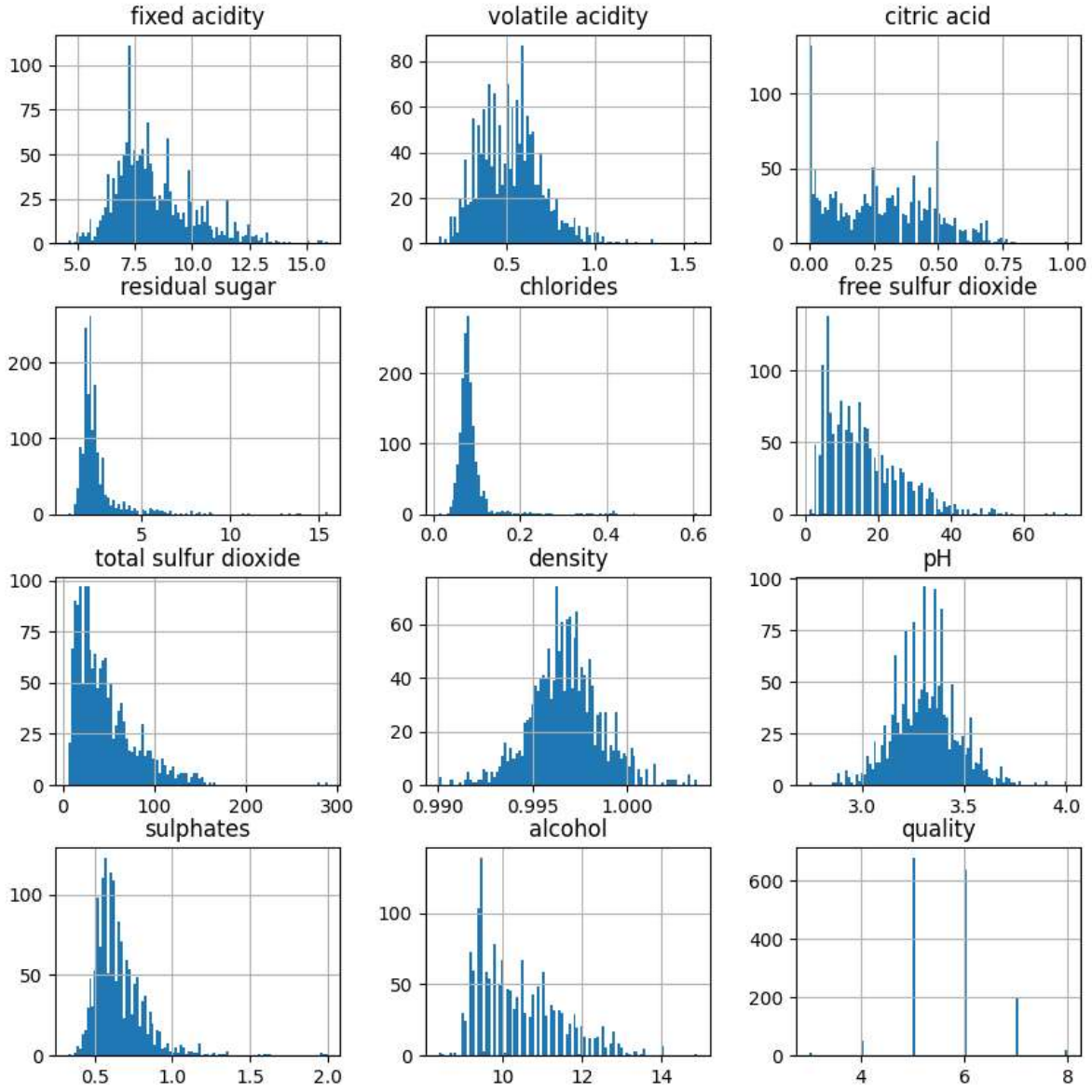


In [10]:
```
plot = plt.figure(figsize=(5,5))
sns.barplot(x="quality", y="citric acid", data=df)
```

Out[10]: <AxesSubplot: xlabel='quality', ylabel='citric acid'>



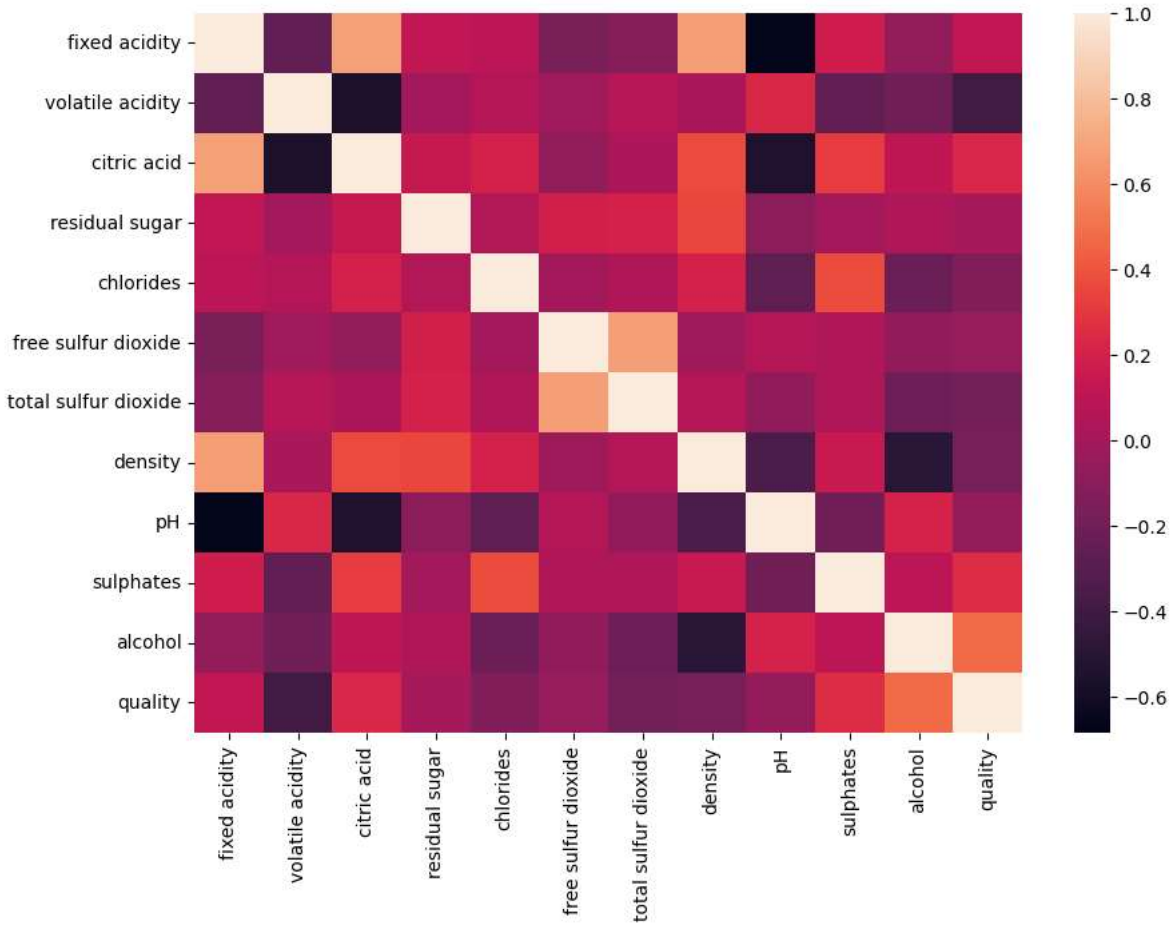
In [11]:
```
df.hist(bins=100, figsize=(10,10))
plt.show()
```



In [12]:
```
correlation = df.corr()
```

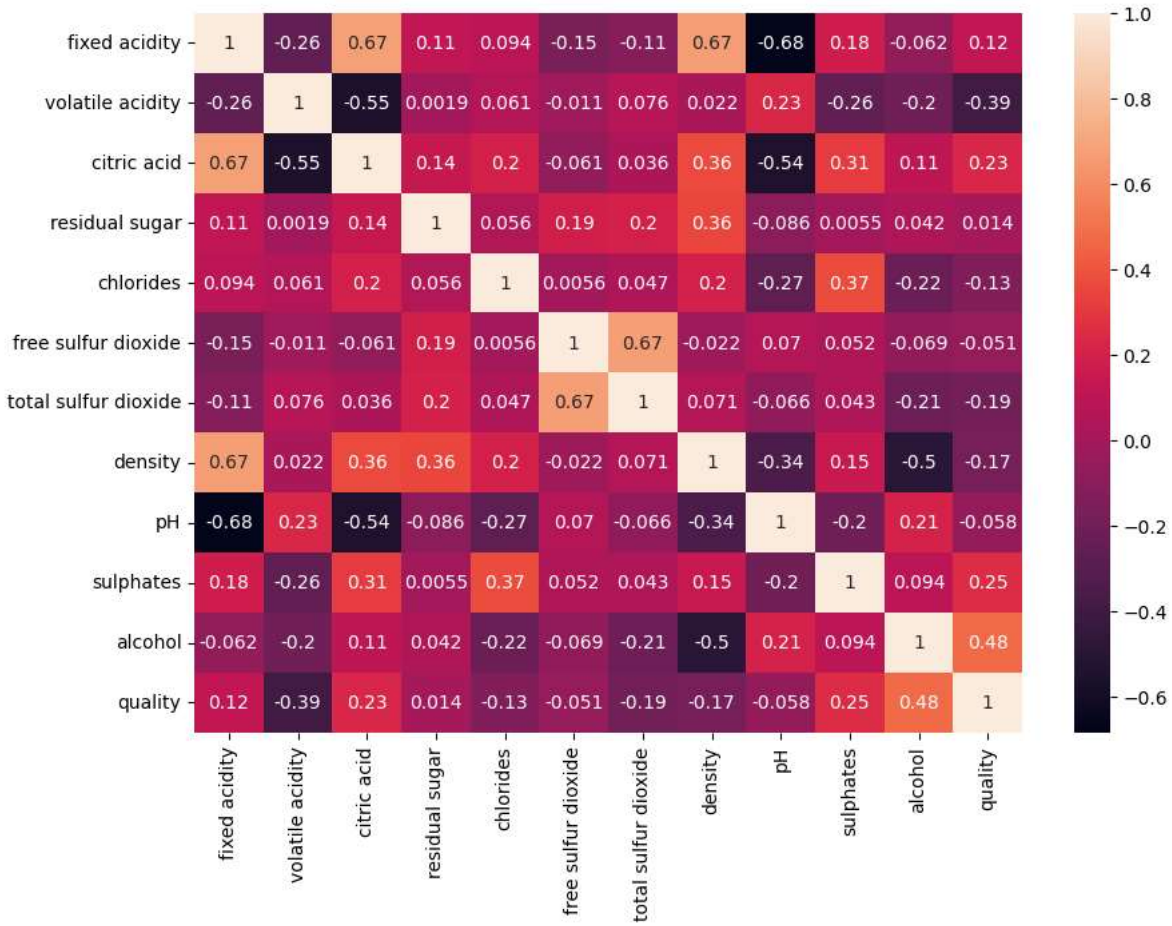
In [13]:
```
plt.figure(figsize=(10,7))
sns.heatmap(correlation, annot=False)
```

Out[13]: <AxesSubplot: >



In [14]:
```
plt.figure(figsize=(10,7))
sns.heatmap(correlation, annot=True)
```

Out[14]: <AxesSubplot: >



In [15]:
```
#printing correlation values
df.corr()["quality"].sort_values()
```

Out[15]:
```
volatile acidity       -0.390558
total sulfur dioxide   -0.185100
density                -0.174919
chlorides              -0.128907
pH                     -0.057731
free sulfur dioxide    -0.050656
residual sugar          0.013732
fixed acidity           0.124052
citric acid             0.226373
sulphates               0.251397
alcohol                 0.476166
quality                 1.000000
Name: quality, dtype: float64
```

alcohol' has higher correlation with = quality

data pre-processing

In [16]:
```
#separating the features and label
X= df.drop("quality", axis=1)
```

In [17]:
```
print(X.head(2))
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70          0.0             1.9      0.076
1            7.8              0.88          0.0             2.6      0.098

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68

   alcohol
0      9.4
1      9.8
```

In [18]:
```
Y = df["quality"].apply(lambda y_value:1 if y_value>=6.5 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
```

So here we have classified the different wine quality ratings to 1 and 0 --GOOD and BAD

train test split

In [19]:
```
from sklearn.model_selection import train_test_split
```

In [20]:
```
#splitting X,Y into training and testing data
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random
```

In [21]:
```
print(X.shape, X_train.shape, X_test.shape)
```

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

modal training

Modal 1: LOGISTIC REGRESSION

In [22]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score #to evaluate the model
```

In [23]:
```
logreg=LogisticRegression()
```

In [24]:
```
logreg.fit(X_train,Y_train)
```

Out[24]:
```
LogisticRegression()
```

In [25]:
```
logreg_pred=logreg.predict(X_test)
logreg_acc=accuracy_score(logreg_pred,Y_test)
print("Test accuracy score is = ",logreg_acc*100)
```

```
Test accuracy score is =  89.6875
```

Model 2 - Decision Tree Model

In [26]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [27]:
```
dtree = DecisionTreeClassifier()
```

In [28]:
```
#training the model
dtree.fit(X_train, Y_train)
```

Out[28]:
```
DecisionTreeClassifier()
```

In [29]:
```
#model evaluation
dtree_pred = dtree.predict(X_test)
dtree_acc = accuracy_score(dtree_pred, Y_test)
print("Test Accuracy score is:", dtree_acc*100)
```

```
Test Accuracy score is: 89.6875
```

Model 3 - Random Forest Classifier

In [30]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [31]:
```
rforest = RandomForestClassifier()
```

In [32]:
```
rforest.fit(X_train, Y_train)
```

Out[32]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [33]:
```
rforest_pred=rforest.predict(X_test)
rforest_acc=accuracy_score(rforest_pred,Y_test)
print("The accuracy score is=",rforest_acc*100)
```

```
The accuracy score is= 93.125
```

Conclusion: Random Forest has better accuracy than other two models (Logistic Regression and Decision Tree

In [ ]:
```

```