Name : Dasari Narsingarao

Domain : Data Science

Organisation : Statsly Analytics

Task 6: Wine Qulity Prediction and Deployment

Linkedin Profile : https://www.linkedin.com/in/narsingarao-dasari-966477152/`

Importing the libraries

In [119…

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,classification_report
```

Data Collection

Reading the data

In [58]:

```
df = pd.read_csv("D:\Internship\EDA ANALYSIS\TASK_6\Winequality-red.csv")
```

In [59]:

```
df.head()
```

Out[59]:

	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 [60]:

```
df.tail()
```

Out[60]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

In [61]:

```
df.shape
```

Out[61]:

```
(1599, 12)
```

In [62]:

```
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
```

Data Analysis and Visulaization

In [63]:

```
df.describe()
```

Out[63]:

	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	0.001887	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 [64]:

```
df.isnull().sum()
```

Out[64]:

```
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 [65]:

```
df.isnull().sum()
```

Out[65]:

```
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 [66]:

```
df.columns
```

Out[66]:

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

In [67]:

```
df.dtypes
```

Out[67]:

```
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
quality                   int64
dtype: object
```

In [68]:

```
df.duplicated().sum()
```

Out[68]:

```
240
```

In [69]:

```
df.drop_duplicates(inplace=True)
```

In [71]:

```
df.duplicated().sum()
```

Out[71]:

```
0
```

In [73]:

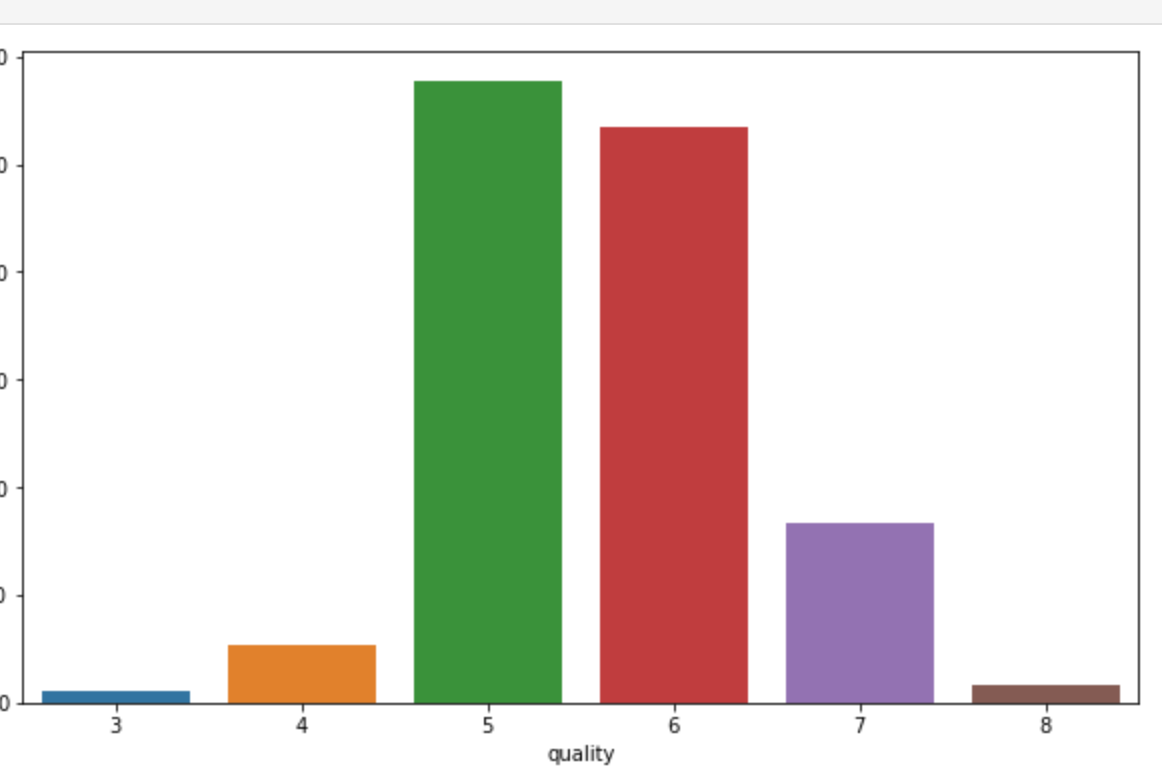
```
df.isnull().sum()
```

Out[73]:

```
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 [74]:

```
plt.figure(figsize =(10,6))
sns.countplot(df['quality'])
plt.show()
```



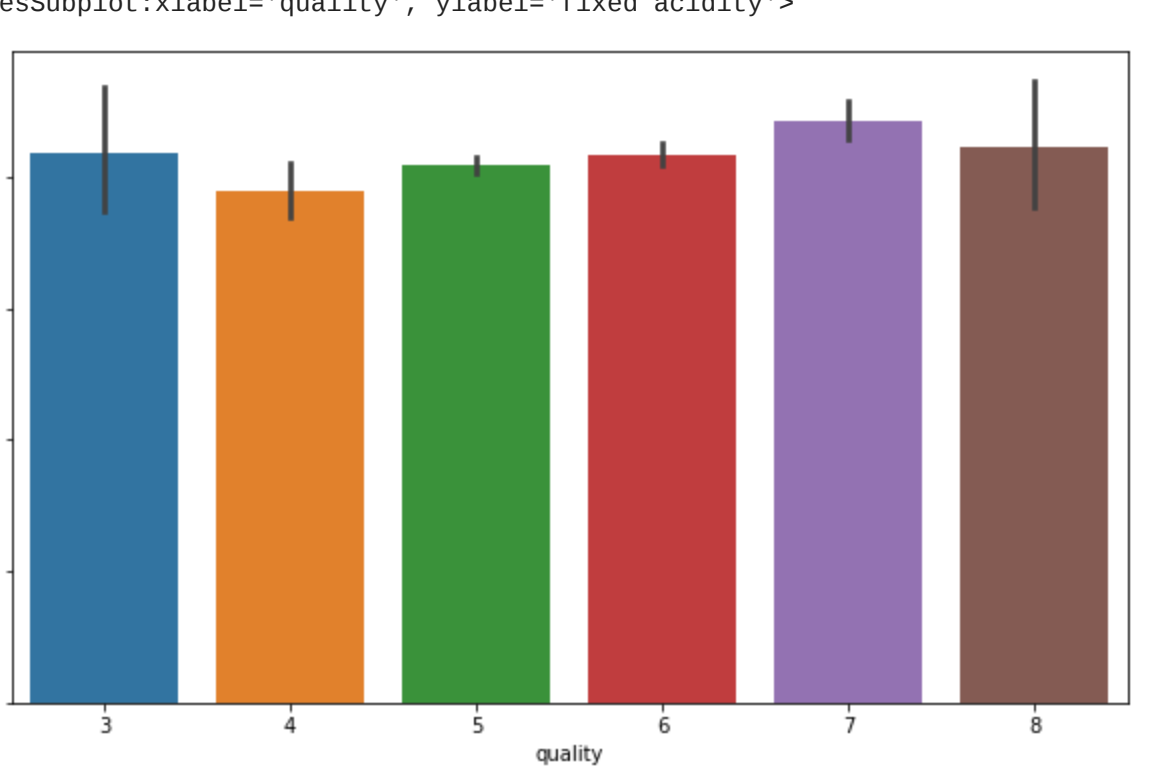
some plotting to know how the data columns are distributed in the dataset

In [75]:

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'fixed acidity', data = df)
```

Out[75]:

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

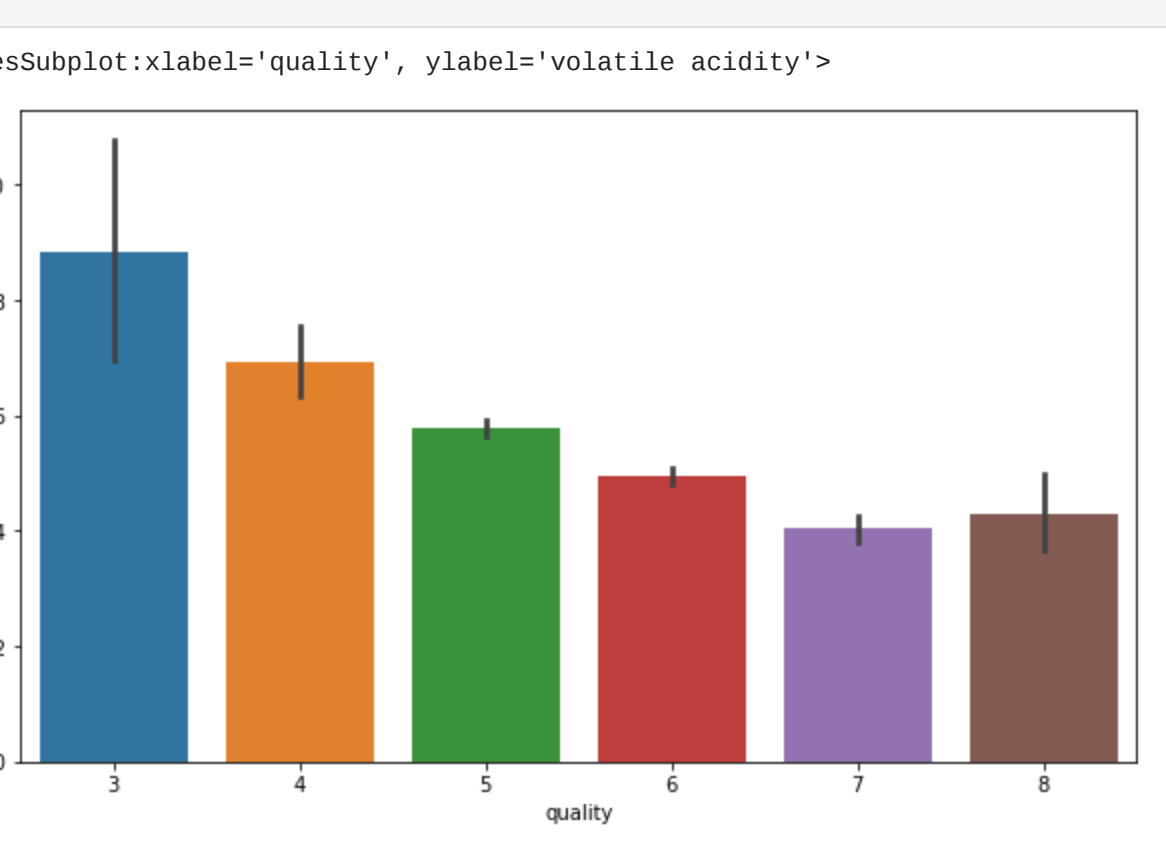


In [76]:

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'volatile acidity', data = df)
```

Out[76]:

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

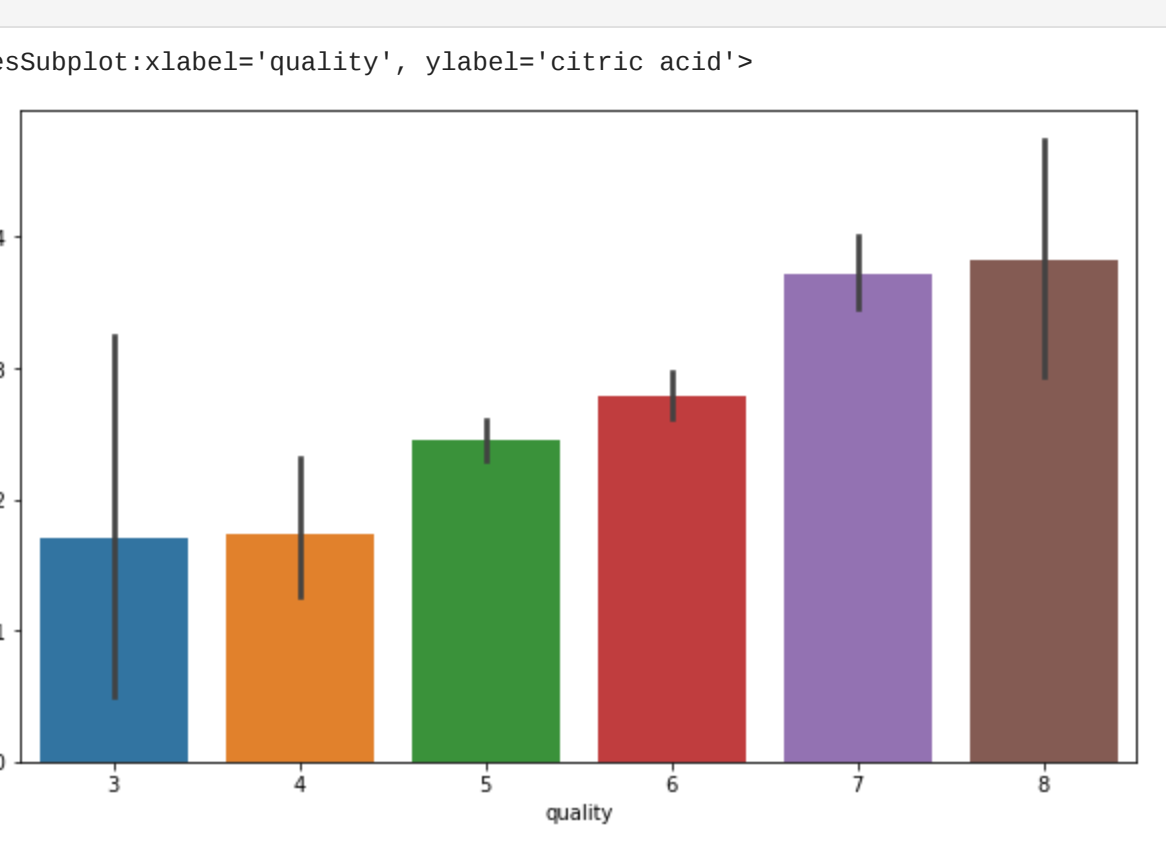


In [26]:

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'citric acid', data = df)
```

Out[26]:

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

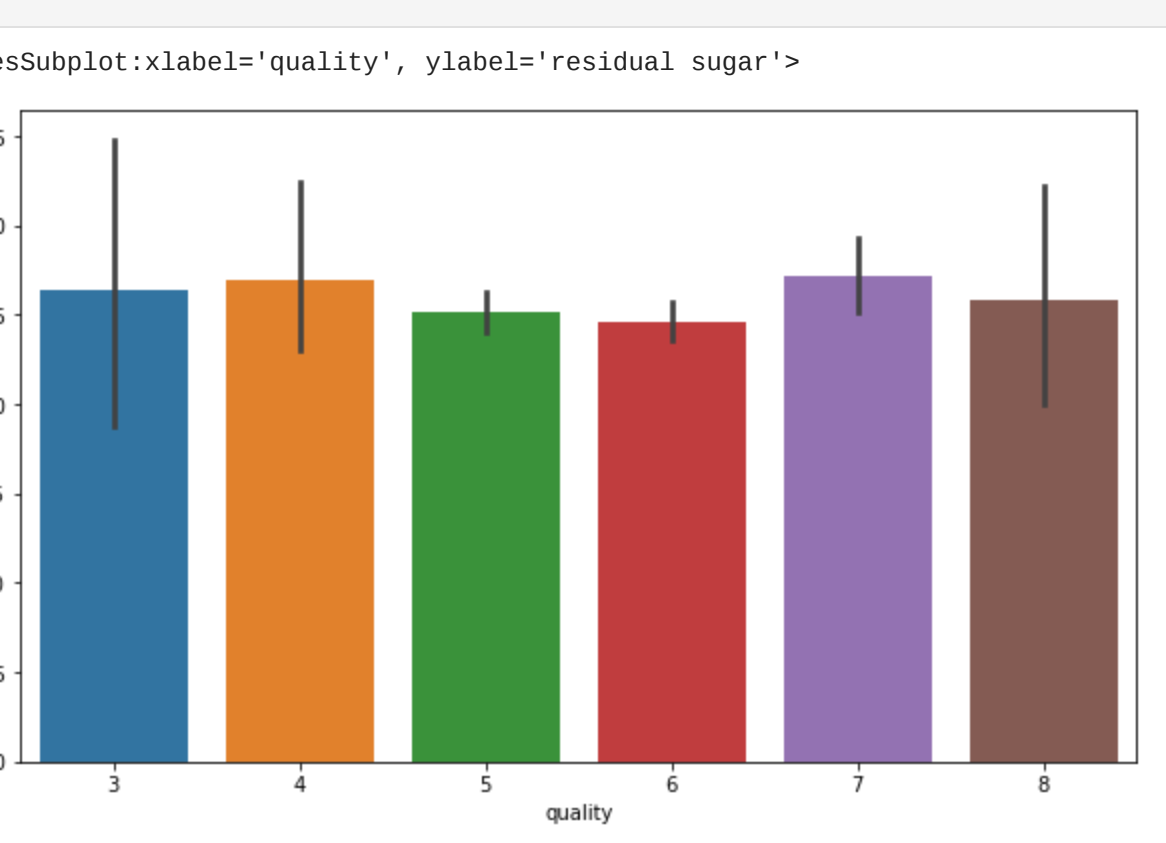


In [77]:

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'residual sugar', data = df)
```

Out[77]:

```
<AxesSubplot:xlabel='quality', ylabel='residual sugar'>
```

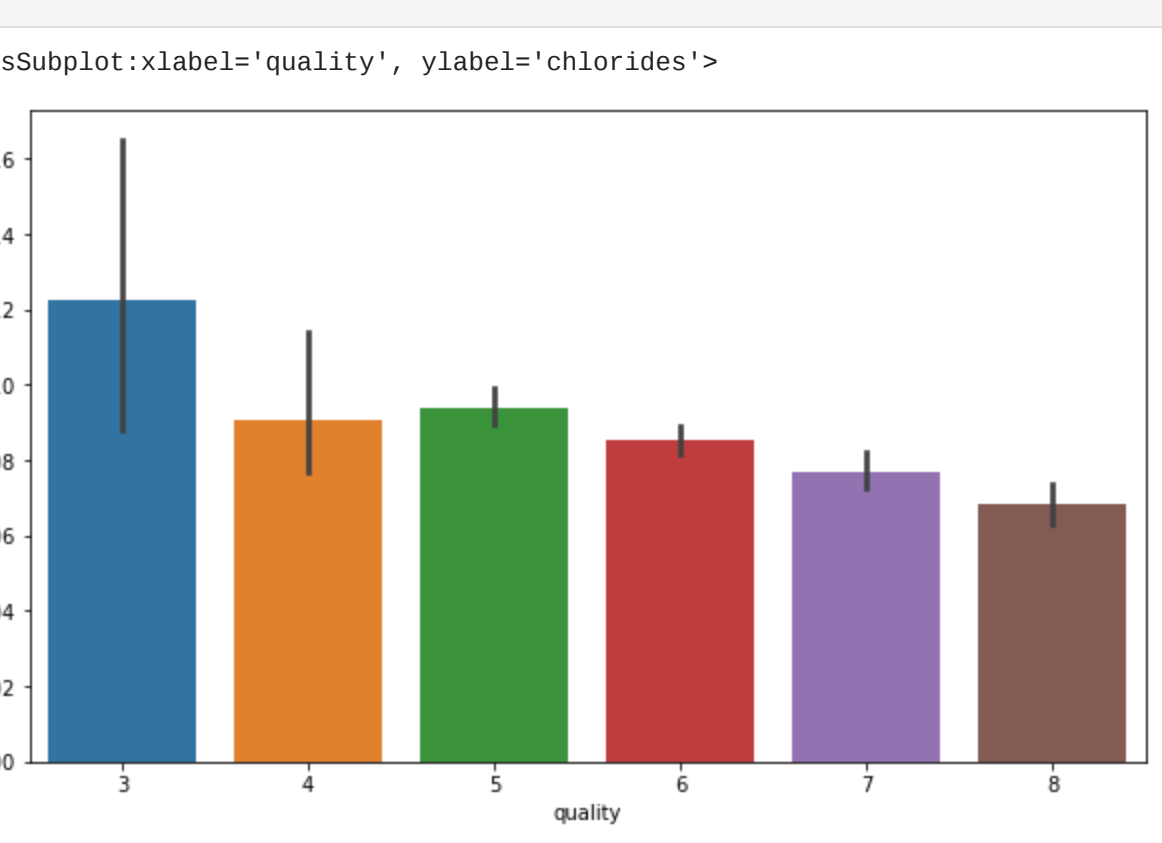


In [78]:

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'chlorides', data = df)
```

Out[78]:

```
<AxesSubplot:xlabel='quality', ylabel='chlorides'>
```

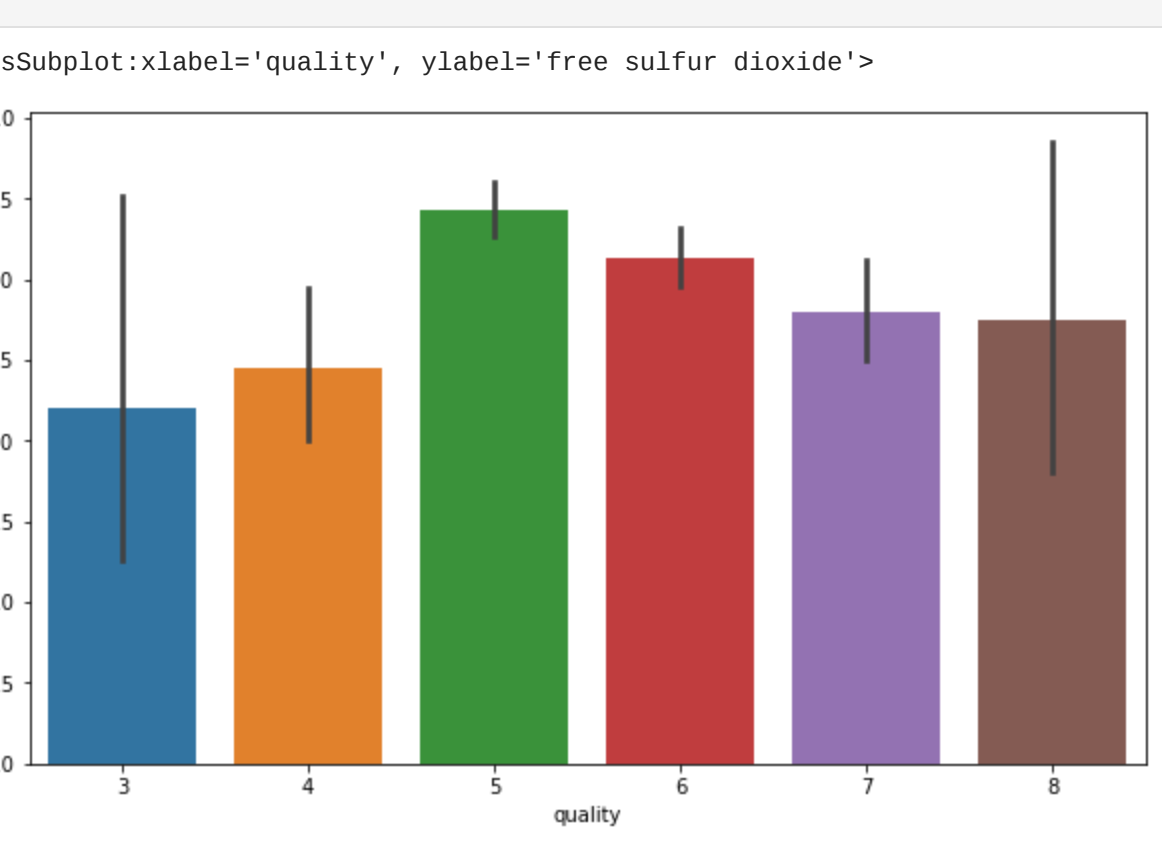


In [79]:

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'free sulfur dioxide', data = df)
```

Out[79]:

```
<AxesSubplot:xlabel='quality', ylabel='free sulfur dioxide'>
```

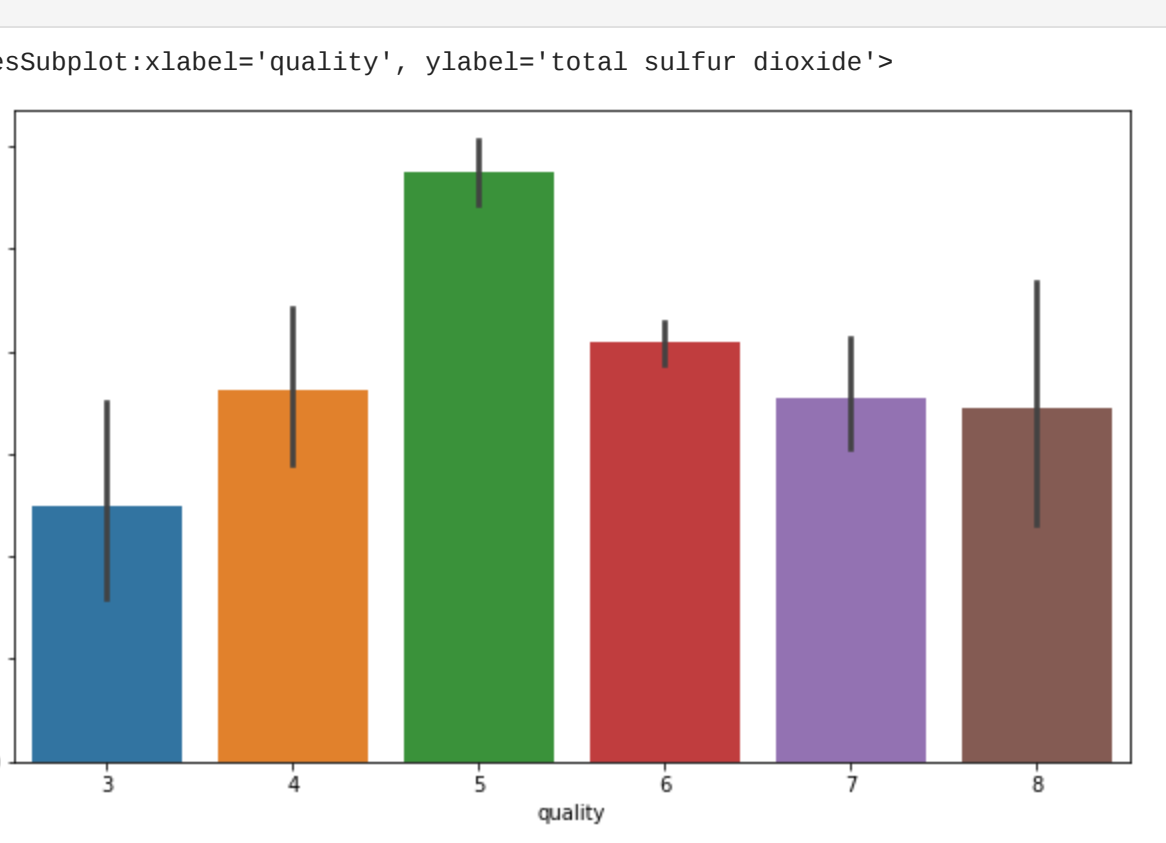


In [80]:

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'total sulfur dioxide', data = df)
```

Out[80]:

```
<AxesSubplot:xlabel='quality', ylabel='total sulfur dioxide'>
```

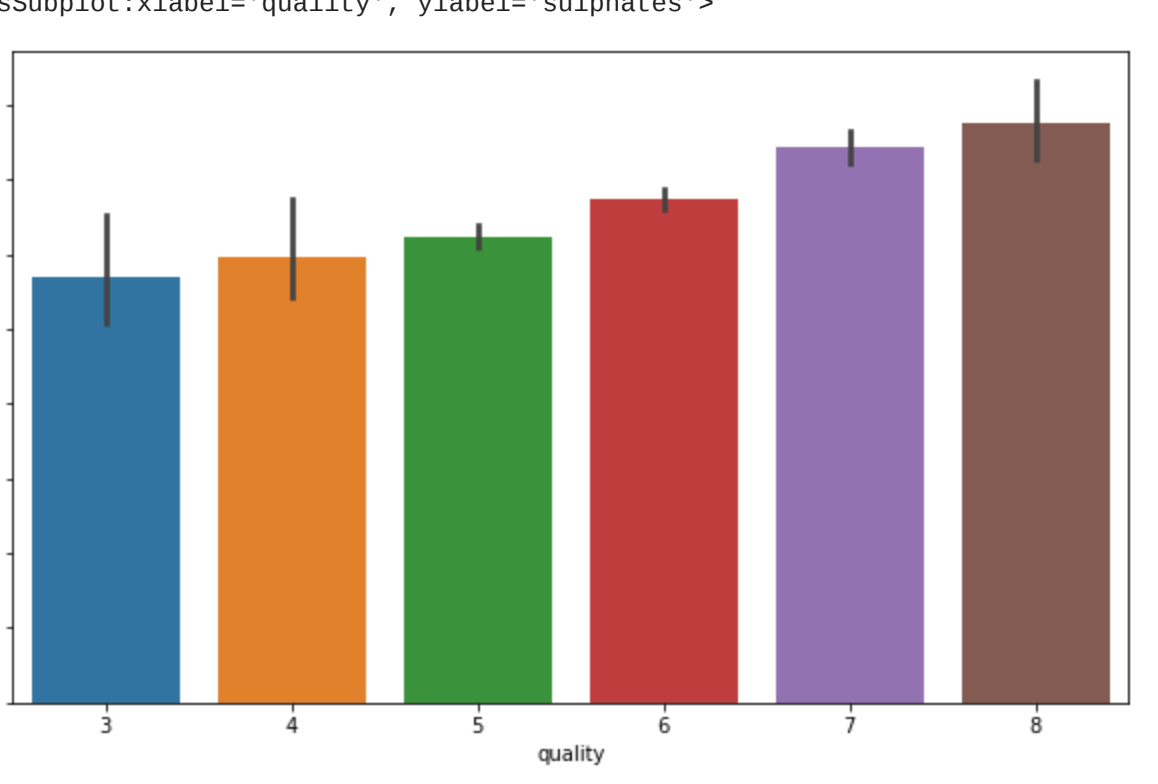


In [81]:

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'sulphates', data = df)
```

Out[81]:

```
<AxesSubplot:xlabel='quality', ylabel='sulphates'>
```

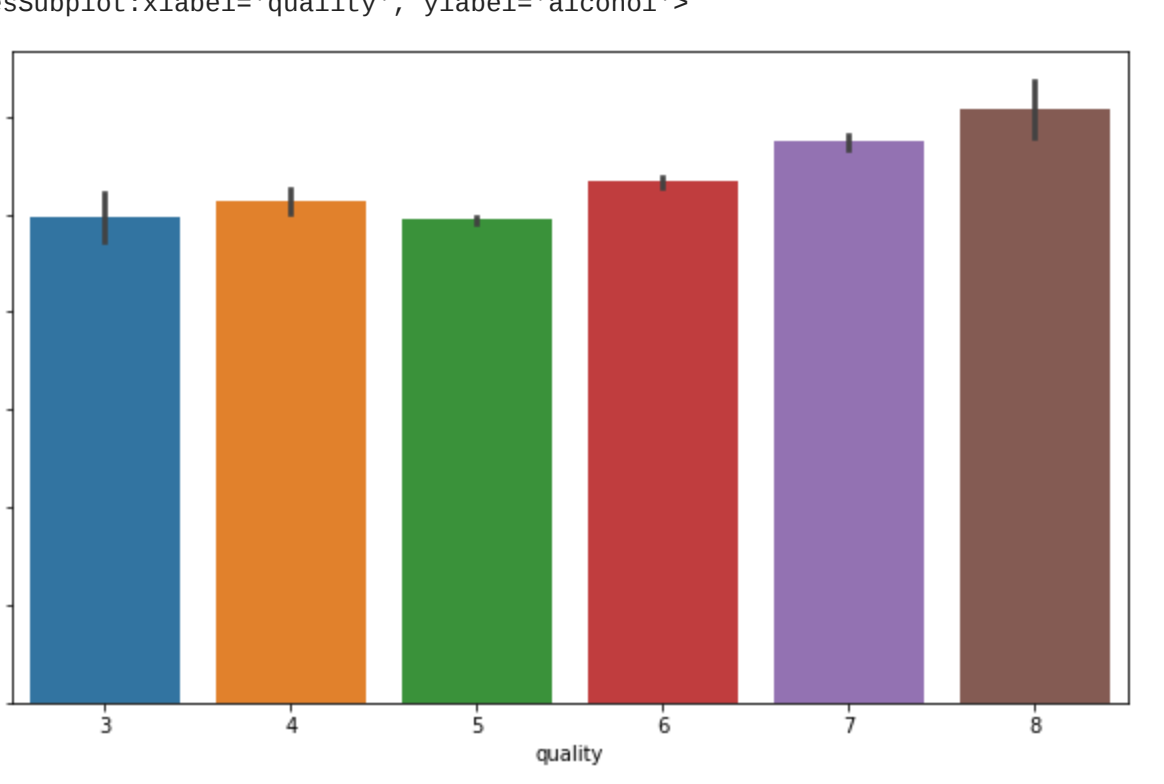


In [82]:

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'alcohol', data = df)
```

Out[82]:

```
<AxesSubplot:xlabel='quality', ylabel='alcohol'>
```



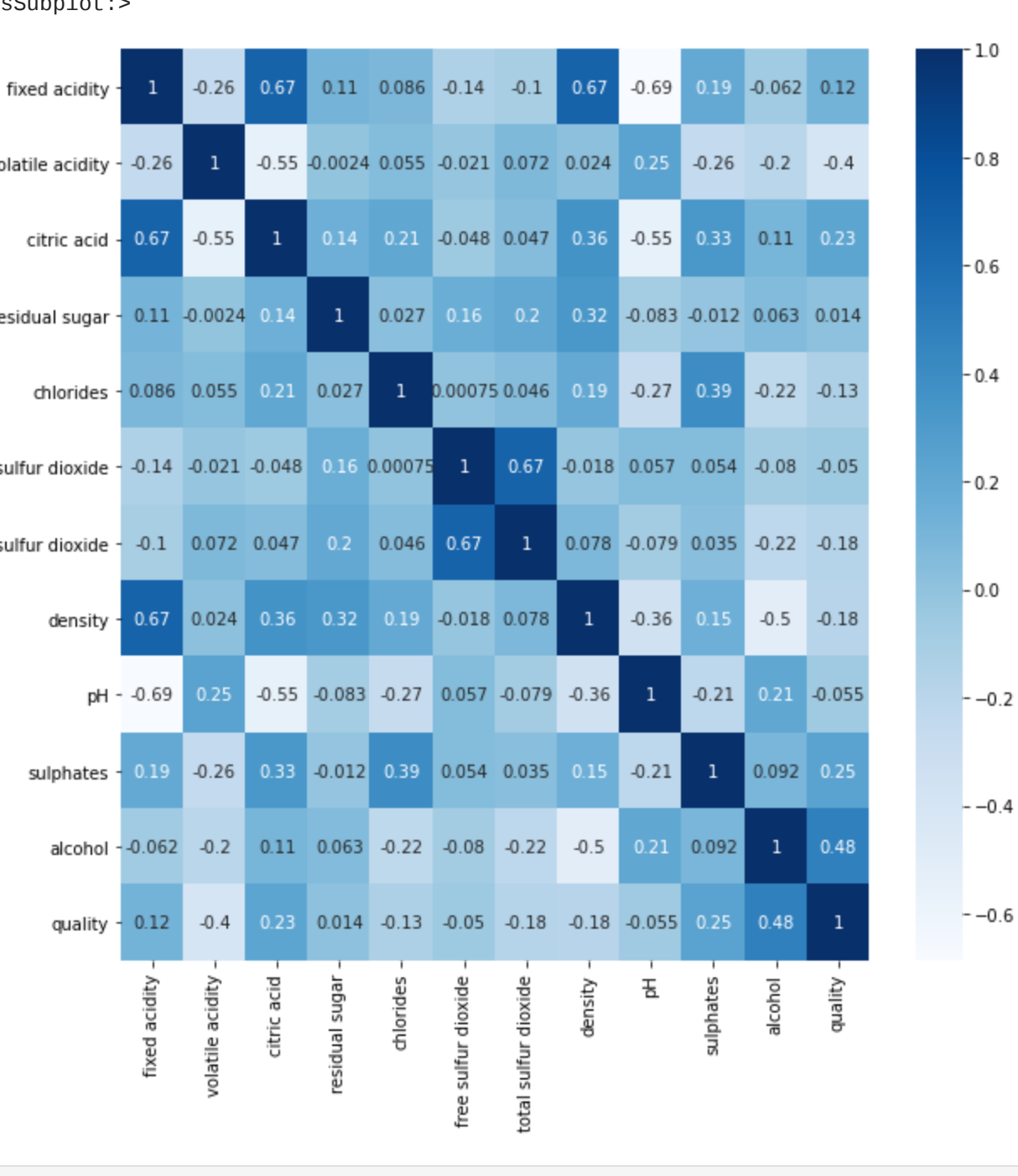
Correlation

In [83]:

```
corr = df.corr()
plt.figure(figsize=(10,10))
sns.heatmap(corr,cbar=True,annot=True,cmap='Blues')
```

Out[83]:

```
<AxesSubplot:>
```



In [84]:

```
corr['quality'].sort_values(ascending=False)
```

Out[84]:

```
quality                 1.000000
alcohol                 0.480343
sulphates               0.248835
citric acid             0.228057
fixed acidity           0.119024
residual sugar          0.013640
free sulfur dioxide    -0.050463
pH                     -0.055245
chlorides              -0.130988
total sulfur dioxide   -0.177855
density                -0.184252
volatile acidity       -0.395214
Name: quality, dtype: float64
```

Data Preprocessing

In [106…

```
x = df.drop('quality',axis=1)
```

In [90]:

```
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   
5               7.4             0.660         0.00             1.8      0.075   
...             ...               ...          ...             ...        ...   
1593            6.8             0.620         0.08             1.9      0.068   
1594            6.2             0.600         0.08             2.0      0.090   
1595            5.9             0.550         0.10             2.2      0.062   
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   
5                    13.0                  40.0  0.99780  3.51       0.56   
...                   ...                   ...      ...   ...        ...   
1593                 28.0                  38.0  0.99651  3.42       0.82   
1594                 32.0                  44.0  0.99490  3.45       0.58   
1595                 39.0                  51.0  0.99512  3.52       0.76   
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  
5         9.4  
...       ...  
1593      9.5  
1594     10.5  
1595     11.2  
1597     10.2  
1598     11.0  

[1359 rows x 11 columns]
```

Label Encoder

In [107…

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

```
0       0
1       0
2       0
3       0
5       0
       ..
1593    0
1594    0
1595    0
1597    0
1598    0
Name: quality, Length: 1359, dtype: int64
```

Train and Test Split

In [108…

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=3)
```

In [118…

```
print(y.shape,y_train.shape,y_test.shape)
```

```
(1359,) (1087,) (272,)
```

Model Training :

Random Forest Classifier

In [110…

```
model = RandomForestClassifier()
```

In [111…

```
model.fit(x_train,y_train)
```

Out[111…

```
RandomForestClassifier()
```

Model Evaluation

Accuracy score

In [115…

```
x_test_prediction = model.predict(x_test)
test_data_accuracy = accuracy_score(x_test_prediction,y_test)
```

In [116…

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

```
Accuracy : 0.8860294117647058
```

In [122…

```
print(classification_report(y_test,x_test_prediction))
```

```
              precision    recall  f1-score   support

           0       0.90      0.97      0.94       237
           1       0.62      0.29      0.39        35

    accuracy                           0.89       272
   macro avg       0.76      0.63      0.66       272
weighted avg       0.87      0.89      0.87       272
```

Building a Predictive System

In [124…

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

input_data_as_numpy_array = np.asarray(input_data)

input_data_reshaped = input_data_as_numpy_array.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]
Good Quality Wine
```

Observation

As a result, we can see Random Forest model has the best accurary ratio for predicting our wine quality!

In [ ]: