Wine Quality Prediction

Download Library

In [ ]:
```
# pip install pandas matplotlib seaborn
```

Required Library

In [1]:
```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
```

Load Dataset

In [2]:
```
dataset=pd.read_csv('WineQT.csv')
```

In [3]:
```
dataset
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphate
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.6
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.6
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.5
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.5
...	...	...	...	...	...	...	...	...	...	.
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.7
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.8
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.5
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.7
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.7

1143 rows × 13 columns

In [4]:
```
dataset.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 [5]:
```
dataset.describe()
```

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	tota
count	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143
mean	8.311111	0.531339	0.268364	2.532152	0.086933	15.615486	45
std	1.747595	0.179633	0.196686	1.355917	0.047267	10.250486	32
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6
25%	7.100000	0.392500	0.090000	1.900000	0.070000	7.000000	21
50%	7.900000	0.520000	0.250000	2.200000	0.079000	13.000000	37
75%	9.100000	0.640000	0.420000	2.600000	0.090000	21.000000	61
max	15.900000	1.580000	1.000000	15.500000	0.611000	68.000000	289

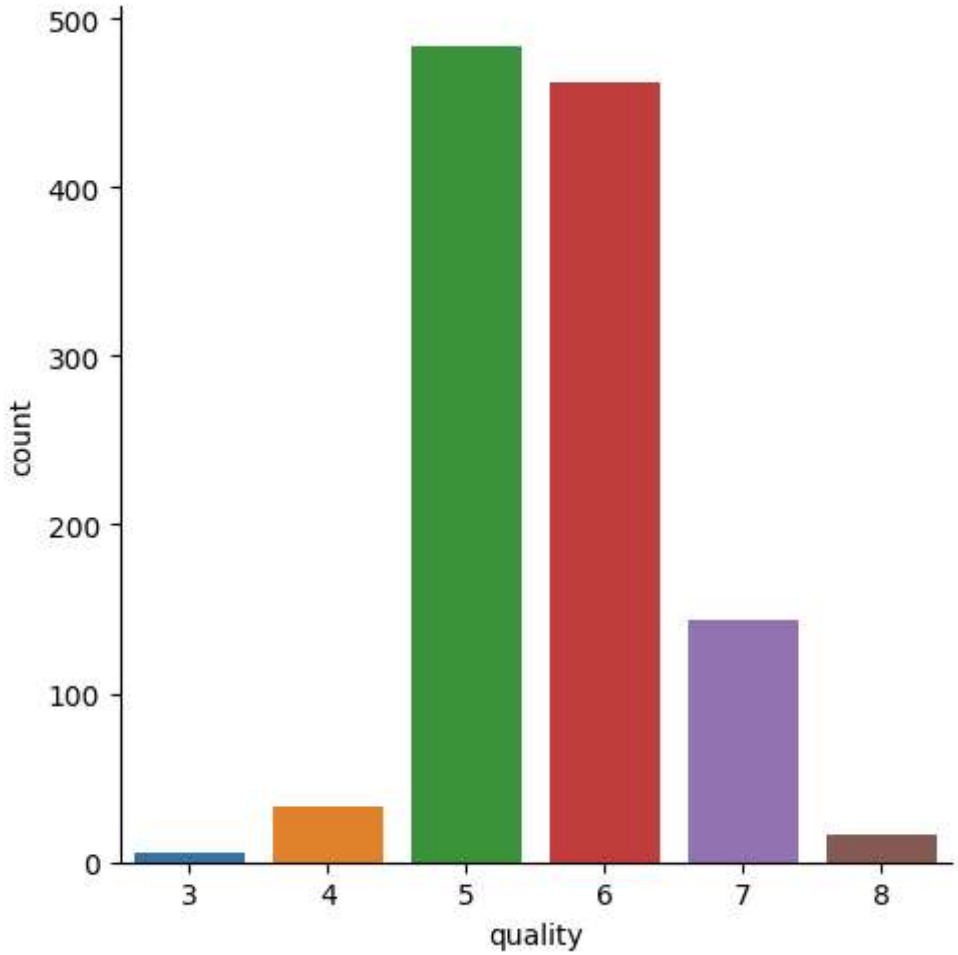
In [6]:
```
dataset.columns
```

Out[6]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality', 'Id'],
      dtype='object')
```

In [7]:
```
dataset=dataset.drop(['Id'],axis=1)
```

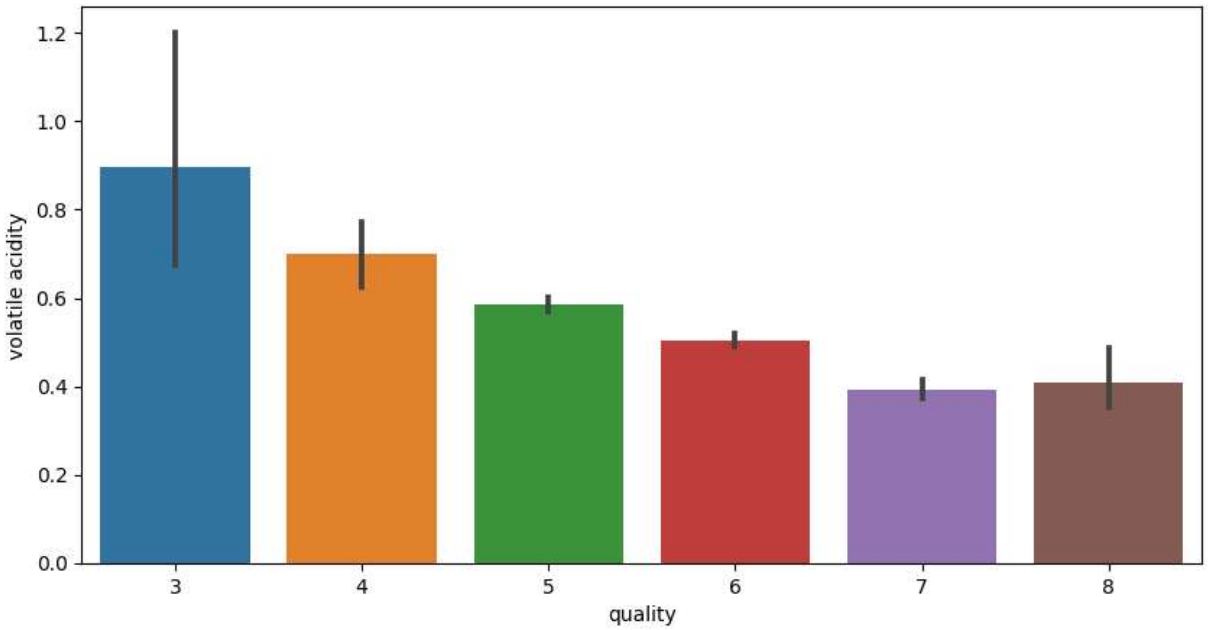
In [8]:
```
sns.catplot(x='quality',data=dataset,kind='count')
```

Out[8]:
```
<seaborn.axisgrid.FacetGrid at 0x23a43c437f0>
```



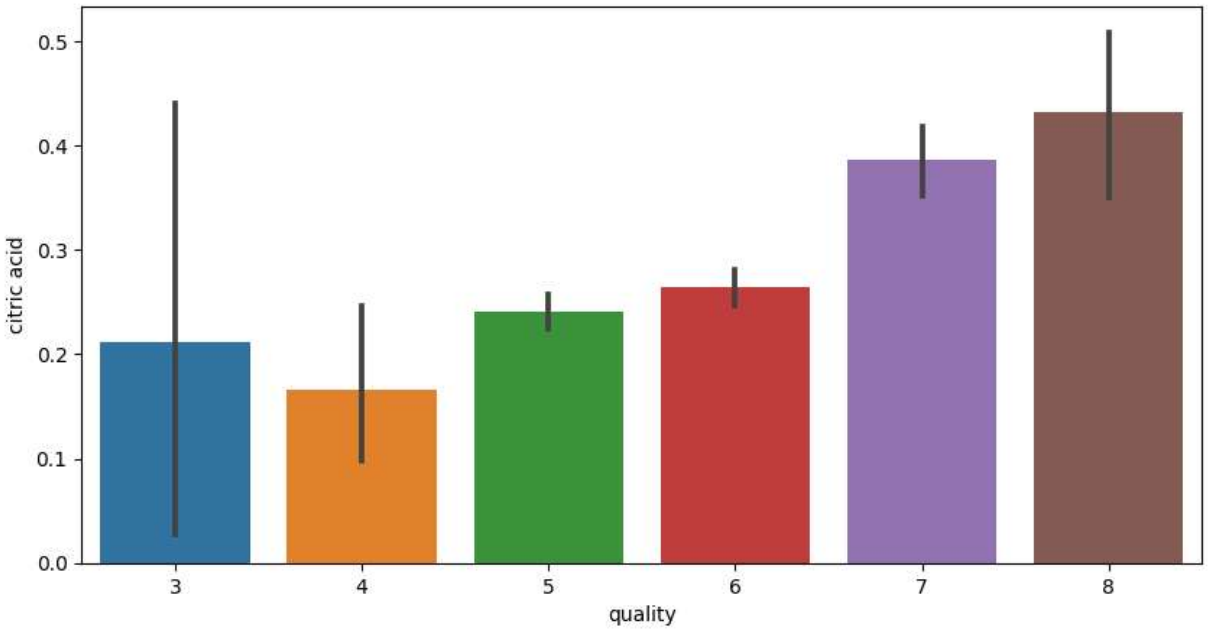
In [9]:
```
# volatile acidity vs quality
plot = plt.figure(figsize=(10,5))
sns.barplot(x='quality',y='volatile acidity',data = dataset)
```

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



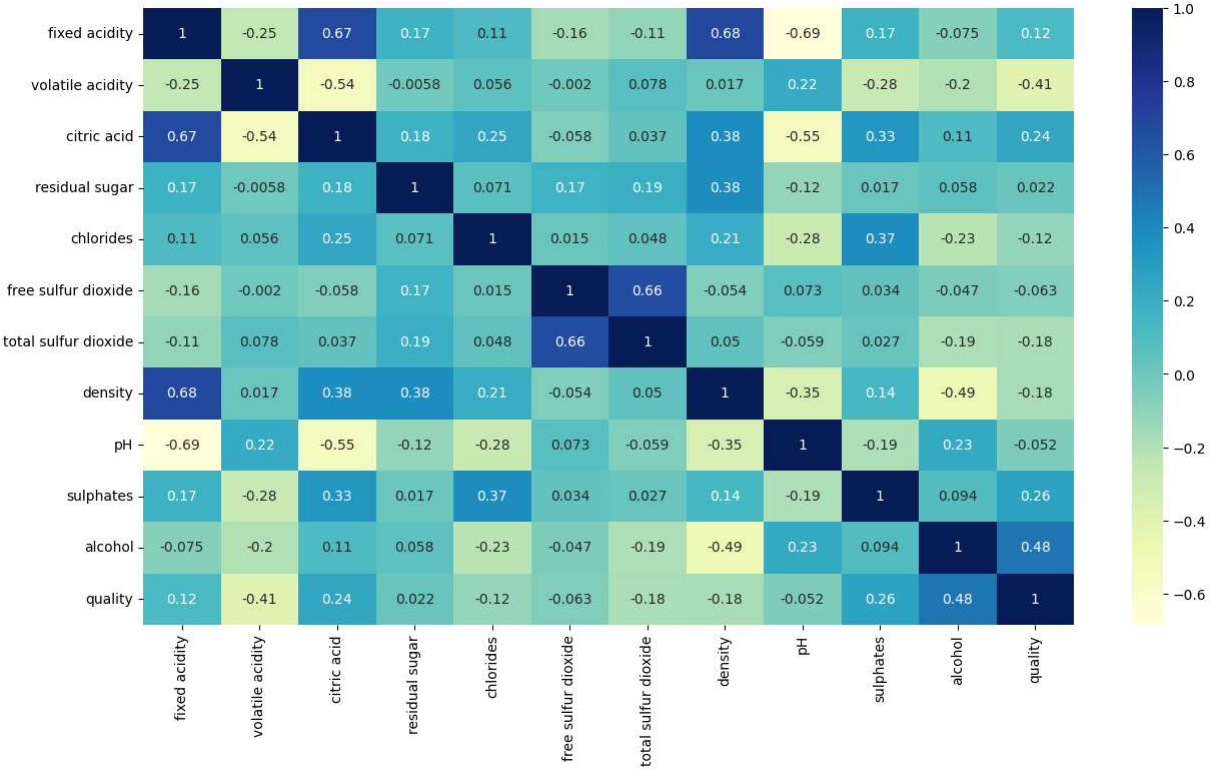
In [10]:
```
plot = plt.figure(figsize=(10,5))
sns.barplot(x='quality',y='citric acid',data = dataset)
```

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



In [11]:
```
# create a heatmap to understand the corr relation between coloums
plt.figure(figsize=(15,8))
sns.heatmap(dataset.corr(),annot=True , cmap="YlGnBu")
```

Out[11]: <Axes: >



In [12]:
```
# Data Preprocessing
```

In [13]:
```
x=dataset.drop(['quality'],axis=1)
```

In [14]:
```
x
```

Out[14]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphate
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.6
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.6
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.5
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.5
...	...	...	...	...	...	...	...	...	...	.
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.7
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.8
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.5
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.7
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.7

1143 rows × 11 columns

In [15]:
```
# Label Binarization
```

In [16]:
```
y=dataset['quality'].apply(lambda y_value: 1 if y_value>=7 else 0 )
```

In [17]:
```
y
```

Out[17]:
```
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
```

In [ ]:

In [ ]:

Split the data into two part

In [18]:
```
from sklearn.model_selection import train_test_split
```

In [19]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
```

In [20]:
```
x_train
```

Out[20]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphate
1130	7.4	0.350	0.33	2.4	0.068	9.0	26.0	0.99470	3.36	0.6
453	9.5	0.885	0.27	2.3	0.084	31.0	145.0	0.99780	3.24	0.5
1072	6.2	0.440	0.39	2.5	0.077	6.0	14.0	0.99555	3.51	0.6
877	6.5	0.670	0.00	4.3	0.057	11.0	20.0	0.99488	3.45	0.5
193	7.9	0.545	0.06	4.0	0.087	27.0	61.0	0.99650	3.36	0.6
...	...	...	...	...	...	...	...	...	...	.
1099	6.2	0.520	0.08	4.4	0.071	11.0	32.0	0.99646	3.56	0.6
466	10.7	0.430	0.39	2.2	0.106	8.0	32.0	0.99860	2.89	0.5
299	9.5	0.780	0.22	1.9	0.077	6.0	32.0	0.99880	3.26	0.5
493	5.1	0.470	0.02	1.3	0.034	18.0	44.0	0.99210	3.90	0.6
527	9.0	0.690	0.00	2.4	0.088	19.0	38.0	0.99900	3.35	0.6

914 rows × 11 columns

In [21]:
```
x_test
```

Out[21]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
835	9.8	0.390	0.43	1.65	0.068	5.0	11.0	0.99478	3.19	0.46
226	7.4	0.360	0.29	2.60	0.087	26.0	72.0	0.99645	3.39	0.68
199	11.4	0.260	0.44	3.60	0.071	6.0	19.0	0.99860	3.12	0.82
158	6.8	0.610	0.04	1.50	0.057	5.0	10.0	0.99525	3.42	0.60
597	6.7	0.280	0.28	2.40	0.012	36.0	100.0	0.99064	3.26	0.39
...	...	...	...	...	...	...	...	...	...	...
86	7.8	0.500	0.17	1.60	0.082	21.0	102.0	0.99600	3.39	0.48
164	8.5	0.370	0.20	2.80	0.090	18.0	58.0	0.99800	3.34	0.70
99	8.1	0.670	0.55	1.80	0.117	32.0	141.0	0.99680	3.17	0.62
286	7.1	0.735	0.16	1.90	0.100	15.0	77.0	0.99660	3.27	0.64
947	7.2	0.835	0.00	2.00	0.166	4.0	11.0	0.99608	3.39	0.52

229 rows × 11 columns

In [ ]:

Train the Data using RandomForestRegressor

In [22]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [23]:
```
newModel=RandomForestClassifier()
```

In [24]:
```
newModel.fit(x_train,y_train)
```

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

In [25]:
```
newModel.score(x_test,y_test)
```

Out[25]:
```
0.9170305676855895
```

In [ ]: