In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.model_selection import train_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
```

In [2]:
```
wine_quality=pd.read_csv("C:\\Users\\Pranav\\Desktop\\DATA SCIENCE DATA\\CVC file\\winequality-red.csv")
wine_quality.head()
```

Out[2]:

	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 [3]:
```
#number of rows and columns
wine_quality.shape
```

Out[3]:
```
(1599, 12)
```

In [4]:
```
#to find missing value in data
wine_quality.isnull().sum()
```

Out[4]:
```
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 [5]:
```
#describe mathematical data
wine_quality.describe()
```

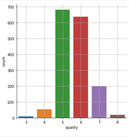
Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	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	10.422983	5.636023
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001897	0.154386	0.169507	1.065668	0.807569
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

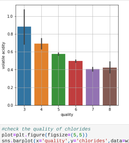
In [6]:
```
#information about data
wine_quality.info()
```

```
<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
```

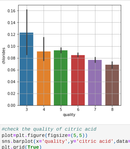
In [7]:
```
#number of values of each quality
sns.catplot(x='quality',data=wine_quality,kind='count')
plt.grid(True)
plt.show()
```



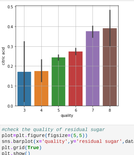
In [8]:
```
#check the quality of volatile acidity
plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality',y='volatile acidity',data=wine_quality)
plt.grid(True)
plt.show()
```



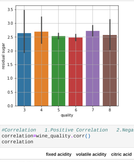
In [10]:
```
#check the quality of chlorides
plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality',y='chlorides',data=wine_quality)
plt.grid(True)
plt.show()
```



In [11]:
```
#check the quality of citric acid
plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality',y='citric acid',data=wine_quality)
plt.grid(True)
plt.show()
```



In [13]:
```
#check the quality of residual sugar
plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality',y='residual sugar',data=wine_quality)
plt.grid(True)
plt.show()
```

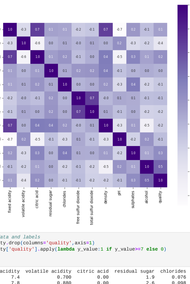


In [15]:
```
#Correlation   1.Positive Correlation   2.Negative Correlation
correlation=wine_quality.corr()
correlation
```

Out[15]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.668047	-0.682978	0.183006	-0.061668	0.124052
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234819	-0.260987	-0.202288	-0.390558
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904	0.312770	0.109803	0.226373
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.355283	-0.085652	0.005527	0.042075	0.013732
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.200632	-0.265026	0.371260	-0.221141	-0.128907
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666	-0.021946	0.070377	0.051658	-0.069408	-0.050656
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000	0.071269	0.066495	0.042947	-0.205654	-0.185100
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269	1.000000	-0.341699	0.148506	-0.496180	-0.174919
pH	-0.682978	0.234819	-0.541904	-0.085652	-0.265026	0.070377	-0.066495	-0.341699	1.000000	-0.196648	0.205633	-0.057731
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051658	0.042947	0.148506	-0.196648	1.000000	0.093595	0.251397
alcohol	-0.061668	-0.202288	0.109803	0.042075	-0.221141	-0.069408	-0.205654	-0.496180	0.205633	0.093595	1.000000	0.476166
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050656	-0.185100	-0.174919	-0.057731	0.251397	0.476166	1.000000

In [24]:
```
#correlation heatmap to understand the correlation between the columns
plt.figure(figsize=(10,10))
sns.heatmap(correlation,cbar=True,square=True,fmt='.1f',annot=True,annot_kws={'size':8},cmap='Purples')
plt.show()
```



In [27]:
```
#separating data and labels
X=wine_quality.drop(columns='quality',axis=1)
y=wine_quality['quality'].apply(lambda y_value:1 if y_value>=7 else 0)
print(X)
print(y)
```

```
      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]
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
```

In [28]:
```
#training and test data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2)
print("shape of X_train ",X_train.shape)
print("shape of X_test= ",X_test.shape)
print("shape of y_train= ",y_train.shape)
print("shape of y_test= ",y_test.shape)
```

```
shape of X_train  (1279, 11)
shape of X_test=  (320, 11)
shape of y_train= (1279,)
shape of y_test=  (320,)
```

In [30]:
```
#model training   #randomforestclassifier
model=RandomForestClassifier()
```

In [33]:
```
model.fit(X_train,y_train)
```

Out[33]:
```
RandomForestClassifier()
```

In [34]:
```
#Accuracy score
 #accuracy of testing data
X_test_prediction=model.predict(X_test)
testing_data_accuracy=accuracy_score(X_test_prediction,y_test)
print('Accuracy on testing data: ',testing_data_accuracy)
```

```
Accuracy on testing data: 0.928125
```

In [39]:
```
#Making a Predictive system
#changing the input_data to a numpy array
input_data=(7.4,0.66,0.0,1.8,0.075,13.0,40.0,0.9978,3.51,0.56,9.4)
input_data_numpy_array=np.asarray(input_data)
input_data_numpy_array1
```

Out[39]:
```
array([7.4, 0.66, 0.0, 1.8 , 0.075 , 13.  ,  40.  ,  0.9978,  3.51,
       0.56,  9.4 ])
```

In [41]:
```
#reshape the np array as we are predicting for one instance
input_data_reshaped1=input_data_numpy_array1.reshape(1,-1)
print(input_data_reshaped1)
```

```
[[ 7.4   0.66  0.    1.8   0.075 13.    40.    0.9978  3.51
   0.56  9.4  ]]
```

In [42]:
```
prediction=model.predict(input_data_reshaped1)
print(prediction)
```

```
[0]
```

```
C:\Users\Pranav\Searches\hjhkhlib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

In [43]:
```
if(prediction[0]==1):
    print('The Good quality of wine')
else:
    print('The Bad Quality of wine')
```

```
The Bad Quality of wine
```

In [47]:
```
input_data3=(8.3,0.75,0.1,2.6,0.086,45.0,87.0,0.9983,3.40,0.53,10.0)
input_data_numpy_array2=np.asarray(input_data2)
```

In [48]:
```
input_data_reshaped2=input_data_numpy_array2.reshape(1,-1)
input_data_reshaped2
```

Out[48]:
```
array([[8.300e+00, 7.800e-01, 1.000e-01, 2.600e+00, 8.100e-02, 4.500e+01,
        8.700e+01, 9.983e-01, 3.400e+00, 5.300e-01, 1.000e+01]])
```

In [49]:
```
prediction=model.predict(input_data_reshaped2)
print(prediction)
```

```
[0]
```

```
C:\Users\Pranav\Searches\hjhkhlib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

In [53]:
```
if(prediction[0]==1):
    print('The Good quality of wine')
else:
    print('The Bad Quality of wine')
```

```
The Bad Quality of wine
```