Task 1 : Wine Quality Prediction

In [1]:

```
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
```

In [2]:

```
# Loading dataset
```

In [3]:

```
data = pd.read_csv("C:\\Users\\Vikas\\OneDrive\\Desktop\\DATASET\\winequality-red.csv")
```

In [4]:

```
data.head()
```

Out[4]:

	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

In [5]:

```
# Number of rows and columns in the dataset
```

In [6]:

```
data.shape
```

Out[6]:

```
(1599, 12)
```

In [7]:

```
# Checking for missing values
```

In [8]:

```
1 data.isnull().sum()
```

Out[8]:

```
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
```

Data Analysis and Visualization

In [9]:

```
1 data.describe()
```

Out[9]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	den
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003

In [10]:

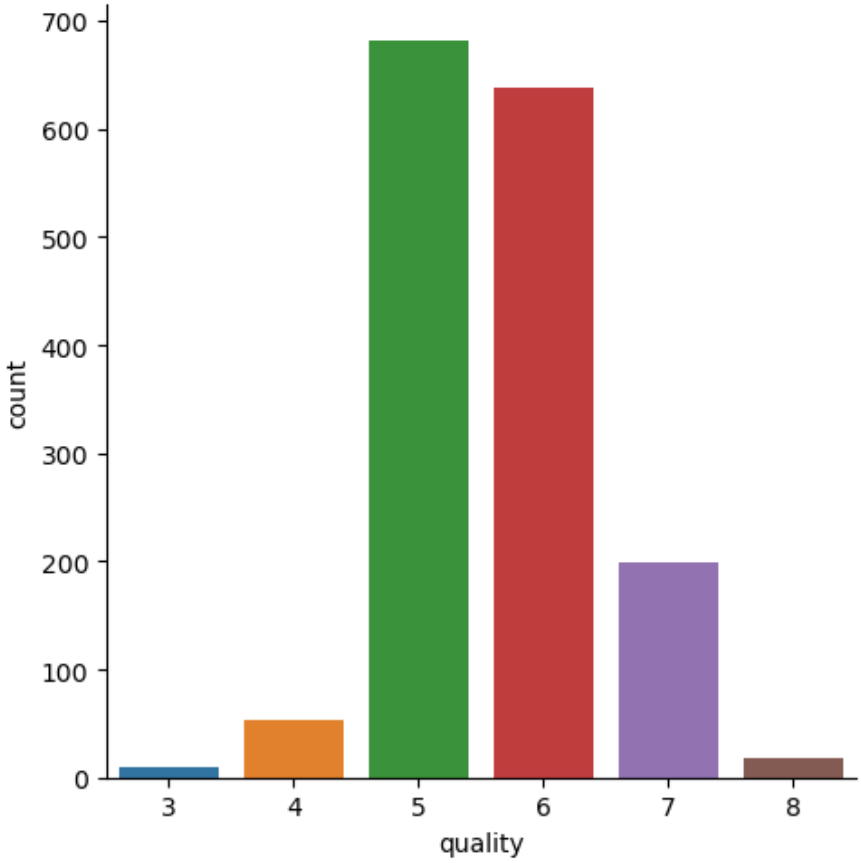
```
1 # Number of values of each quality
```

In [11]:

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

Out[11]:

```
<seaborn.axisgrid.FacetGrid at 0x1bbf7b999a0>
```



In [12]:

```
1 # volatile acidity Vs Quality
```

In [13]:

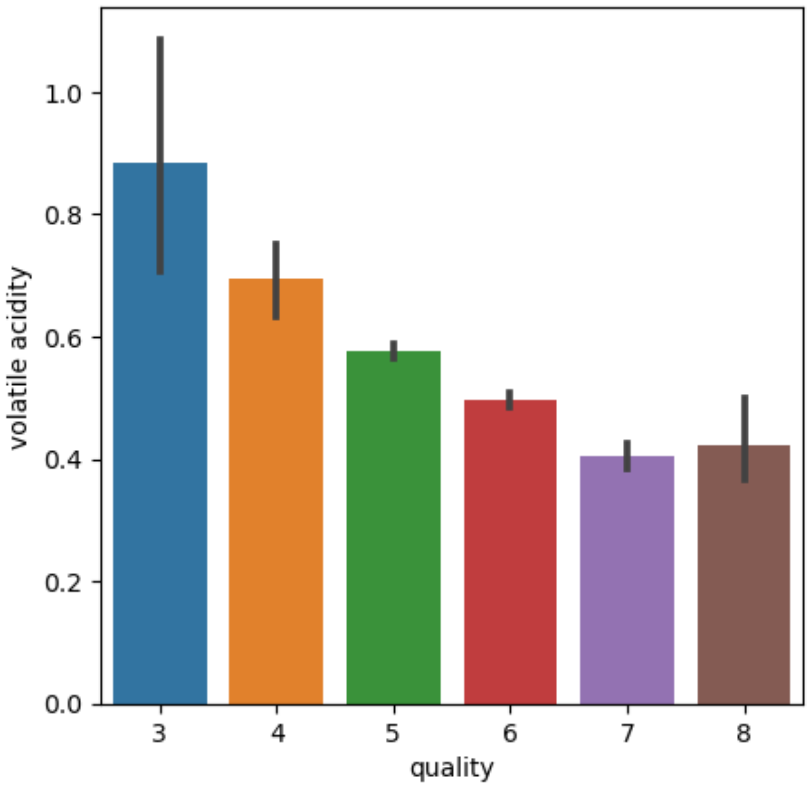
```
1 plot = plt.figure(figsize= (5,5))
2 sns.barplot(x='quality',y= 'volatile acidity',data = data)
```

Out[13]:

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

In [14]:

```
1 plt.show()
```



In [ ]:

```
1
```

In [15]:

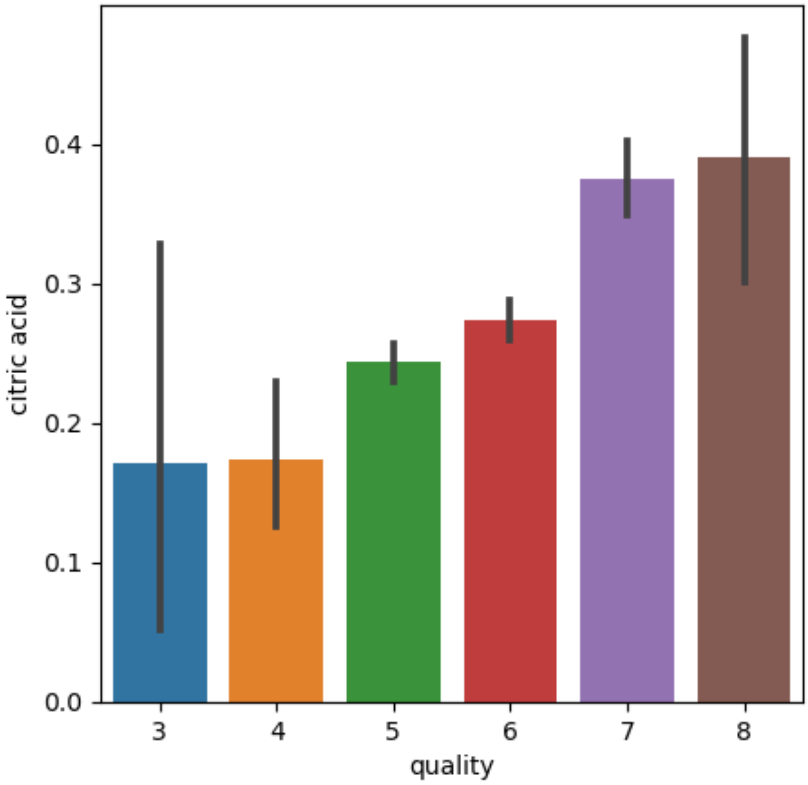
```
1 plot = plt.figure(figsize= (5,5))
2 sns.barplot(x='quality',y= 'citric acid',data = data)
```

Out[15]:

```
<AxesSubplot:xlabel='quality', ylabel='citric acid'>
```

In [16]:

```
1 plt.show()
```



Correlation

In [17]:

```
1 correlation = data.corr()
```

In [18]:

```
1 # Constructing a heatmap to understand the correlation between the columns
```

In [19]:

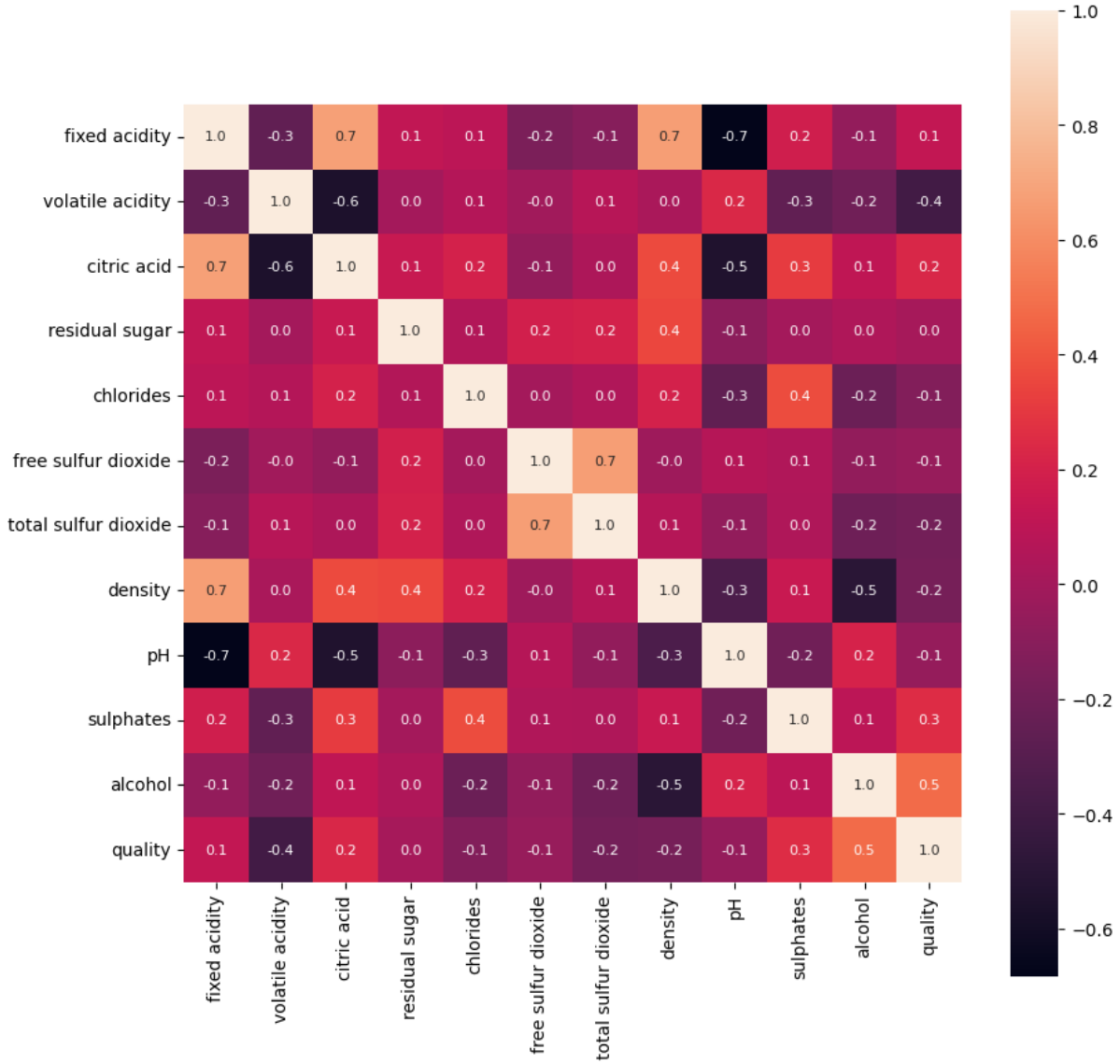
```
1 plt.figure(figsize=(10,10))
2 sns.heatmap(correlation, cbar = True, square=True,fmt = '.1f',annot = True, annot_kws= {'size':8
```

Out[19]:

```
<AxesSubplot:>
```

In [20]:

```
1 plt.show()
```



Data Preprocessing

In [21]:

```
1 # Separate the data and Label
```

In [22]:

```
1 X = data.drop('quality',axis = 1)
```

In [23]:

```
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 [24]:

```
Y = data['quality'].apply(lambda y_value: 1 if y_value>=7 else 0)
```

In [25]:

```
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 [26]:

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=3)
```

In [27]:

```
print(Y.shape,Y_train.shape,Y_test.shape)
```

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

Model Training

Random Forest Classifier

In [28]:

```
model = RandomForestClassifier()
```

In [29]:

```
model.fit(X_train,Y_train)
```

Out[29]:

```
RandomForestClassifier()
```

Model Evaluation

Accuracy Score

In [30]:

```
# Accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction,Y_test)
```

In [31]:

```
print('Accuracy : ', test_data_accuracy)
```

```
Accuracy :  0.93125
```

Building a Predictive System

In [32]:

```
input_data = (7.3,0.65,0.0,1.2,0.065,15.0,21.0,0.9946,3.39,0.47,10.0)

# Changing the input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

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

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

```
[1]

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does no
t have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

In [33]:

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

```
Good Quality Wine
```

In [34]:

```
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 the input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

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

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

```
[0]

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does no
t have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

In [35]:

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

```
Bad Quality Wine
```