Name: Shruthi Gangam Registration Number: 21BCB0063 Campus: Vellore

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df=pd.read_csv('/content/winequality-red.csv')

df.shape

(1599, 12)

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

df.describe()

       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1599.000000       1599.000000  1599.000000     1599.000000   
mean        8.319637          0.527821     0.270976        2.538806   
std         1.741096          0.179060     0.194801        1.409928   
min         4.600000          0.120000     0.000000        0.900000   
25%         7.100000          0.390000     0.090000        1.900000   
50%         7.900000          0.520000     0.260000        2.200000   
75%         9.200000          0.640000     0.420000        2.600000   
max        15.900000          1.580000     1.000000       15.500000   

         chlorides  free sulfur dioxide  total sulfur dioxide  
density  \
count  1599.000000          1599.000000           1599.000000  
```

```
1599.000000
mean      0.087467             15.874922             46.467792
0.996747
std       0.047065             10.460157             32.895324
0.001887
min       0.012000              1.000000              6.000000
0.990070
25%       0.070000              7.000000             22.000000
0.995600
50%       0.079000             14.000000             38.000000
0.996750
75%       0.090000             21.000000             62.000000
0.997835
max       0.611000             72.000000            289.000000
1.003690

                 pH    sulphates      alcohol      quality
count  1599.000000  1599.000000  1599.000000  1599.000000
mean      3.311113     0.658149    10.422983     5.636023
std       0.154386     0.169507     1.065668     0.807569
min       2.740000     0.330000     8.400000     3.000000
25%       3.210000     0.550000     9.500000     5.000000
50%       3.310000     0.620000    10.200000     6.000000
75%       3.400000     0.730000    11.100000     6.000000
max       4.010000     2.000000    14.900000     8.000000
```

```
df.head()
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
0            7.4              0.70         0.00             1.9
0.076
1            7.8              0.88         0.00             2.6
0.098
2            7.8              0.76         0.04             2.3
0.092
3           11.2              0.28         0.56             1.9
0.075
4            7.4              0.70         0.00             1.9
0.076

   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

2                 15.0                  54.0   0.9970  3.26       0.65

3                 17.0                  60.0   0.9980  3.16       0.58
```

```
4                  11.0                    34.0   0.9978  3.51       0.56
```

```
   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
```

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

```
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
```

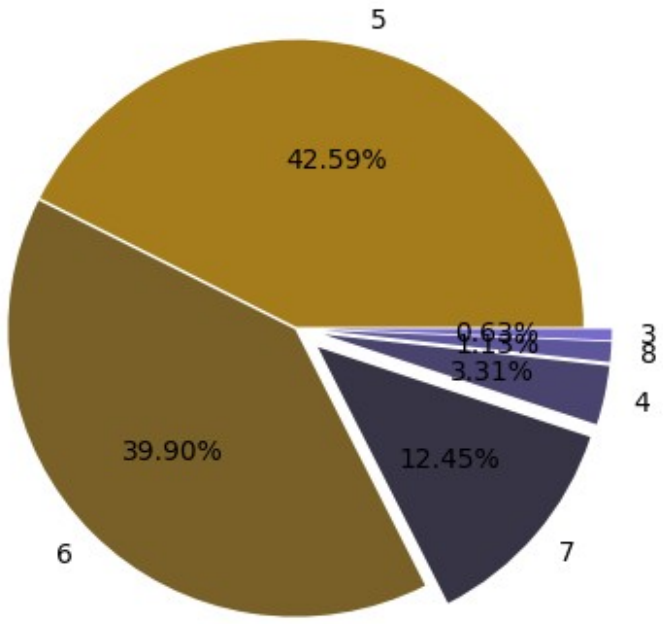
Visualization

Uni-variant Analysis

```
df['quality'].unique()
```

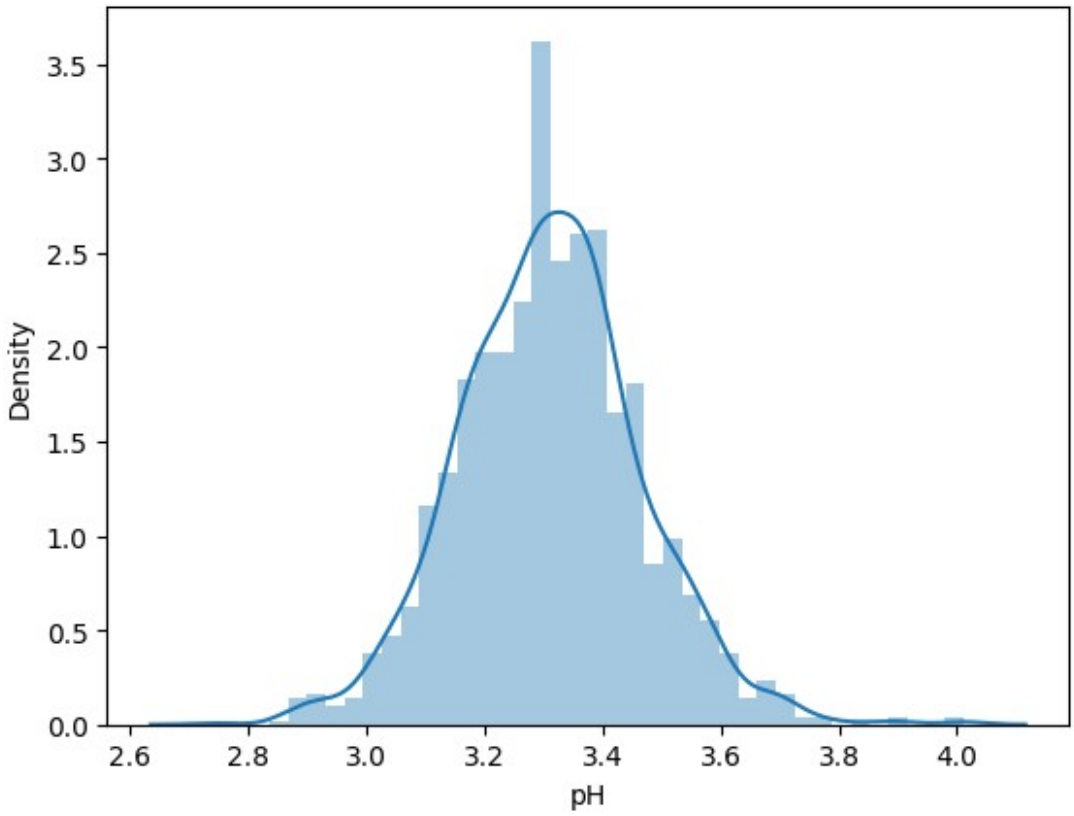
```
array([5, 6, 7, 4, 8, 3])
```

```
plt.pie(df['quality'].value_counts(),autopct='%0.2f%
%',colors=["#a57c1b", "#786028", "#363445", "#48446e", "#5e569b",
"#776bcd"],labels=df['quality'].value_counts().keys(),explode=[0.005,0
.005,0.1,0.1,0.1,0.1])
plt.show()
```



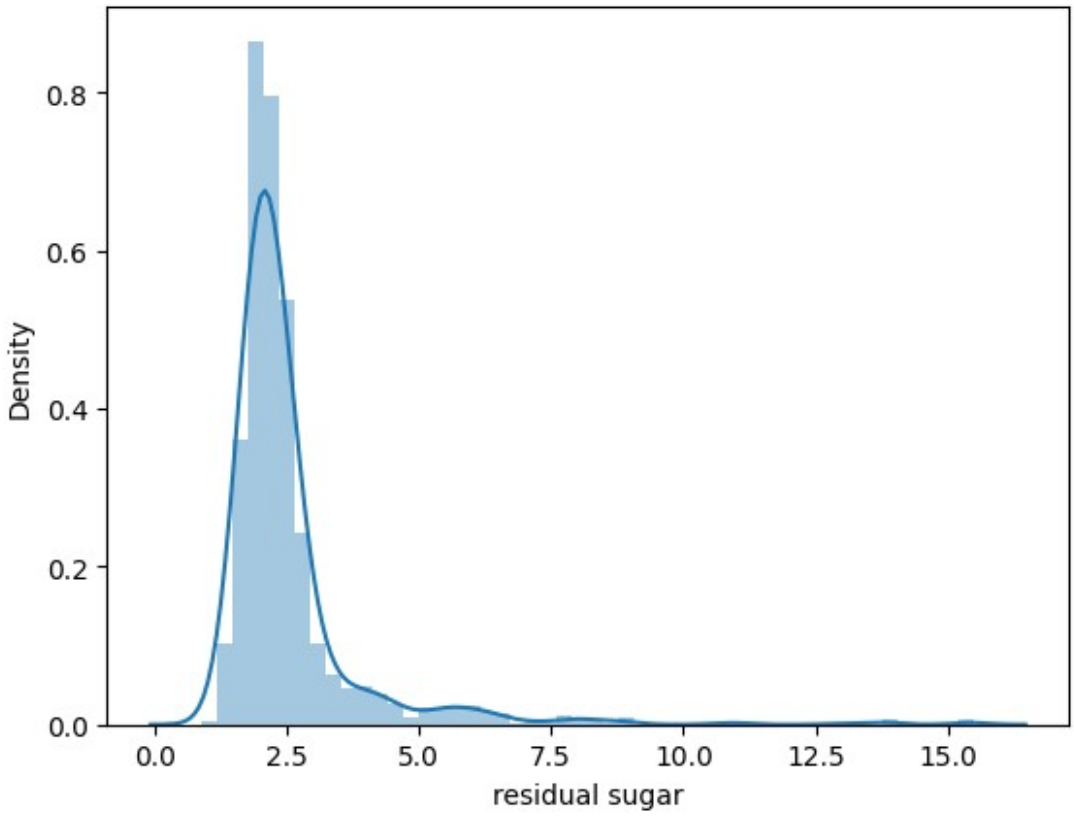

```
sns.distplot(df['pH'])
```

```
<ipython-input-11-d020e64af2d2>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['pH'])

<Axes: xlabel='pH', ylabel='Density'>
```



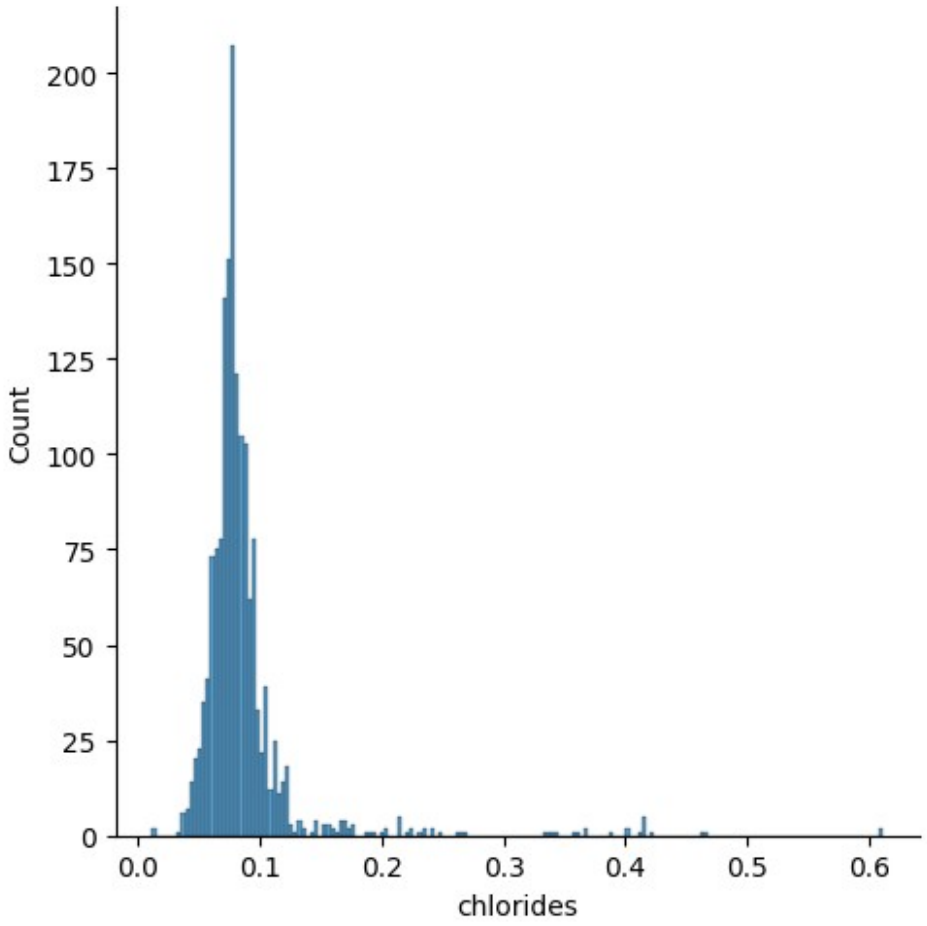

```
sns.distplot(df['residual sugar'])
```

```
<ipython-input-12-17c4014efccf>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['residual sugar'])

<Axes: xlabel='residual sugar', ylabel='Density'>
```



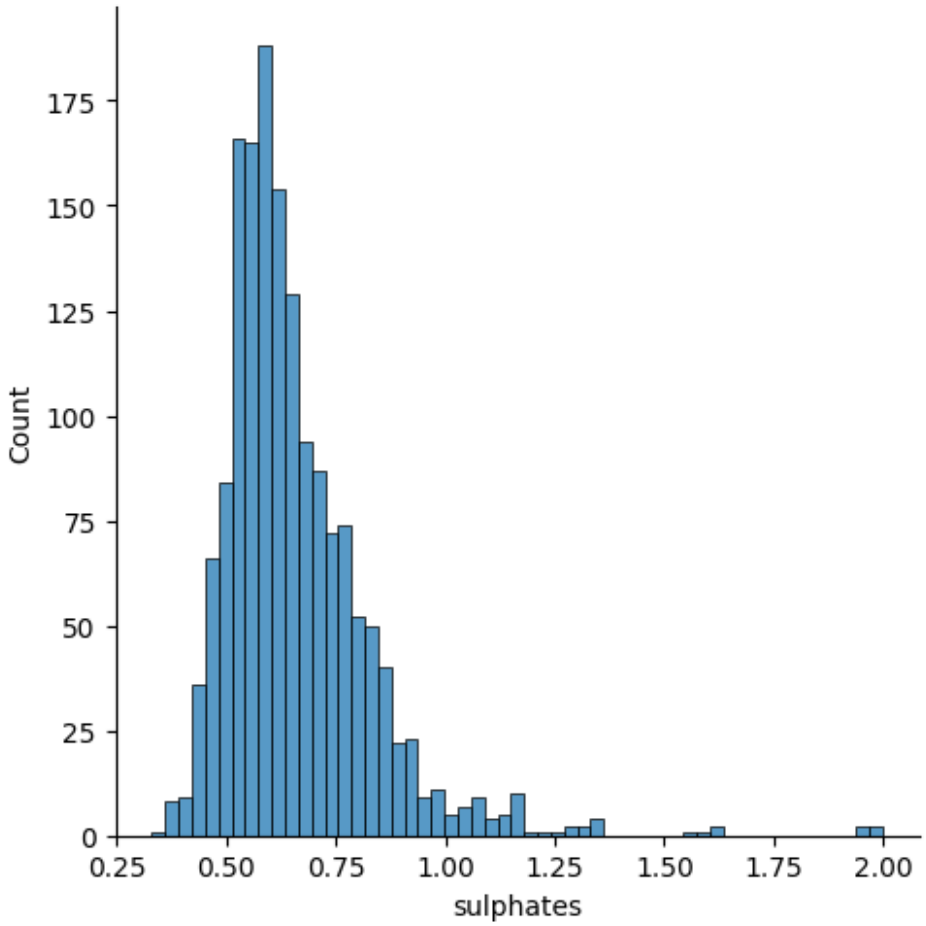

```
sns.displot(df.chlorides)
```

```
<seaborn.axisgrid.FacetGrid at 0x78d17b4dff10>
```




```
sns.displot(df.sulphates)
```

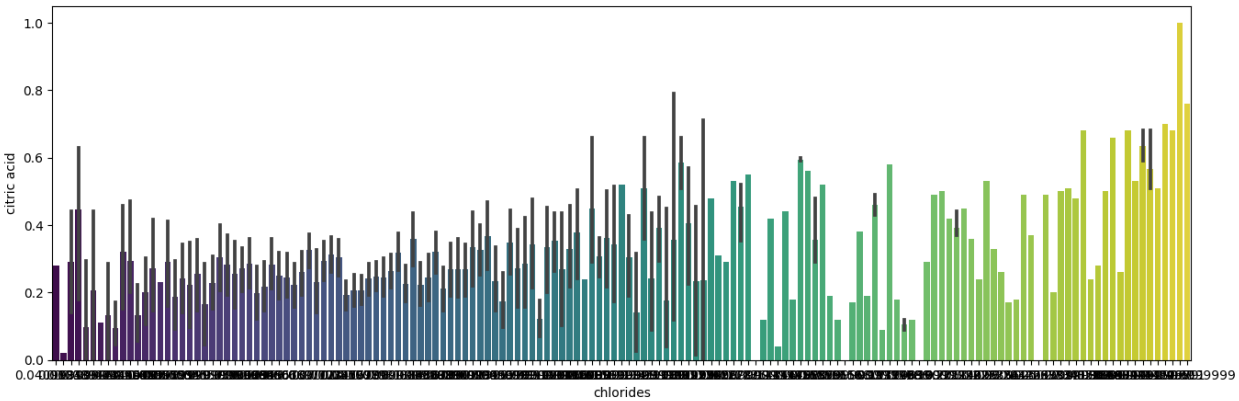
```
<seaborn.axisgrid.FacetGrid at 0x78d177089ff0>
```



Bivariant Analysis

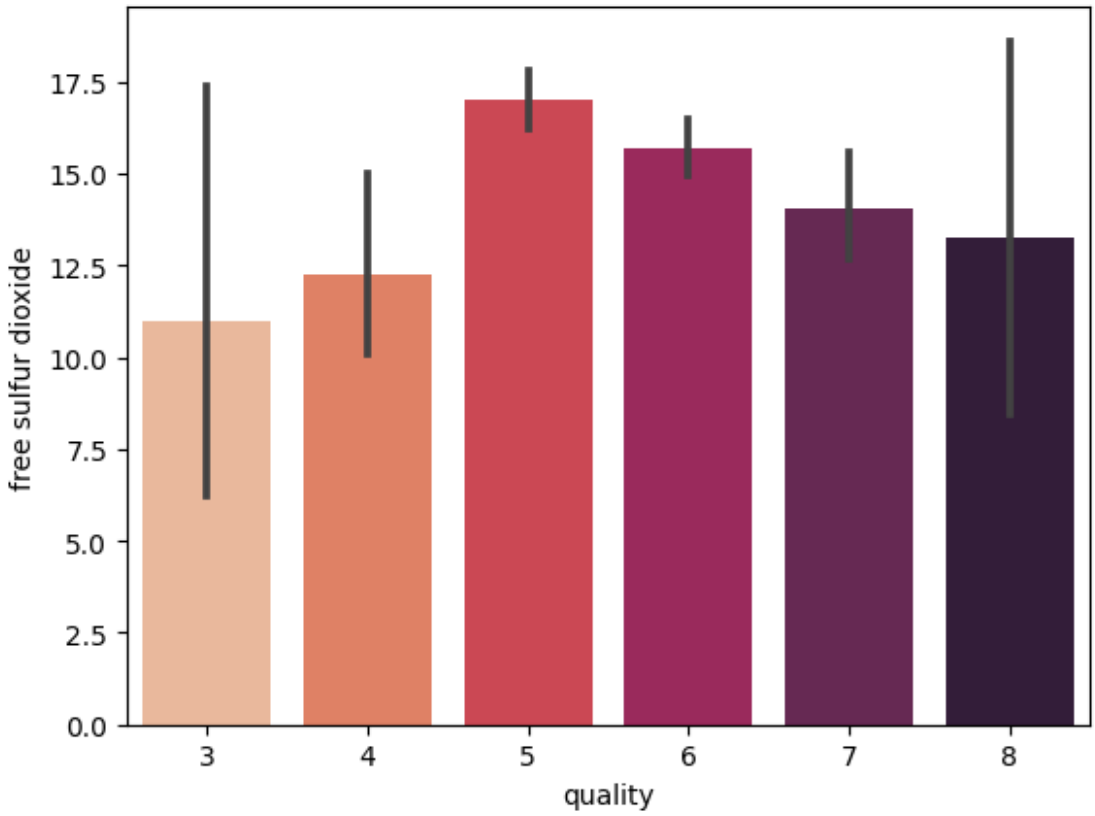
```
plt.figure(figsize=(16,5))
sns.barplot(x='chlorides',y='citric acid',data=df,palette='viridis')
```

```
<Axes: xlabel='chlorides', ylabel='citric acid'>
```



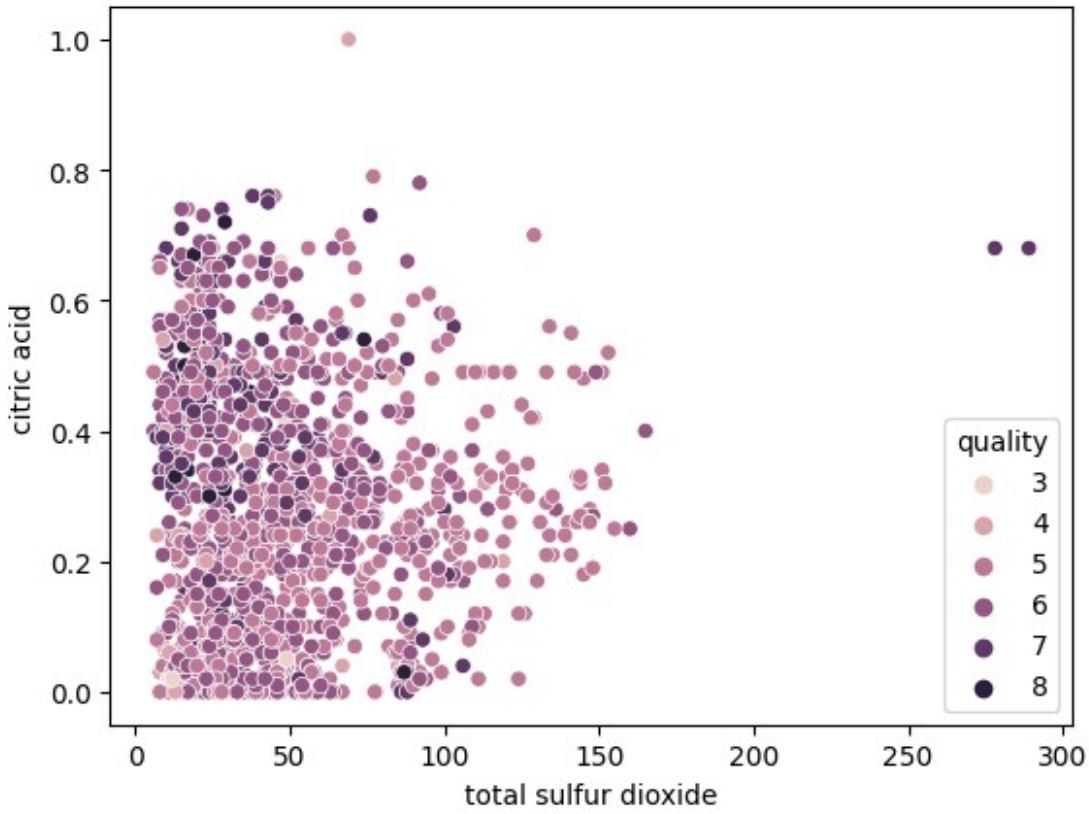
```
#plt.figure(figsize=(16,5))
sns.barplot(y='free sulfur
dioxide',x='quality',data=df,palette='rocket_r')
```

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



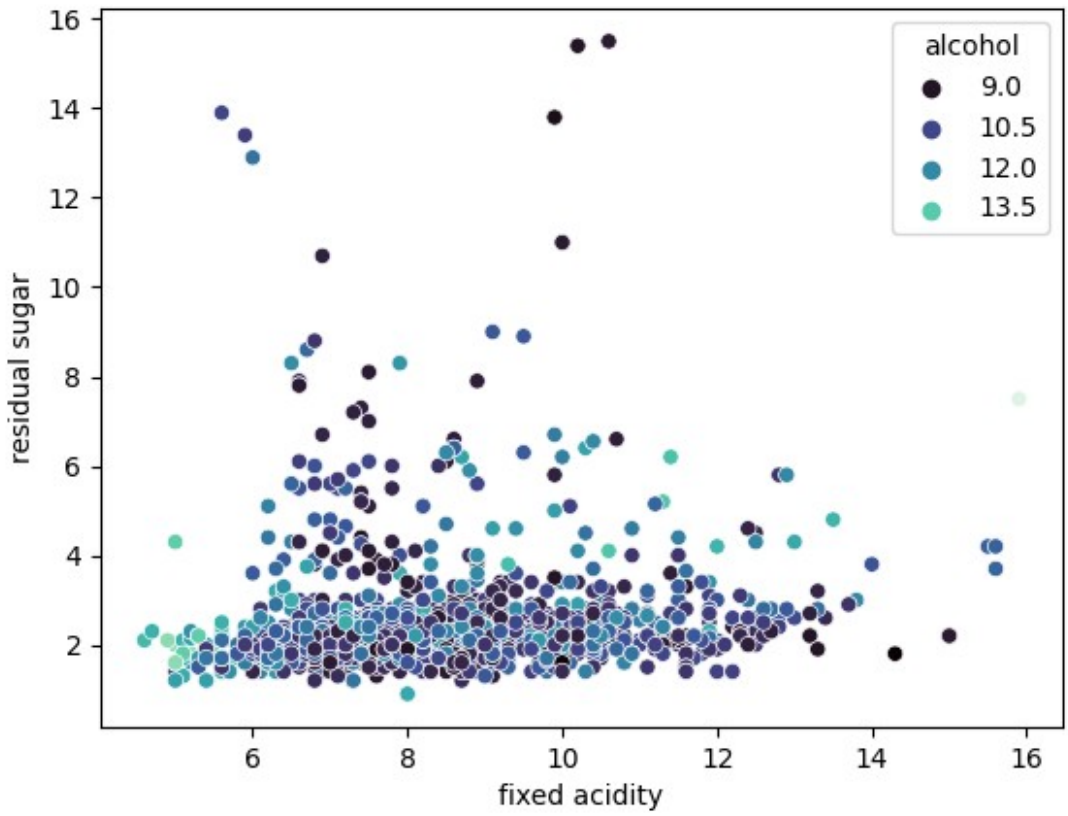

```
sns.scatterplot(x='total sulfur dioxide',y='citric
acid',data=df,hue='quality')
```

```
<Axes: xlabel='total sulfur dioxide', ylabel='citric acid'>
```



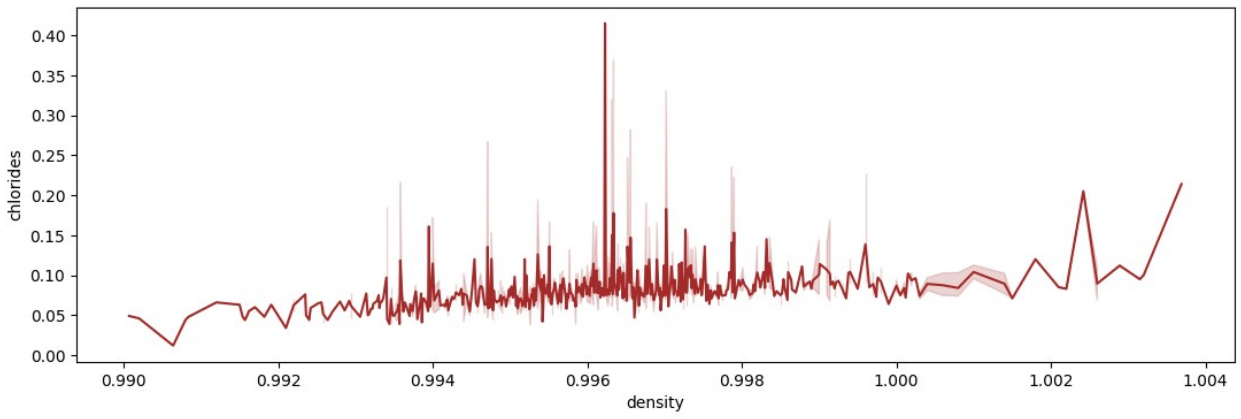

```
sns.scatterplot(x='fixed acidity',y='residual
sugar',data=df,hue='alcohol',palette='mako')
#colors=["#3c4e4b", "#466964", "#599e94", "#6cd4c5"]
```

```
<Axes: xlabel='fixed acidity', ylabel='residual sugar'>
```



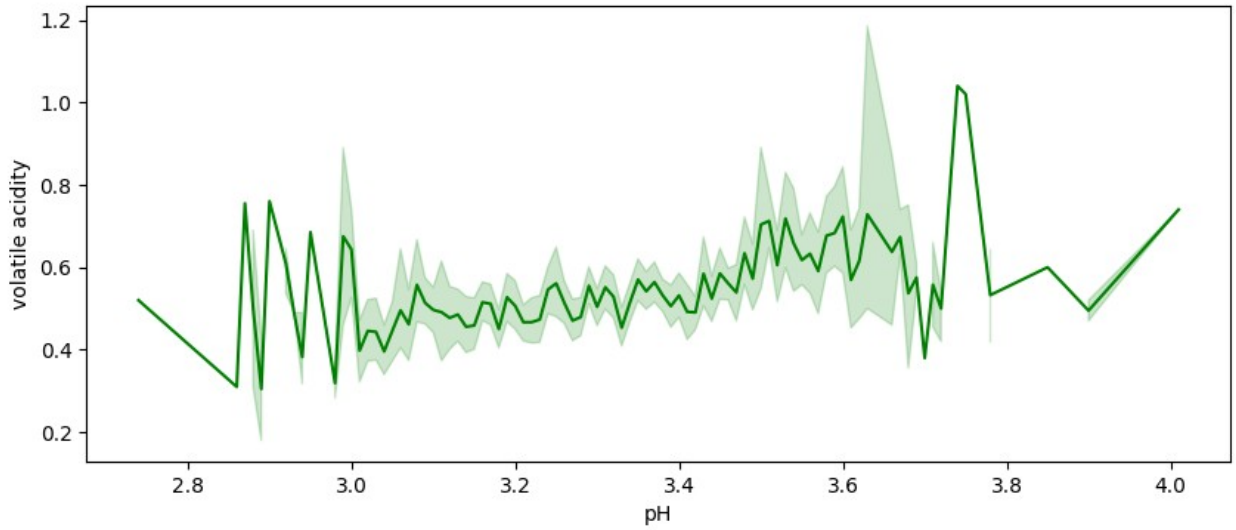

```
plt.figure(figsize=(13,4))
sns.lineplot(x='density',y='chlorides',data=df,color='brown')
```

```
<Axes: xlabel='density', ylabel='chlorides'>
```




```
plt.figure(figsize=(10,4))
sns.lineplot(x='pH',y='volatile acidity',data=df,color='green')
```

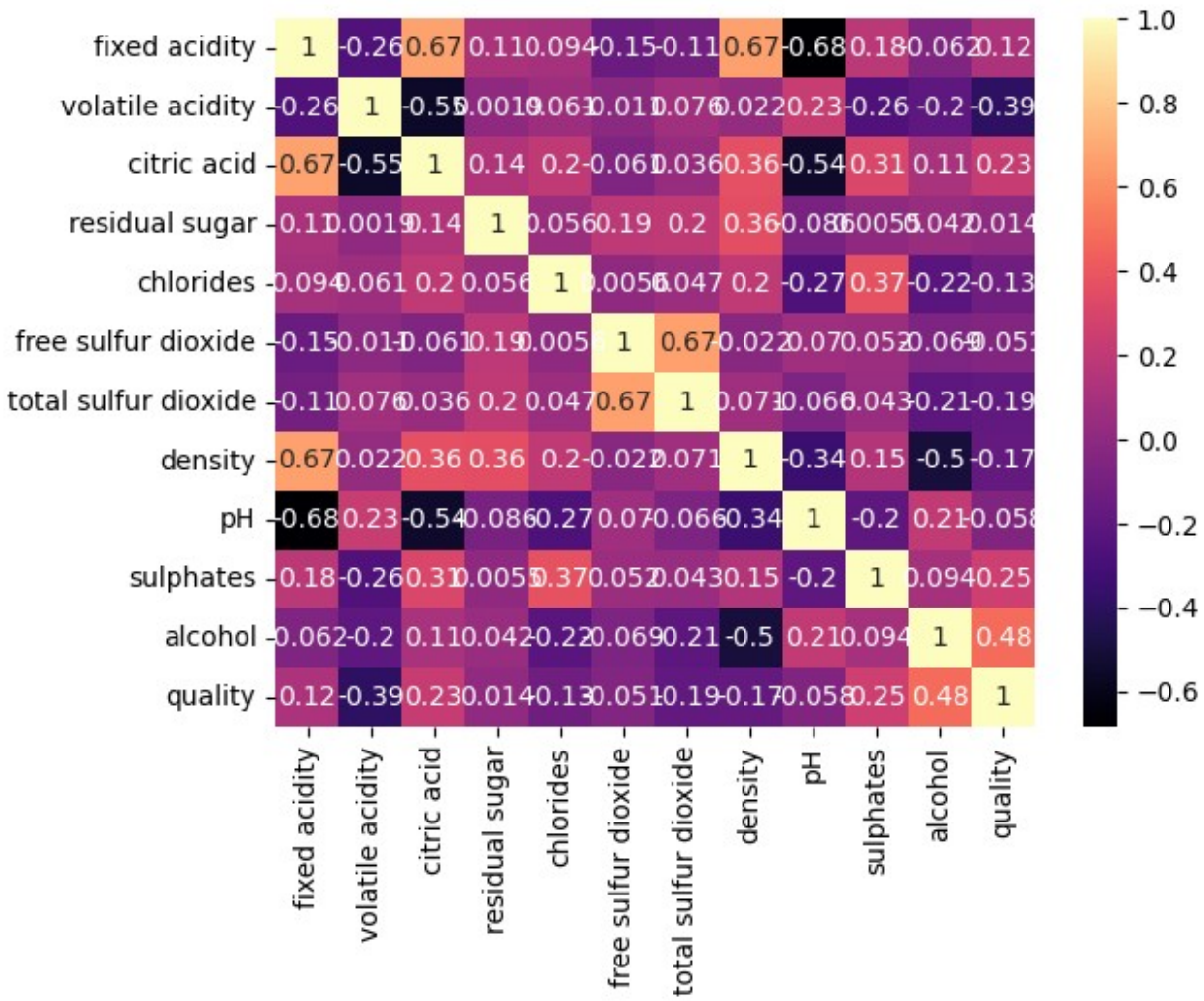
```
<Axes: xlabel='pH', ylabel='volatile acidity'>
```



Multi Variant Analysis

```
sns.heatmap(df.corr(),annot=True,cmap='magma')
```

```
<Axes: >
```



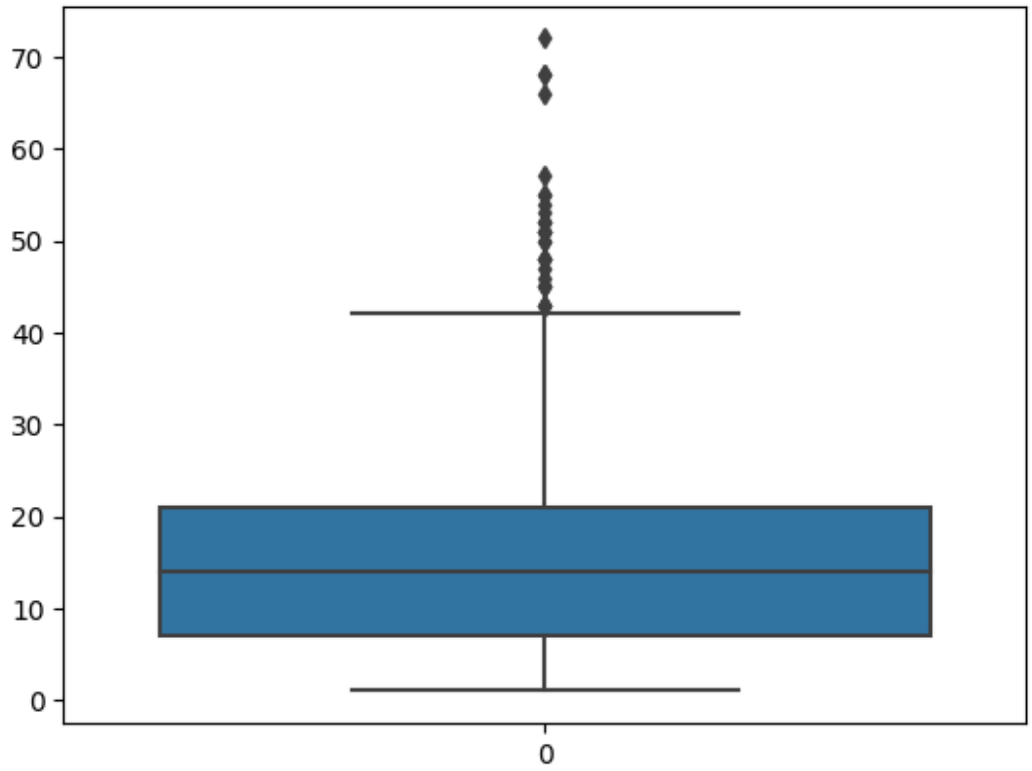

```
df.corr().quality.sort_values(ascending=False)
```

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

Outliers Checking

```
df=pd.read_csv('/content/winequality-red.csv')
sns.boxplot(df['free sulfur dioxide'])
```

```
<Axes: >
```




```
a=['citric acid']
for column in a:
  q1=df[column].quantile(0.25)
  q3=df[column].quantile(0.75)
  i=q3-q1
  upl=q3+1.5*i
  lpl=q1-1.5*i
  a=df.median()

df[column]=np.where(df[column]>upl,upl,np.where(df[column]<lpl,lpl,df[
column]))
upl=1.5
df['sulphates']=np.where(df['sulphates']>upl,upl,df[column])
q1=df['quality'].quantile(0.25)
q3=df['quality'].quantile(0.75)
upl=q3+1.5*(q3-q1)
df['quality']=np.where(df['quality']==3,upl,df['quality'])
q1=df['quality'].quantile(0.25)
q3=df['quality'].quantile(0.75)
upl=q3+1.5*(q3-q1)
```

```
lpl=q1-1.5*(q3-q1)
df['pH']=np.where(df['pH']<2.8,lpl,df['pH'])
df['pH']=np.where(df['pH']<3.8,df['pH'],upl)
q1=df['sulphates'].quantile(0.25)
q3=df['sulphates'].quantile(0.75)
upl=q3+1.5*(q3-q1)
lpl=q1-1.5*(q3-q1)
df['sulphates']=np.where(df['sulphates']<1.5,df['sulphates'],upl)
mask = (df['total sulfur dioxide']>90)  & (df['quality']==7)
unwanted_rows = df[~mask]
df = df.drop(unwanted_rows.index)
df = df.drop(df[df['volatile acidity']>1.1].index)
```

Independent and dependent variable split

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

```
     fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
16             8.5             0.280         0.56             1.8
0.092
198            5.4             0.835         0.08             1.2
0.046
230            5.2             0.480         0.04             1.6
0.054
836            6.7             0.280         0.28             2.4
0.012
837            6.7             0.280         0.28             2.4
0.012

     free sulfur dioxide  total sulfur dioxide  density    pH
sulphates  \
16                  35.0                 103.0  0.99690  3.30
0.56
198                 13.0                  93.0  0.99240  3.57
0.08
230                 19.0                 106.0  0.99270  3.54
0.04
836                 36.0                 100.0  0.99064  3.26
0.28
837                 36.0                 100.0  0.99064  3.26
0.28

     alcohol
16      10.5
198     13.0
230     12.2
```

```
836     11.7
837     11.7
```

```
y=df.quality
y.head()
```

```
16     7.0
198    7.0
230    7.0
836    7.0
837    7.0
Name: quality, dtype: float64
```

Scaling not required

Train-test spliting

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,rando
m_state=4)
```

Modeling

Linear Regression

```
from sklearn.linear_model import LinearRegression
mod=LinearRegression()
p=mod.fit(x_train,y_train)

p_pre=mod.predict(x_test)

from sklearn import metrics
metrics.r2_score(y_test,p_pre)
```

```
1.0
```

```
np.sqrt(metrics.mean_squared_error(y_true=y_test,y_pred=p_pre))
```

```
0.0
```

Decision Tree

```
#y_train = y_train.astype(float)
from sklearn.tree import DecisionTreeClassifier
mod=DecisionTreeClassifier()
d=mod.fit(x_train,y_train)

p_pre1=mod.predict(x_test)
```

```
y_test = y_test.astype(float)
from sklearn.metrics import classification_report
print(classification_report(y_true=y_test, y_pred=p_pre1))
```

```
              precision    recall  f1-score   support

         3.0       0.00      0.00      0.00         2
         4.0       0.29      0.18      0.22        11
         5.0       0.71      0.66      0.68       146
         6.0       0.63      0.63      0.63       118
         7.0       0.48      0.61      0.54        41
         8.0       0.33      0.50      0.40         2

    accuracy                           0.62       320
   macro avg       0.41      0.43      0.41       320
weighted avg       0.63      0.62      0.62       320
```

```
y_test = y_test.astype(float)
from sklearn.metrics import classification_report
print(classification_report(y_true=y_test, y_pred=p_pre1))
```

```
              precision    recall  f1-score   support

         7.0       1.00      1.00      1.00         2

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```