WINE QUALITY ANALYSIS

Exploratory data analysis & classification work with

machine learning models

---

* | INTRODUCTION 🔎

1.fixed acidity:---------

most acids involved with wine or fixed or nonvolatile (do not evaporate readily). Acidity is a characteristic determined by the total sum of acids that a sample contains. We can quantify the set of all of them in an undifferentiated way (total acidity) or in a grouped way (fixed acidity and volatile acidity). Fixed acidity corresponds to the set of low volatility organic acids such as malic, lactic, tartaric or citric acids and is inherent to the characteristics of the sample.

2.volatile acidity:-----

the amount of acetic acid in wine, which at too high of levels can lead to an unpleasant, vinegar taste. Volatile acidity corresponds to the set of short chain organic acids that can be extracted from the sample by means of a distillation process: formic acid, acetic acid, propionic acid and butyric acid.

3.citric acid:------

found in small quantities, citric acid can add 'freshness' and flavor to wines. Citric acid is a colorless weak organic acid. It occurs naturally in citrus fruits. In biochemistry, it is an intermediate in the citric acid cycle, which occurs in the metabolism of all aerobic organisms.

4.residual sugar:------

the amount of sugar remaining after fermentation stops, it's rare to find wines with less than 1 gram/liter. Residual sugar refers to the sugars left unfermented in a finished wine. It is measured by grams of sugar per litre (g/l). The amount of residual sugar affects a wine's sweetness and, in the EU, the RS level is linked to specific labelling terms

5.chlorides:-----

the amount of salt in the wine. The higher extraction of chloride during red winemaking is due to the ions extracted from skins during fermentation. Therefore, red juice should have no more than 356mg/L chloride ions so that finished wine does not exceed the maximum legal level of 606mg/L chloride(356mg/L in red juice x 1.7 = 606).

6. free sulfur dioxide:-----

the free form of SO2 exists in equilibrium between molecular SO2 (as a dissolved gas) and bisulfite ion. What is free sulphur dioxide in wine? The free sulfites are those available to react and thus exhibit both germicidal and antioxidant properties. The bound sulfites are those that have reacted (both reversibly and irreversibly) with other molecules within the wine medium. The sum of the free and bound sulfites defines the total sulfite concentration

7.free sulfur dioxide:-----

the free form of SO2 exists in equilibrium between molecular SO2 (as a dissolved gas) and bisulfite ion. What is free sulphur dioxide in wine? The free sulfites are those available to react and thus exhibit both germicidal and antioxidant properties. The bound sulfites are those that have reacted (both reversibly and irreversibly) with other molecules within the wine medium. The sum of the free and bound sulfites defines the total sulfite concentration

8.density:------

the density of water is close to that of water depending on the percent alcohol and sugar content. How do you measure the density of wine? A hydrometer is an instrument used to measure liquid density. It is a sealed glass tube with a weighted bulb at one end, winemakers use this instrument to measure density of juice, fermenting wine and completed wine in relation to pure water. This ratio is called specific gravity (SG).

9.pH:--------

describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic); most wines are between 3-4. What is a high pH in wine? Wines which have higher pH values (>3.65) have a series of potential challenges during vinification and aging. First, high pH wines have an increased chance of microbial spoilage. Traditionally, sulfur dioxide (often in the form of potassium metabisulfite) is used to keep wines stable during aging.

10.sulphates:------

a wine additive which can contribute to sulfur dioxide gas (S02) levels, wich acts as an antimicrobial. Wine sulfites are naturally occurring at low levels in all wines, and are one of the thousands of chemical by-products created during the fermentation process. However, sulfites are also added by the winemaker to preserve and protect the wine from bacteria and yeast-laden invasions. For some, sulfur allergies may be associated with headaches and stuffy sinuses after a glass or two of wine. It is a a wine additive which can contribute to sulfur dioxide gas (S02) levels, wich acts as an antimicrobial.

11.Alcohol:-----

this is the percent alcohol content of the wine

12.quality:-------

output variable (based on sensory data, score between 3 and 8)

*| IMPORT NECESSARY LIBRARIES

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib.colors import ListedColormap
from sklearn.metrics import plot_confusion_matrix
from scipy.stats import norm, boxcox
from collections import Counter
from scipy import stats

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [2]:
```
from ydata_profiling import ProfileReport
```

Loading Dataset

In [3]:
```
data = pd.read_csv('WQ data.csv')
```

Exploratory Data Analysis

1) Using Manual Methods

In [4]:
```
data.head(11)
```

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	5.0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5.0
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5.0
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6.0
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5.0
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4	5.0
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4	5.0
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0	7.0
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5	7.0
9	7.5	0.50	0.36	6.1	0.071	17.0	NaN	0.9978	3.35	0.80	10.5	5.0
10	6.7	0.58	0.08	1.8	0.097	15.0	65.0	0.9959	3.28	0.54	9.2	5.0

In [5]:
```
dataset = data.dropna()
```

In [6]:
```
dataset.head(11)
```

Out[6]:

	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	5.0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5.0
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5.0
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6.0
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5.0
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4	5.0
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4	5.0
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0	7.0
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5	7.0
10	6.7	0.58	0.08	1.8	0.097	15.0	65.0	0.9959	3.28	0.54	9.2	5.0
11	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	5.0

In [7]:
```
dataset.shape
```

Out[7]:
```
(1596, 12)
```

In [8]:
```
df = dataset.copy()
df.head(n = 10).style.background_gradient(cmap = "Purples_r")
```

Out[8]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alco
0	7.400000	0.700000	0.000000	1.900000	0.076000	11.000000	34.000000	0.997800	3.510000	0.560000	9.400
1	7.800000	0.880000	0.000000	2.600000	0.098000	25.000000	67.000000	0.996800	3.200000	0.680000	9.800
2	7.800000	0.760000	0.040000	2.300000	0.092000	15.000000	54.000000	0.997000	3.260000	0.650000	9.800
3	11.200000	0.280000	0.560000	1.900000	0.075000	17.000000	60.000000	0.998000	3.160000	0.580000	9.800
4	7.400000	0.700000	0.000000	1.900000	0.076000	11.000000	34.000000	0.997800	3.510000	0.560000	9.400
5	7.400000	0.660000	0.000000	1.800000	0.075000	13.000000	40.000000	0.997800	3.510000	0.560000	9.400
6	7.900000	0.600000	0.060000	1.600000	0.069000	15.000000	59.000000	0.996400	3.300000	0.460000	9.400
7	7.300000	0.650000	0.000000	1.200000	0.065000	15.000000	21.000000	0.994600	3.390000	0.470000	10.000
8	7.800000	0.580000	0.020000	2.000000	0.073000	9.000000	18.000000	0.996800	3.360000	0.570000	9.500
10	6.700000	0.580000	0.080000	1.800000	0.097000	15.000000	65.000000	0.995900	3.280000	0.540000	9.200

In the code block above, we load the dataset. Then, just in case, we get a copy of the dataset. Because in some cases it may be necessary to use the original dataset.

*|INITIAL INFORMATION ABOUT DATASET ✅

1. | Descriptive Statistics of Numeric Variables

In [9]:
```
dataset.describe().T.style.background_gradient(cmap = "magma")
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
fixed acidity	1596.000000	8.321366	1.742121	4.600000	7.100000	7.900000	9.200000	15.900000
volatile acidity	1596.000000	0.527666	0.179154	0.120000	0.390000	0.520000	0.640000	1.580000
citric acid	1596.000000	0.271128	0.194847	0.000000	0.090000	0.260000	0.420000	1.000000
residual sugar	1596.000000	2.536936	1.408341	0.900000	1.900000	2.200000	2.600000	15.500000
chlorides	1596.000000	0.087487	0.047107	0.012000	0.070000	0.079000	0.090000	0.611000
free sulfur dioxide	1596.000000	15.882206	10.467380	1.000000	7.000000	14.000000	21.000000	72.000000
total sulfur dioxide	1596.000000	46.431078	32.893072	6.000000	22.000000	38.000000	62.000000	289.000000
density	1596.000000	0.996745	0.001889	0.990070	0.995600	0.996745	0.997842	1.003690
pH	1596.000000	3.498716	0.080297	2.740000	3.520000	3.520000	3.520000	3.900000
sulphates	1596.000000	0.658189	0.169587	0.330000	0.550000	0.620000	0.730000	2.000000
alcohol	1596.000000	10.424217	1.066046	8.400000	9.500000	10.200000	11.100000	14.900000
quality	1596.000000	5.636591	0.807963	3.000000	5.000000	6.000000	6.000000	8.000000

The average value of fixed acidity is 8.32, the highest value is 15.9

The average value of volatile acidity is 0.52, the highest value is 1.58

The average value of citric acid is 0.27, the highest value is 1

The average value of residual sugar is 2.53, the highest value is 15.5

The average value of chlorides is 0.08, the highest value is 0.61

The average value of free sulfur dioxide is 15.87, the highest value is 72

The average value of total sulfur dioxide is 46.46, the highest value is 289

The average value of density is 0.99, the highest value is 1

The average value of pH is 3.49, the highest value is 3.9

The average value of sulphates is 0.65, the highest value is 2

The average value of alcohol is 10.42, the highest value is 14.90

The average value of quality is 5.63, the highest value is 8

2 | Get basic information

In [10]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1596 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1596 non-null   float64
 1   volatile acidity      1596 non-null   float64
 2   citric acid           1596 non-null   float64
 3   residual sugar        1596 non-null   float64
 4   chlorides             1596 non-null   float64
 5   free sulfur dioxide   1596 non-null   float64
 6   total sulfur dioxide  1596 non-null   float64
 7   density               1596 non-null   float64
 8   pH                    1596 non-null   float64
 9   sulphates             1596 non-null   float64
 10  alcohol               1596 non-null   float64
 11  quality               1596 non-null   float64
dtypes: float64(12)
memory usage: 162.1 KB
```

In [11]:
```
dataset.shape
```

Out[11]:
```
(1596, 12)
```

Brief information

It turns out that the dataset does not have null values. The dataset consists of 1596 rows and 12 columns. The data type of all variables are numeric.

3. | Check null Values

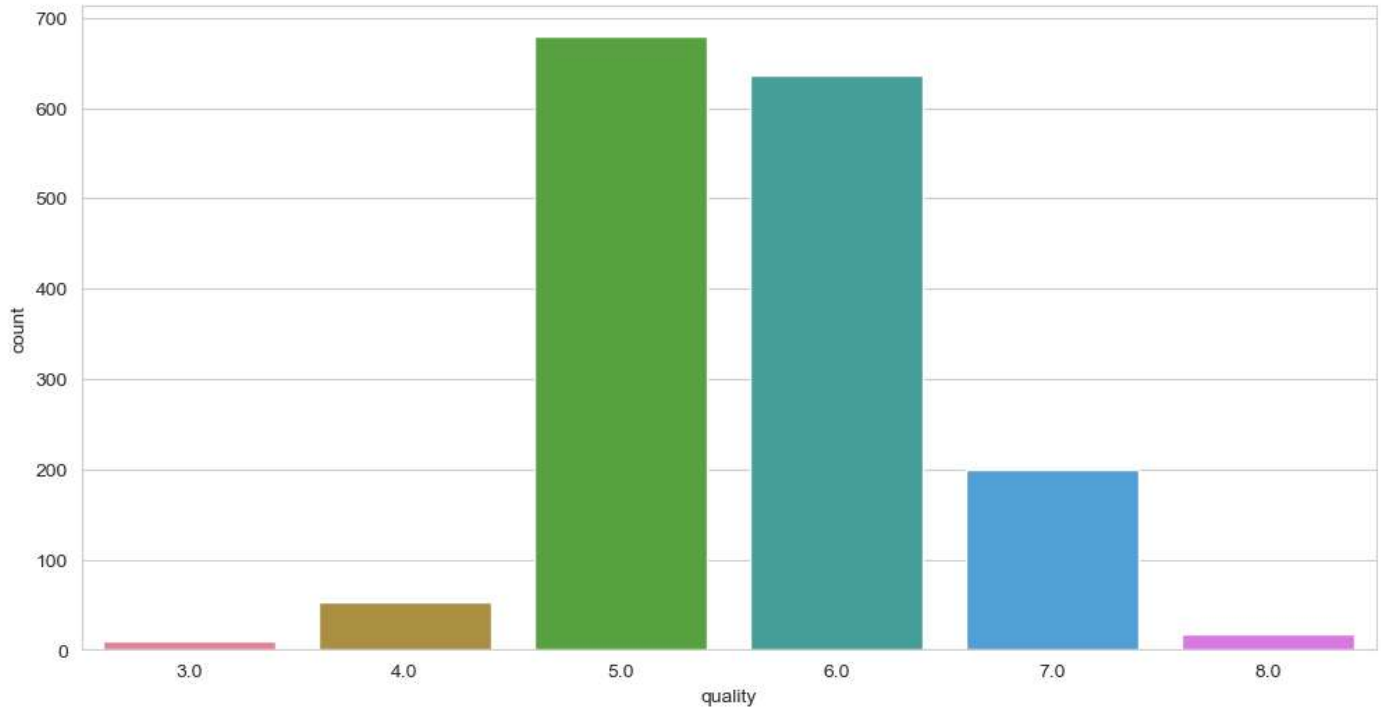
In [12]:
```
dataset.isnull().values.any()
```

Out[12]: False

There are not any null values in the dataset. Total "null" values in the dataset is zero

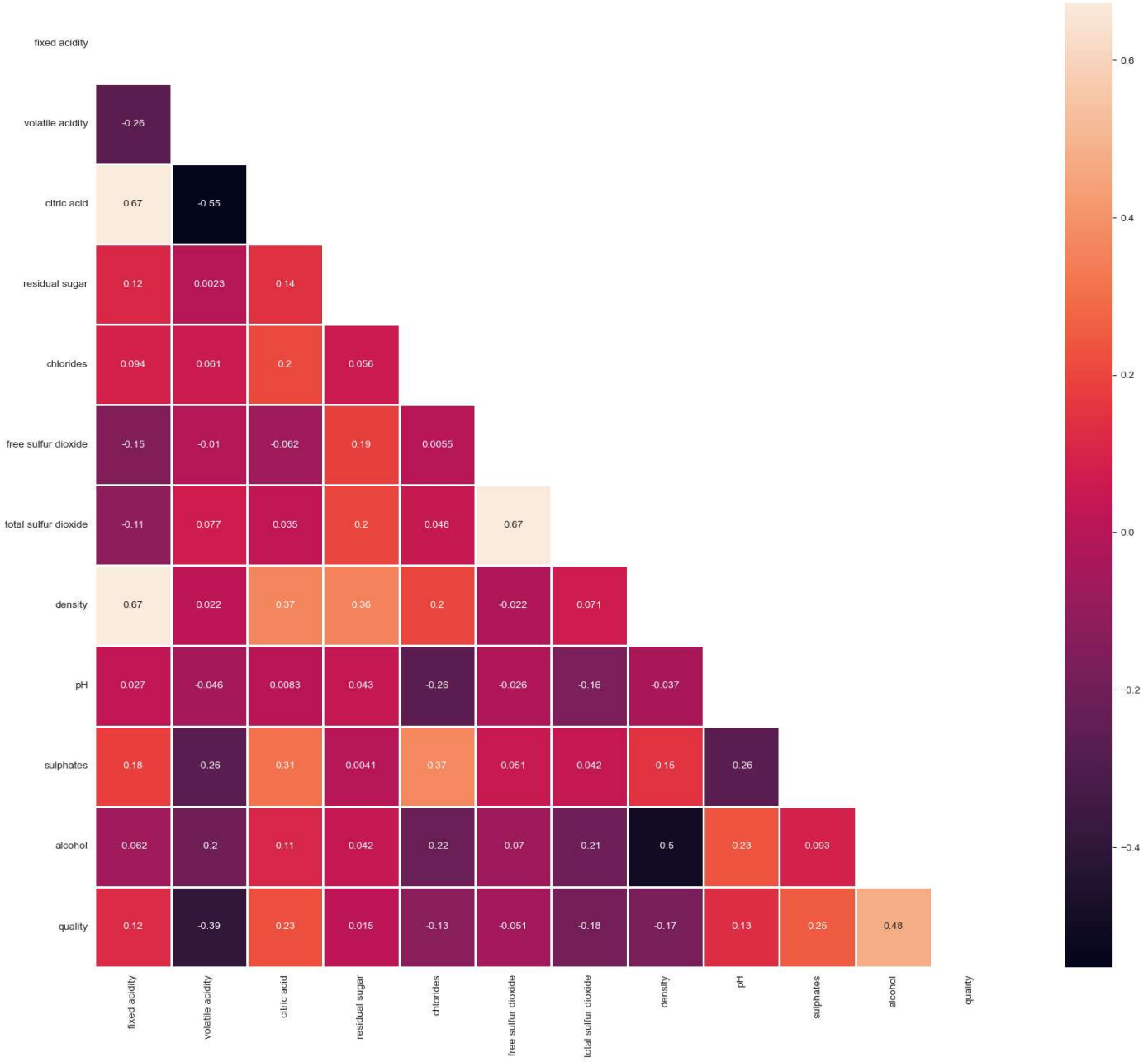
* Plotting Count for Qualities

In [13]:
```
sns.set_style('whitegrid')
plt.figure(figsize=(12, 6))
sns.countplot(x="quality", data=dataset, palette='husl');
```



* Finding Correlation among the variables

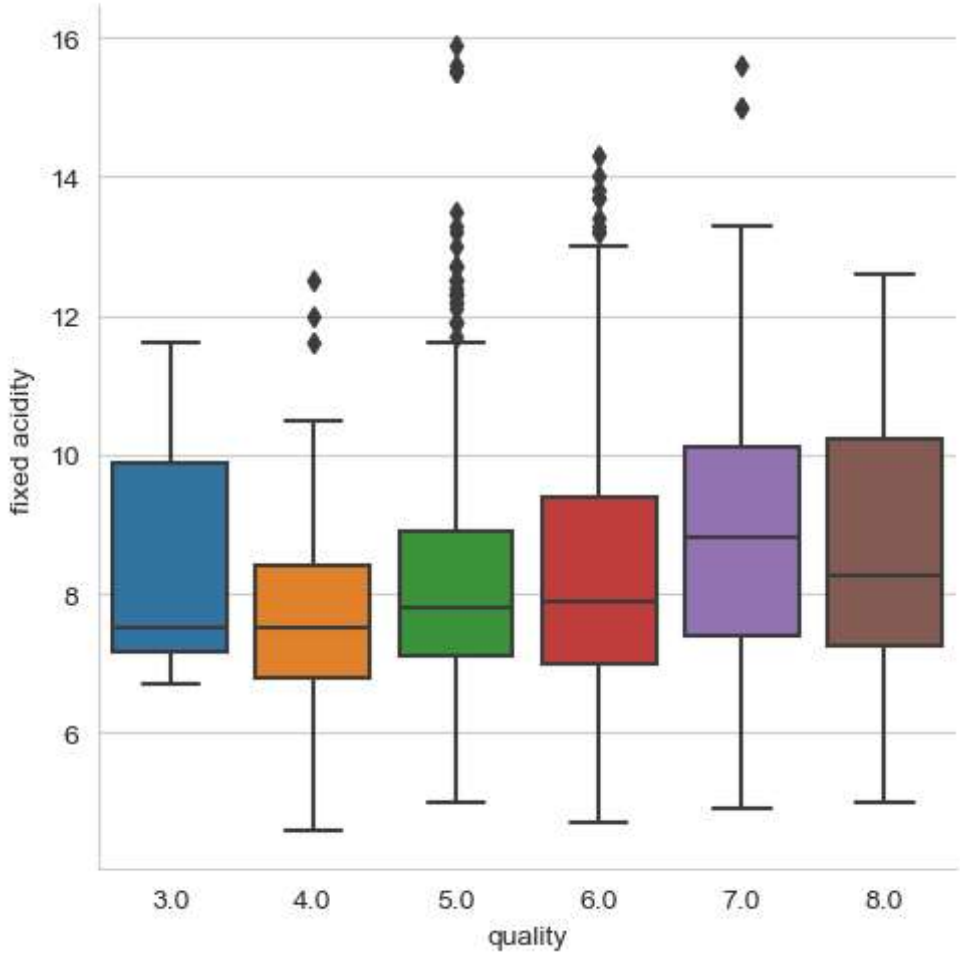
In [14]:
```
plt.figure(figsize=(20, 17))
matrix = np.triu(dataset.corr())
sns.heatmap(dataset.corr(), annot=True,
            linewidth=.8, mask=matrix, cmap="rocket");
```



* Visualising Numerical Data

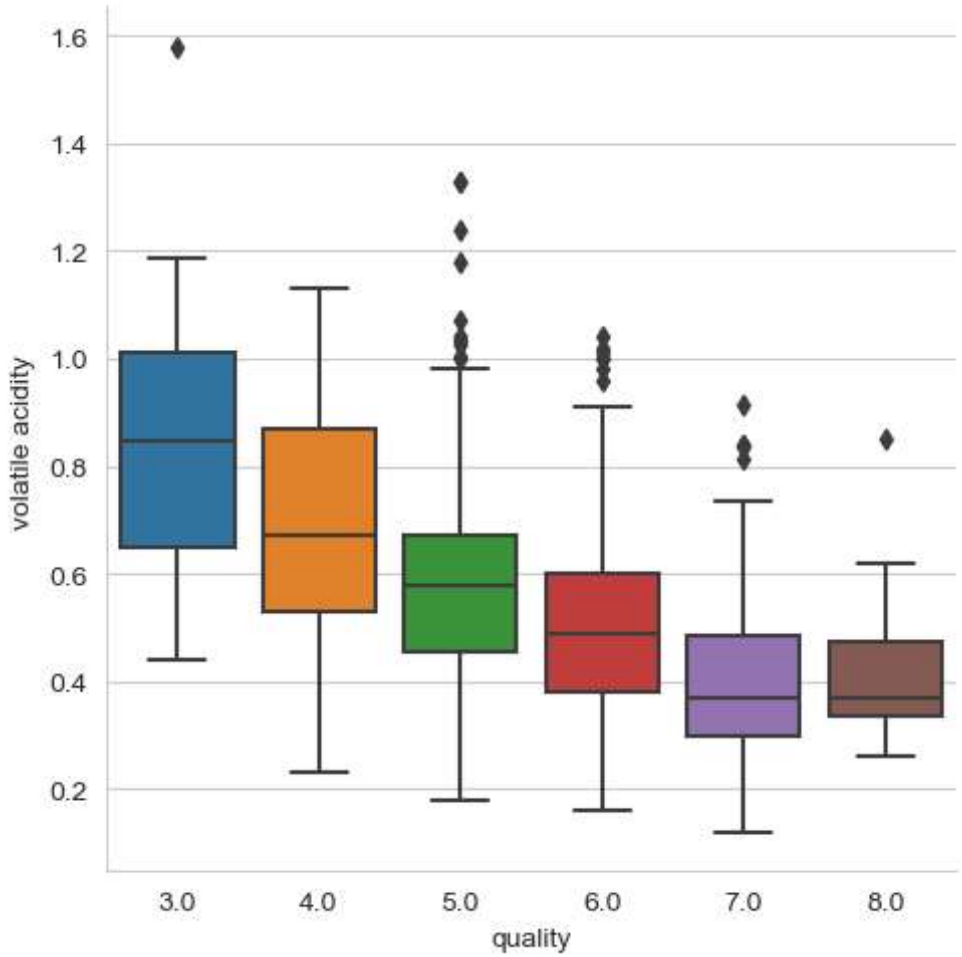
In [15]:
```
sns.catplot(x="quality", y="fixed acidity", data=dataset, kind="box")
```

Out[15]:
```
<seaborn.axisgrid.FacetGrid at 0x2ee9b409130>
```



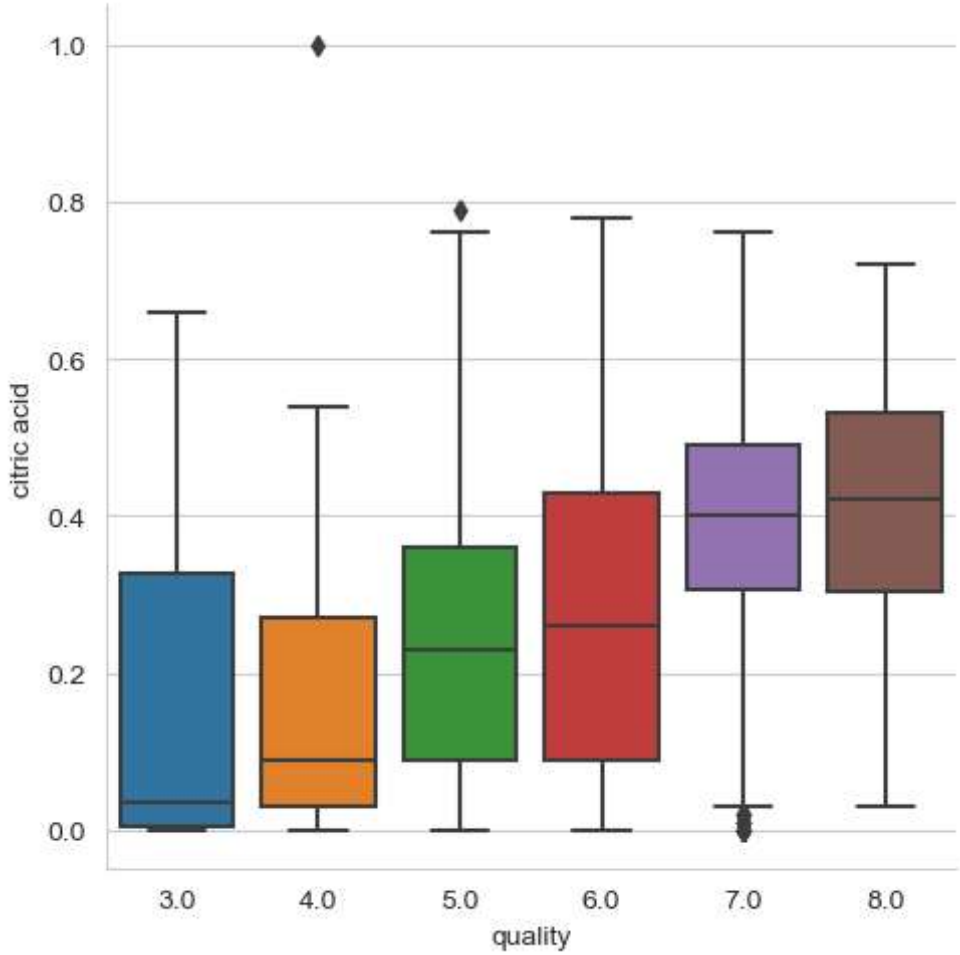
In [16]:
```
sns.catplot(x="quality", y="volatile acidity", data=dataset, kind="box")
```

Out[16]:
```
<seaborn.axisgrid.FacetGrid at 0x2ee9b427f70>
```



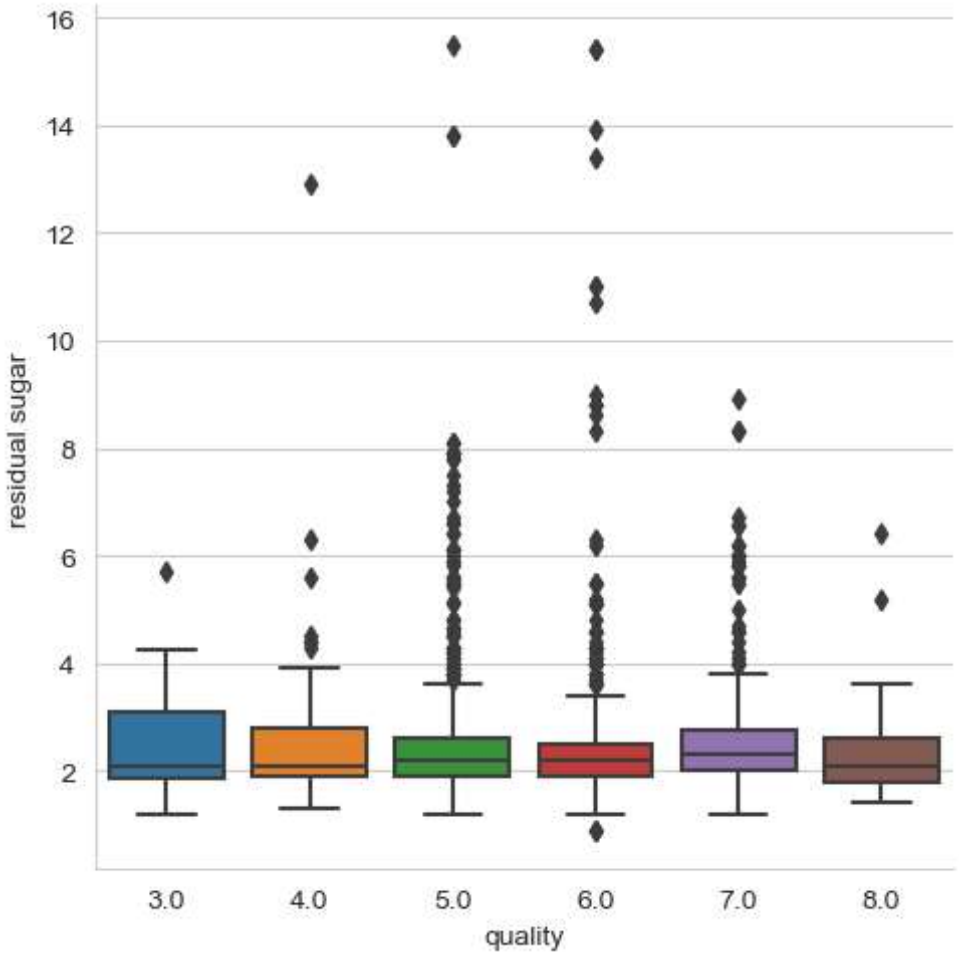
In [17]:
```
sns.catplot(x="quality", y="citric acid", data=dataset, kind="box")
```

Out[17]:
```
<seaborn.axisgrid.FacetGrid at 0x2ee9b80e3d0>
```



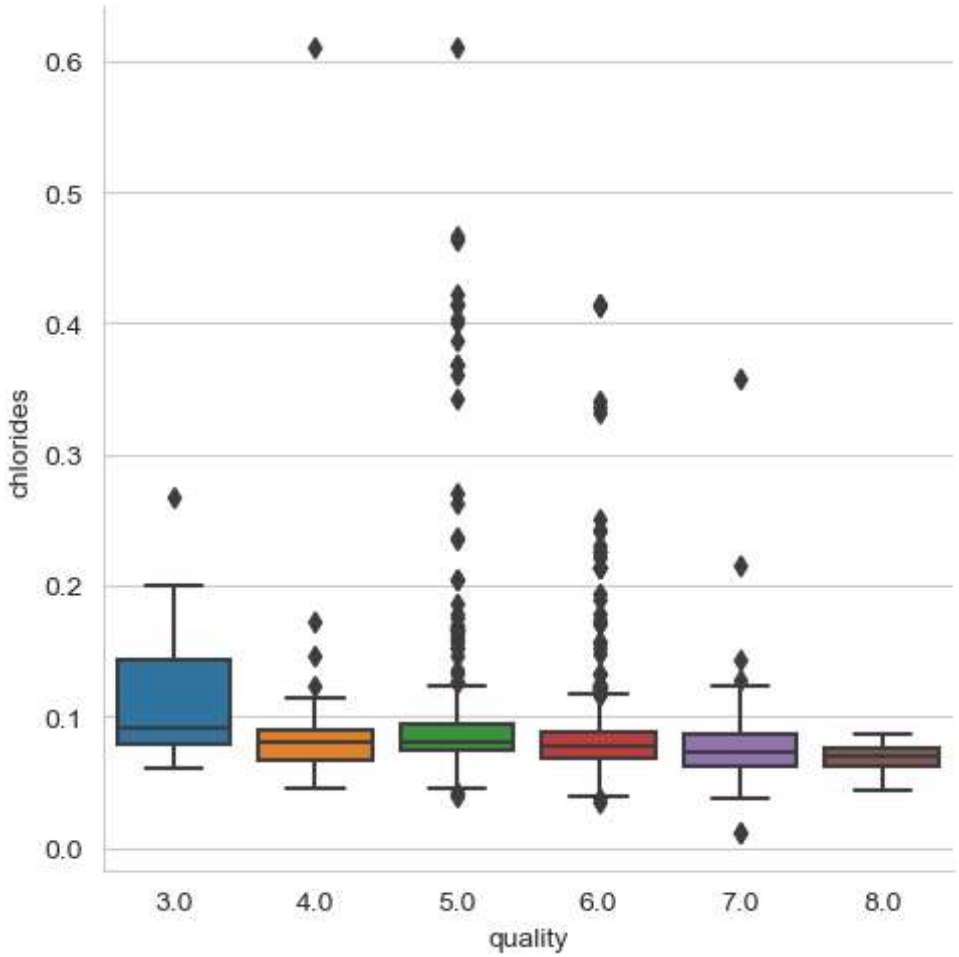
In [18]:
```
sns.catplot(x="quality", y="residual sugar", data=dataset, kind="box")
```

Out[18]:
```
<seaborn.axisgrid.FacetGrid at 0x2ee9b42a160>
```



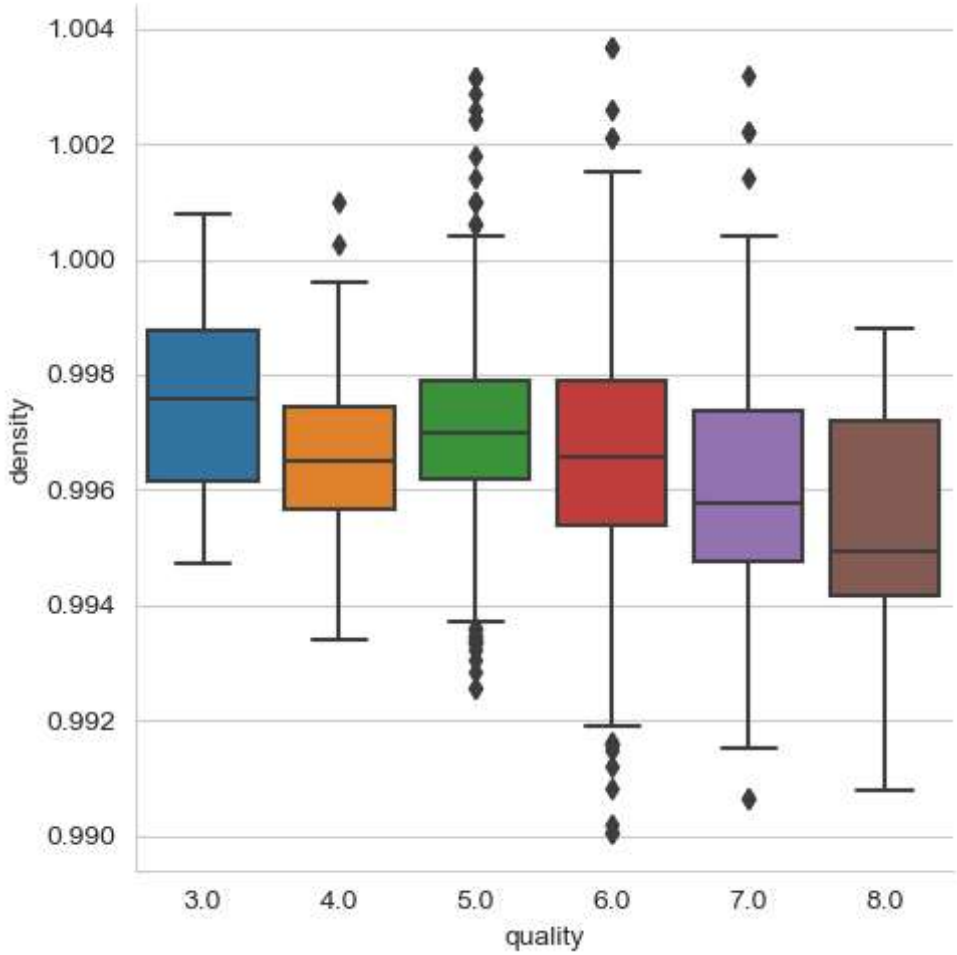
In [19]:
```
sns.catplot(x="quality", y="chlorides", data=dataset, kind="box")
```

Out[19]:
```
<seaborn.axisgrid.FacetGrid at 0x2ee9b475550>
```



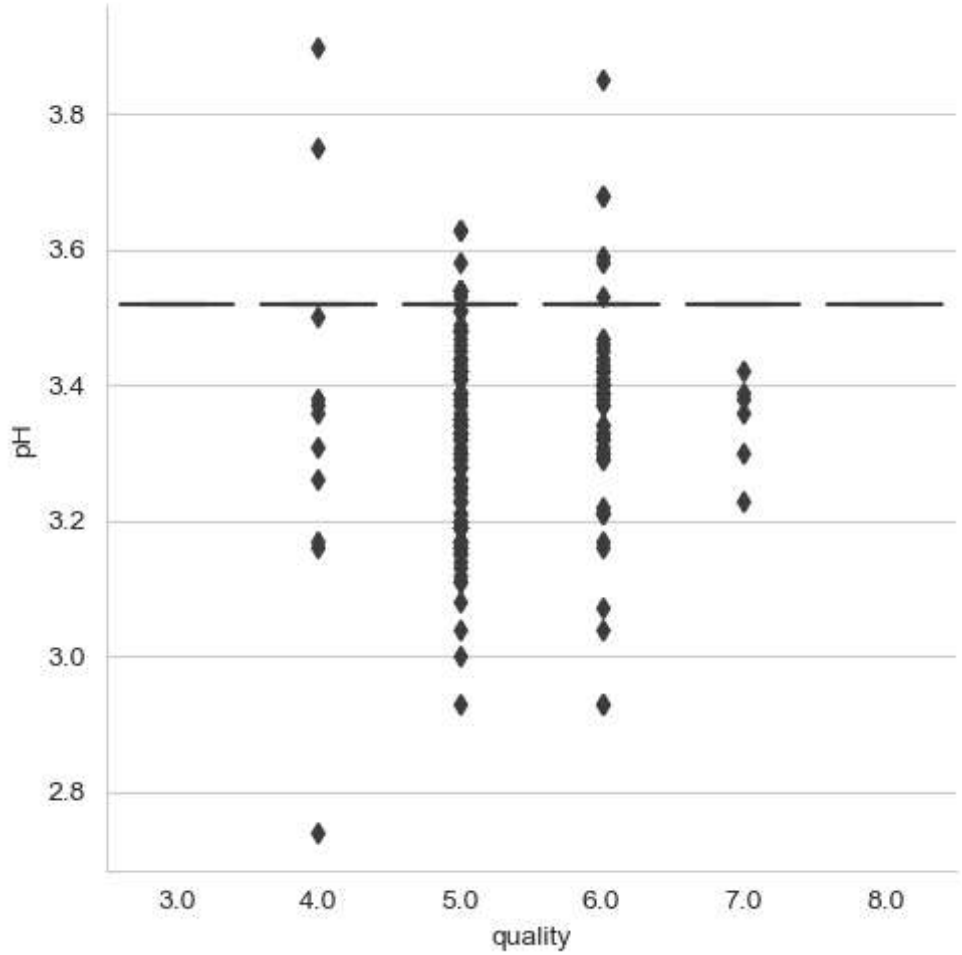
In [20]:
```
sns.catplot(x="quality", y="density", data=dataset, kind="box")
```

Out[20]:
```
<seaborn.axisgrid.FacetGrid at 0x2ee9b8737f0>
```



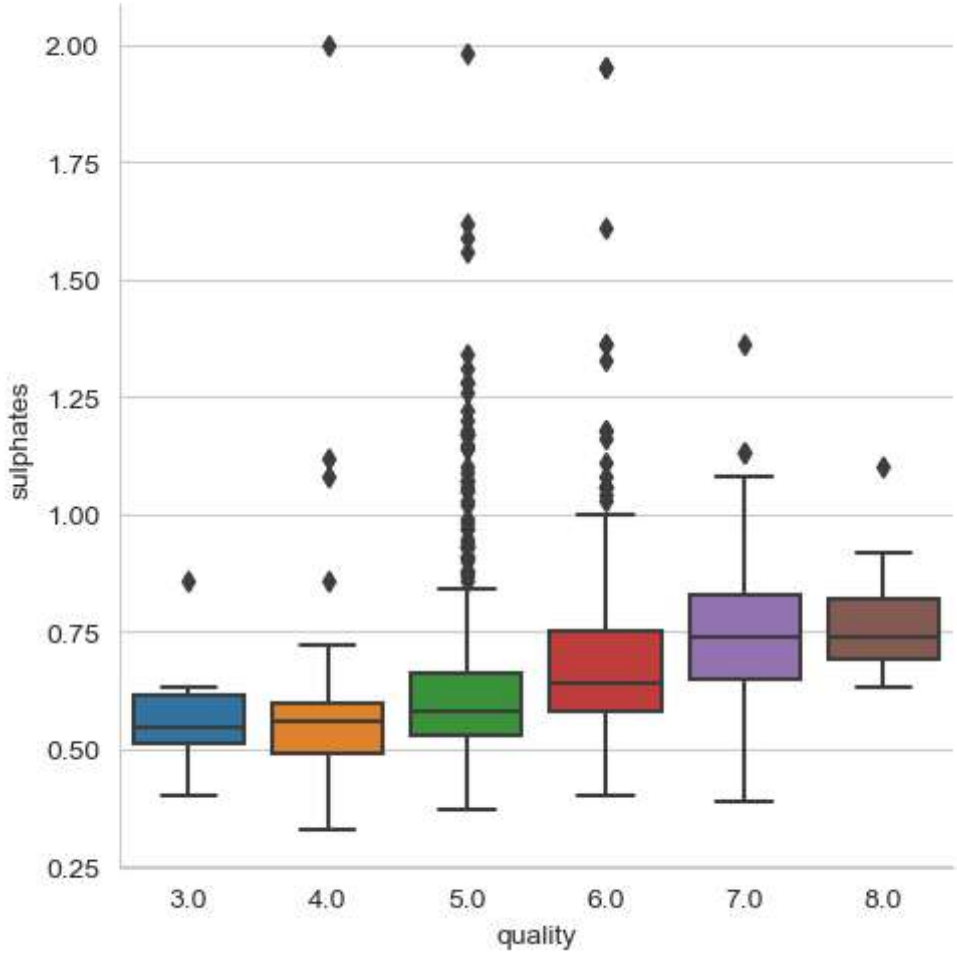
In [21]:
```
sns.catplot(x="quality", y="pH", data=dataset, kind="box")
```

Out[21]:
```
<seaborn.axisgrid.FacetGrid at 0x2ee9b4e3220>
```

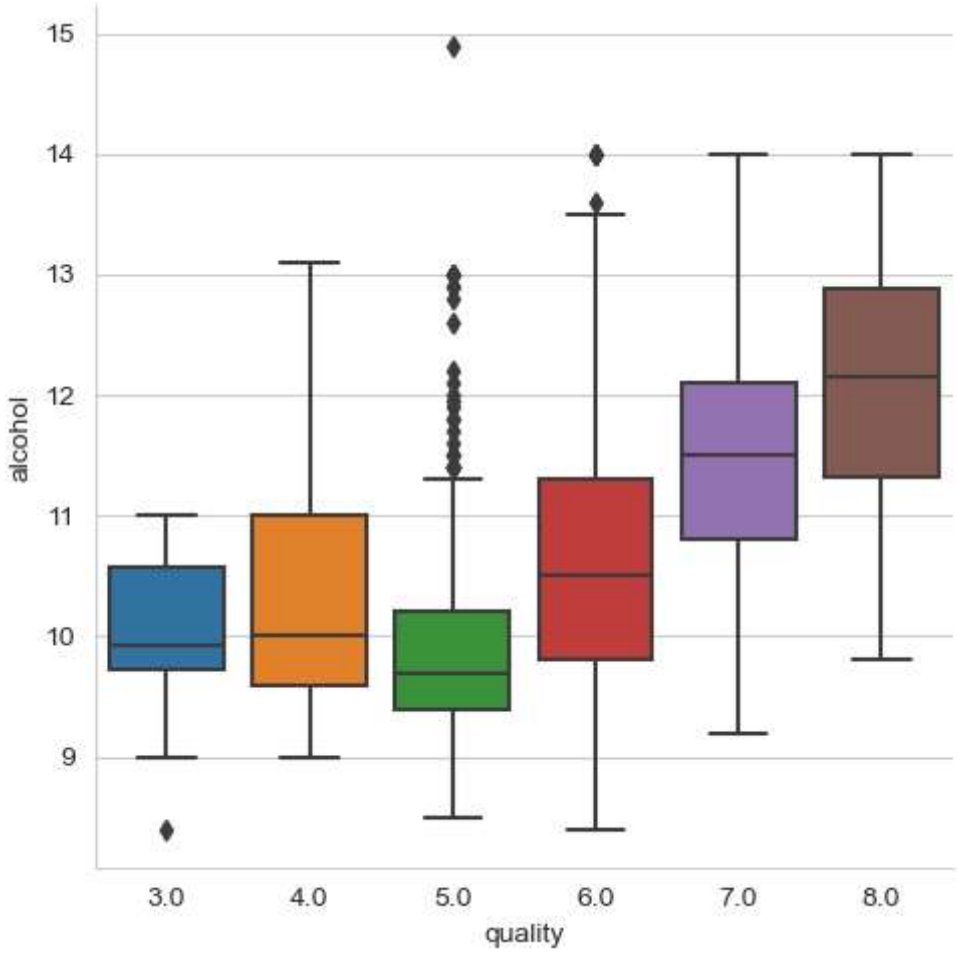


In [22]:
```
sns.catplot(x="quality", y="sulphates", data=dataset, kind="box")
```

Out[22]:
```
<seaborn.axisgrid.FacetGrid at 0x2ee9b46dbe0>
```



In [23]:
```
sns.catplot(x="quality", y="alcohol", data=dataset, kind="box");
```



Acidity Type with Different Qualities of Wine

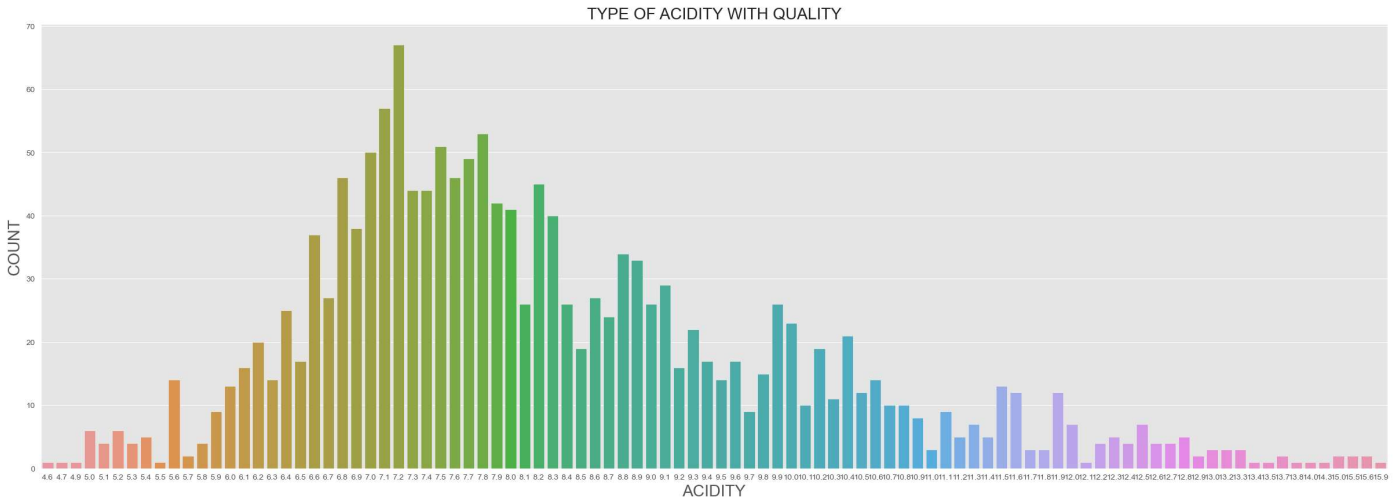
In [24]:
```
acidity_count = dataset["fixed acidity"].value_counts().reset_index()
acidity_count
```

Out[24]:

	index	fixed acidity
0	7.2	67
1	7.1	57
2	7.8	53
3	7.5	51
4	7.0	50
...	...	...
91	13.5	1
92	13.8	1
93	13.4	1
94	4.7	1
95	5.5	1

96 rows × 2 columns

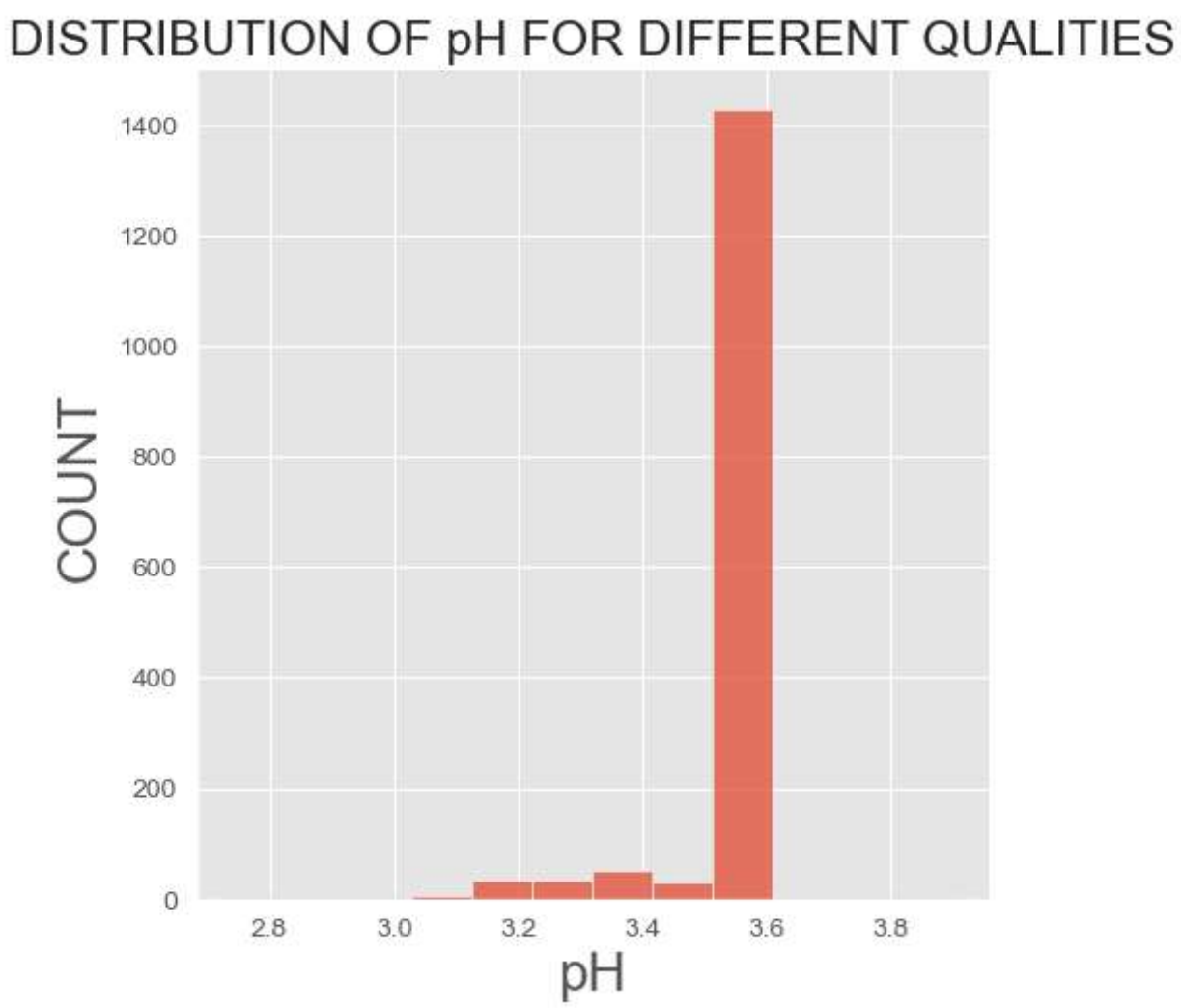
In [25]:
```
plt.figure(figsize=(30, 10))
plt.style.use("ggplot")
sns.barplot(x=acidity_count["index"], y=acidity_count["fixed acidity"])
plt.title("TYPE OF ACIDITY WITH QUALITY", fontsize=20)
plt.xlabel("ACIDITY", fontsize=20)
plt.ylabel("COUNT", fontsize=20)
plt.show()
```



Distribution of pH with Different Qualities of Wine

In [26]:
```
plt.style.use("ggplot")
sns.displot(dataset["pH"]);  # using displot here
plt.title("DISTRIBUTION OF pH FOR DIFFERENT QUALITIES", fontsize=18)
plt.xlabel("pH", fontsize=20)
```

```
plt.ylabel("COUNT", fontsize=20)
plt.show()
```



Skewness Correction

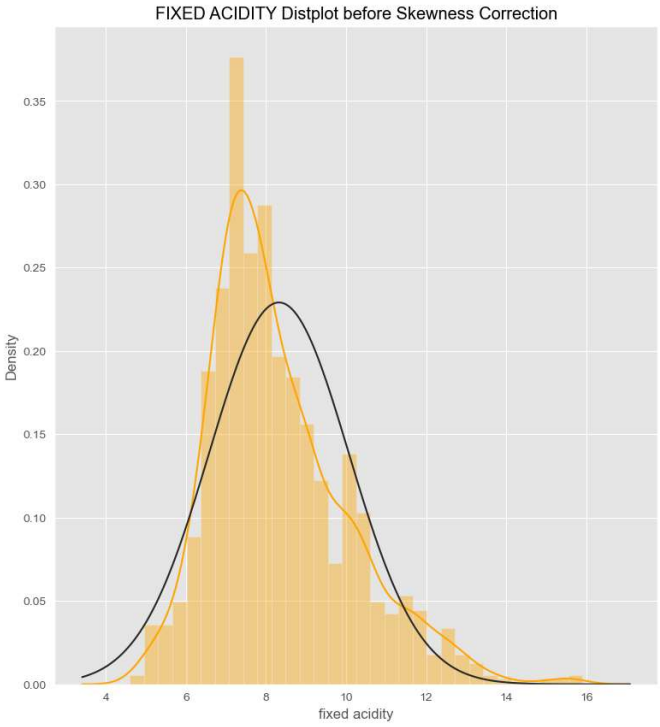
Here we will try to correct Skewness in some independent varaibles of our dataset

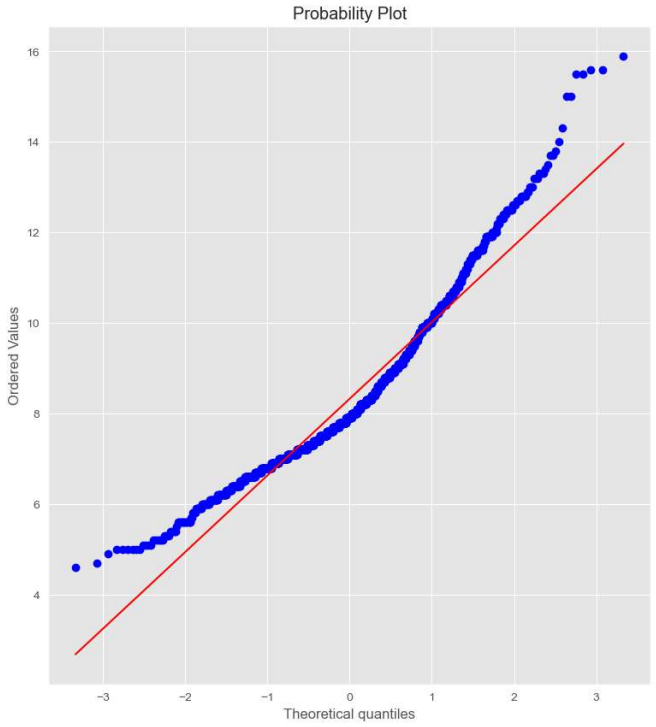
In [27]:
```
def skewnessCorrector(columnName):
    print('''Before Correcting''')
    (mu, sigma) = norm.fit(dataset[columnName])
    print("Mu before correcting {} : {}, Sigma before correcting {} : {}".format(
        columnName.upper(), mu, columnName.upper(), sigma))
    plt.figure(figsize=(20,10))
    plt.subplot(1,2,1)
    sns.distplot(dataset[columnName], fit=norm, color="orange")
    plt.title(columnName.upper() +
              " Distplot before Skewness Correction", color="black")
    plt.subplot(1,2,2)
    stats.probplot(dataset[columnName], plot=plt)
    plt.show();
    dataset[columnName], lam_fixed_acidity = boxcox(
        dataset[columnName])
    print('''After Correcting''')
    print("Mu after correcting {} : {}, Sigma after correcting {} : {}".format(
        columnName.upper(), mu, columnName.upper(), sigma))
    plt.figure(figsize=(20, 10))
    plt.subplot(1,2,1)
    sns.distplot(dataset[columnName], fit=norm, color="orange")
    plt.title(columnName.upper() +
```

```
            " Distplot After Skewness Correction", color="black")
    plt.subplot(1,2,2)
    stats.probplot(dataset[columnName], plot = plt)
    plt.show();
```

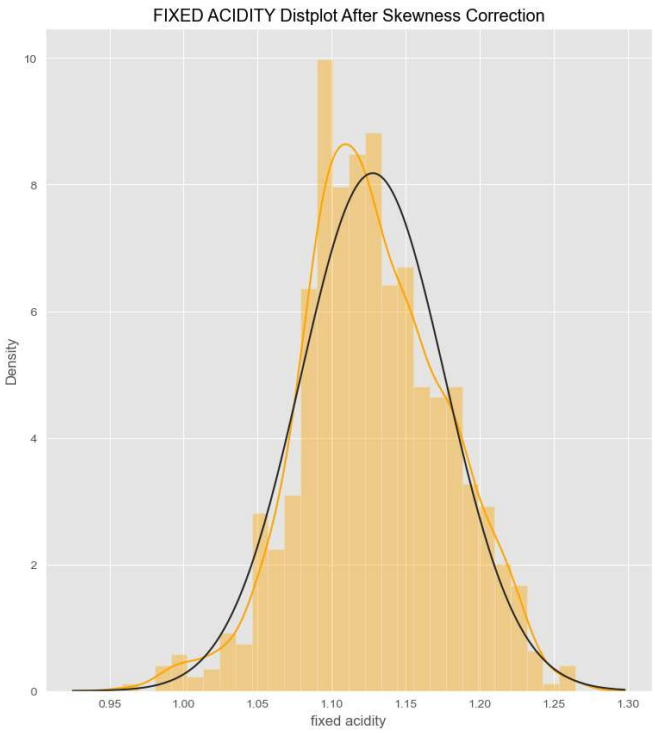
In [28]:
```
skewColumnList = [
    'fixed acidity', 'residual sugar', 'free sulfur dioxide', 'total sulfur dioxide', 'sulphates
]
for columns in skewColumnList:
    skewnessCorrector(columns)
```

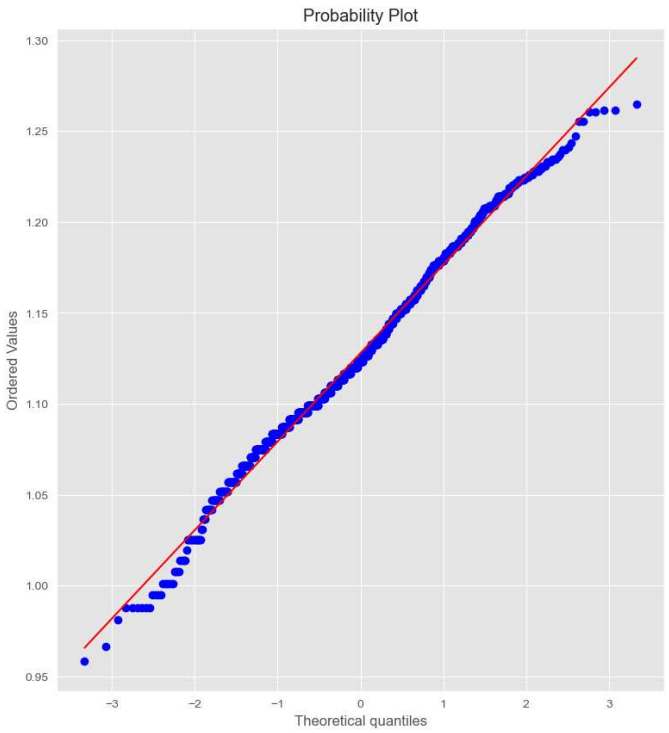
```
Before Correcting
Mu before correcting FIXED ACIDITY : 8.321365914786968, Sigma before correcting FIXED ACIDITY :
1.741575033679555
```



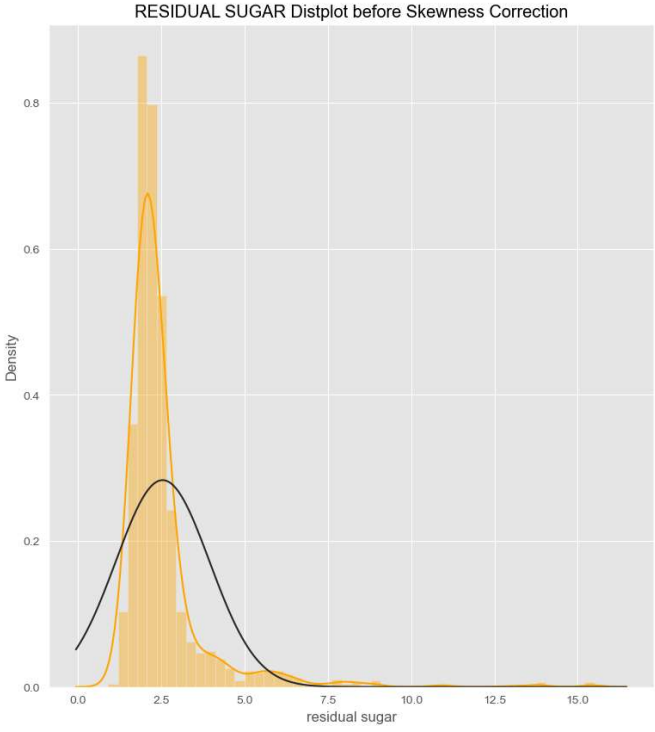


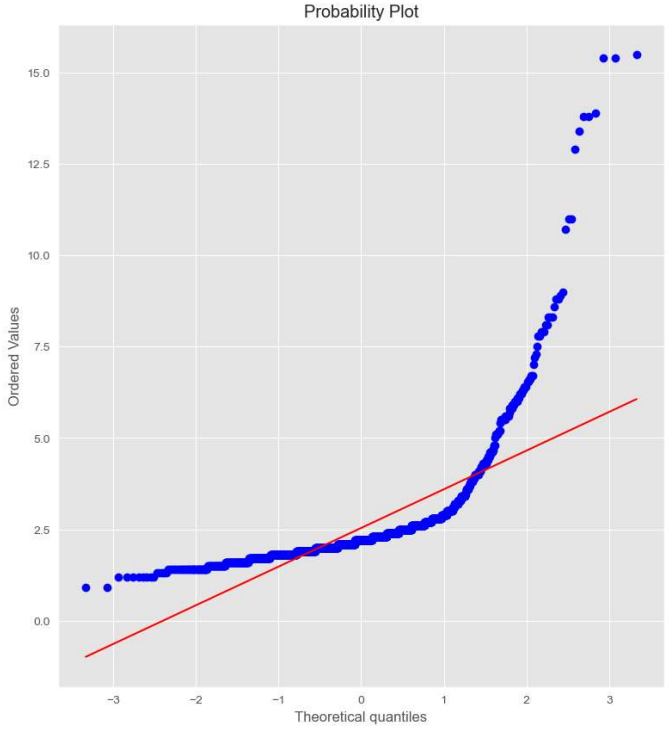

```
C:\Users\rohit\AppData\Local\Temp\ipykernel_5500\442355379.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy
  dataset[columnName], lam_fixed_acidity = boxcox(
After Correcting
Mu after correcting FIXED ACIDITY : 8.321365914786968, Sigma after correcting FIXED ACIDITY : 1.
741575033679555
```



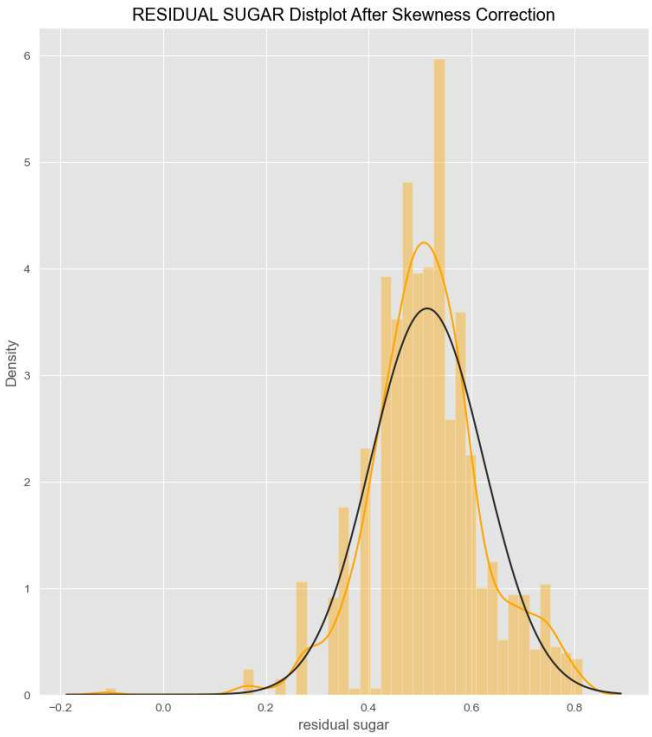


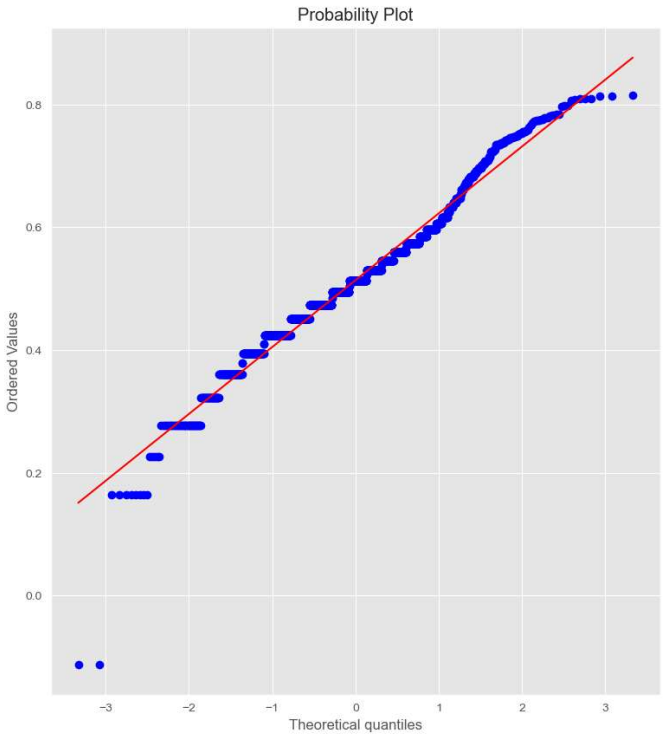

```
Before Correcting
Mu before correcting RESIDUAL SUGAR : 2.536936090225564, Sigma before correcting RESIDUAL SUGAR
: 1.407899330469901
```



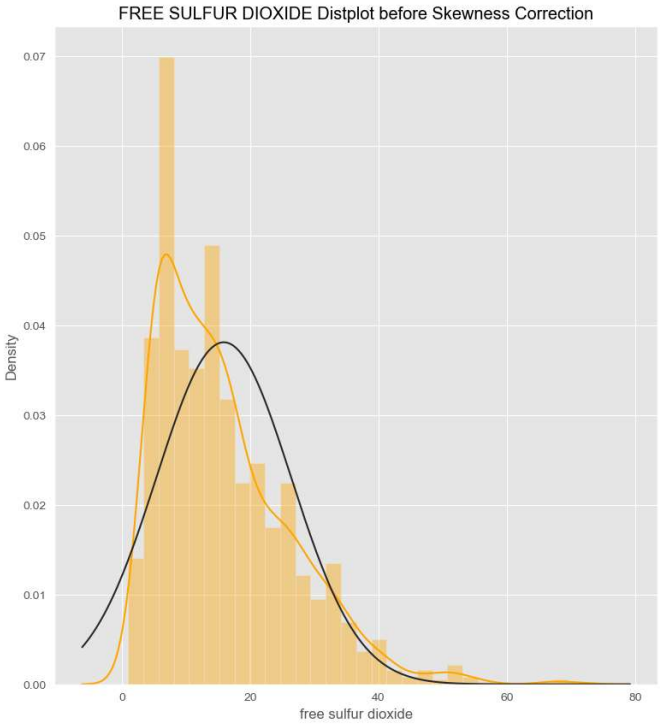


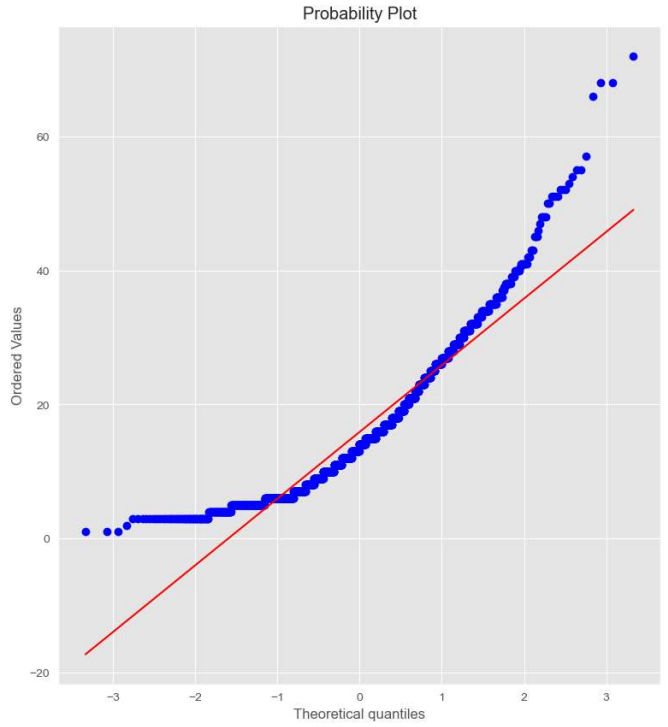

```
C:\Users\rohit\AppData\Local\Temp\ipykernel_5500\442355379.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy
  dataset[columnName], lam_fixed_acidity = boxcox(
After Correcting
Mu after correcting RESIDUAL SUGAR : 2.536936090225564, Sigma after correcting RESIDUAL SUGAR :
1.407899330469901
```



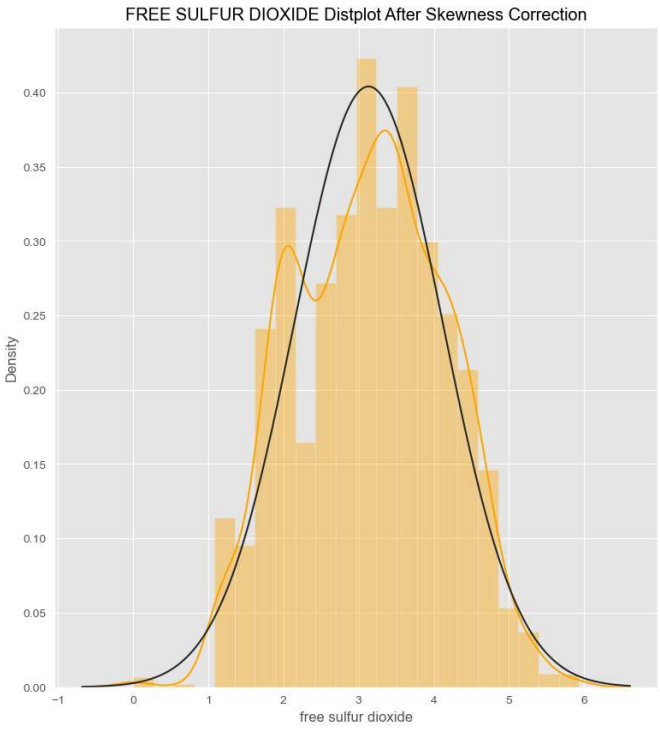


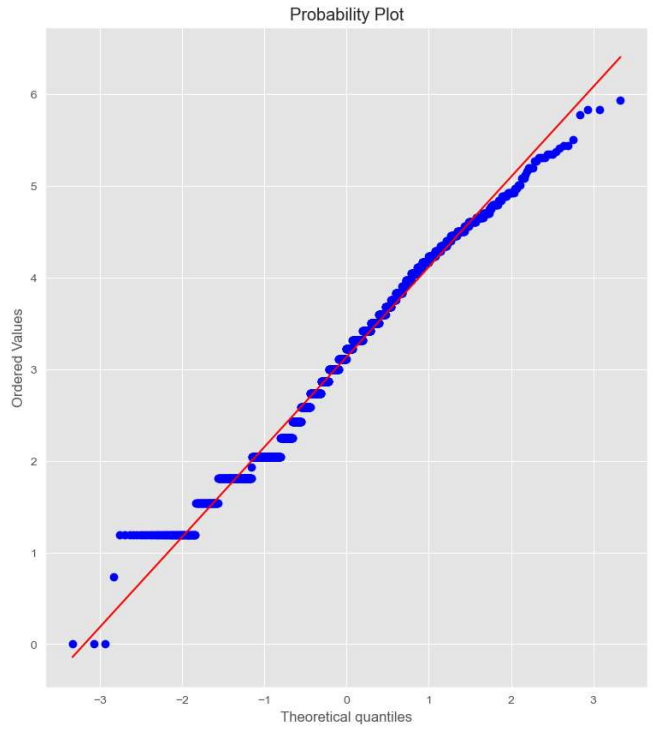

```
Before Correcting
Mu before correcting FREE SULFUR DIOXIDE : 15.882205513784461, Sigma before correcting FREE SULF
UR DIOXIDE : 10.464100661523947
```



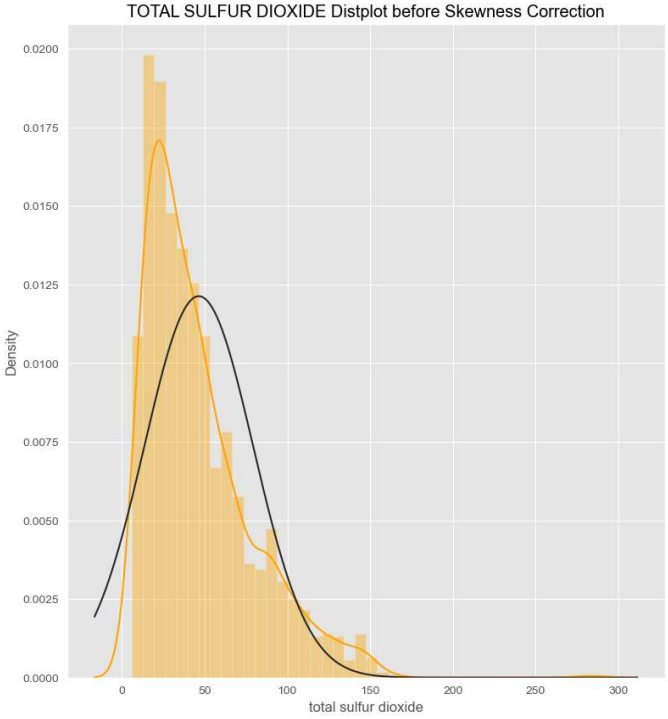


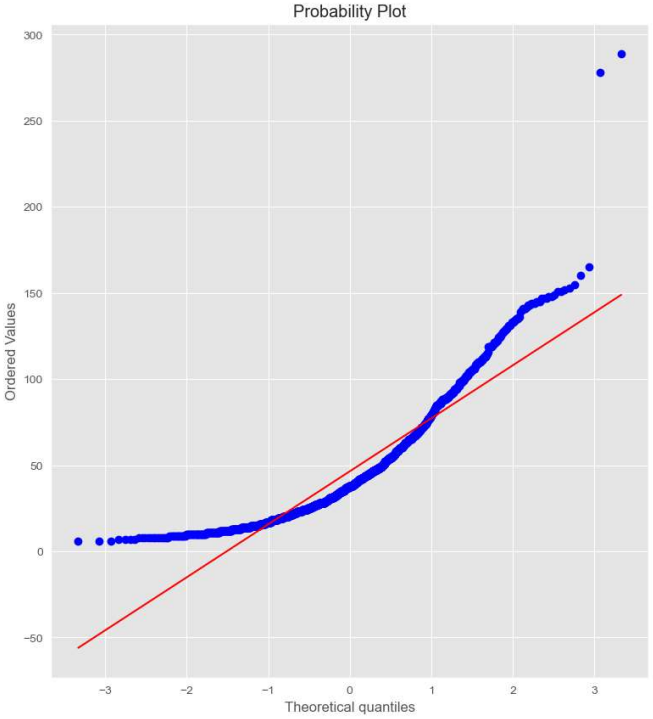

```
C:\Users\rohit\AppData\Local\Temp\ipykernel_5500\442355379.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy
  dataset[columnName], lam_fixed_acidity = boxcox(
After Correcting
Mu after correcting FREE SULFUR DIOXIDE : 15.882205513784461, Sigma after correcting FREE SULFUR
DIOXIDE : 10.464100661523947
```



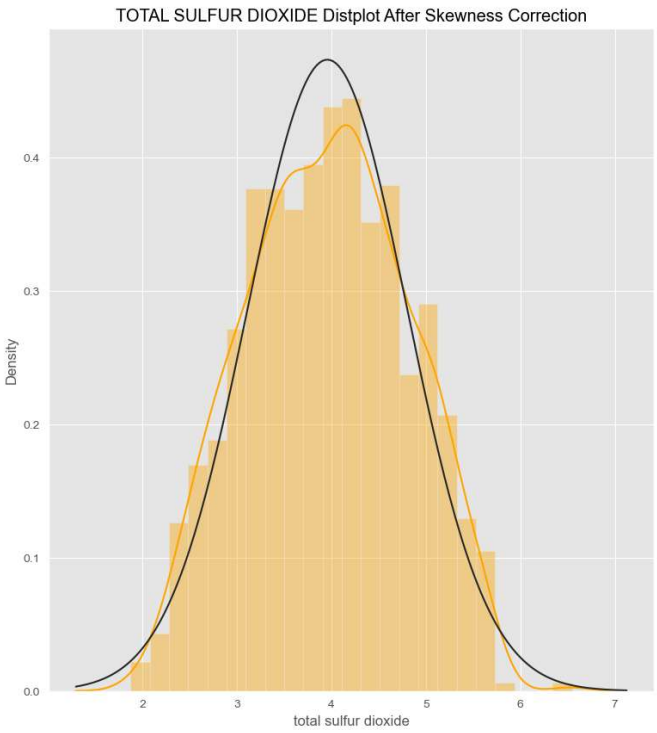


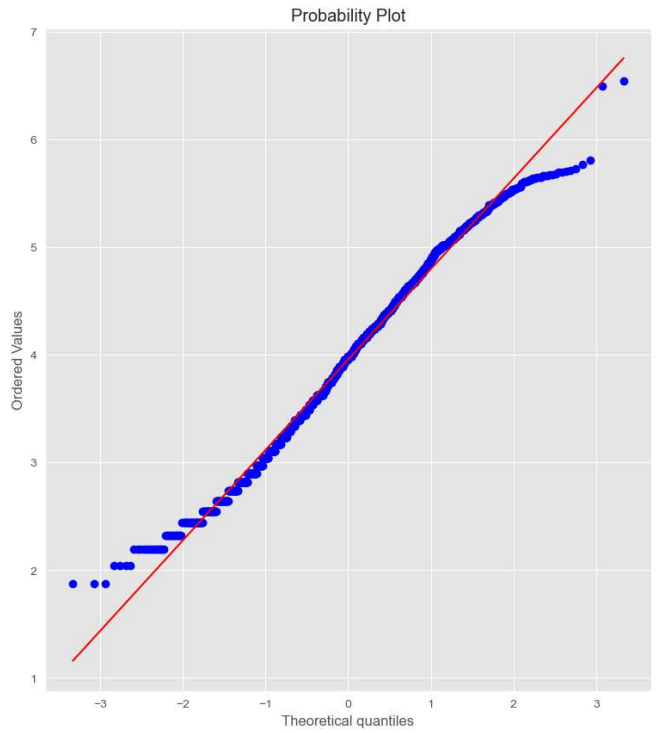

```
Before Correcting
Mu before correcting TOTAL SULFUR DIOXIDE : 46.43107769423559, Sigma before correcting TOTAL SUL
FUR DIOXIDE : 32.88276546693362
```



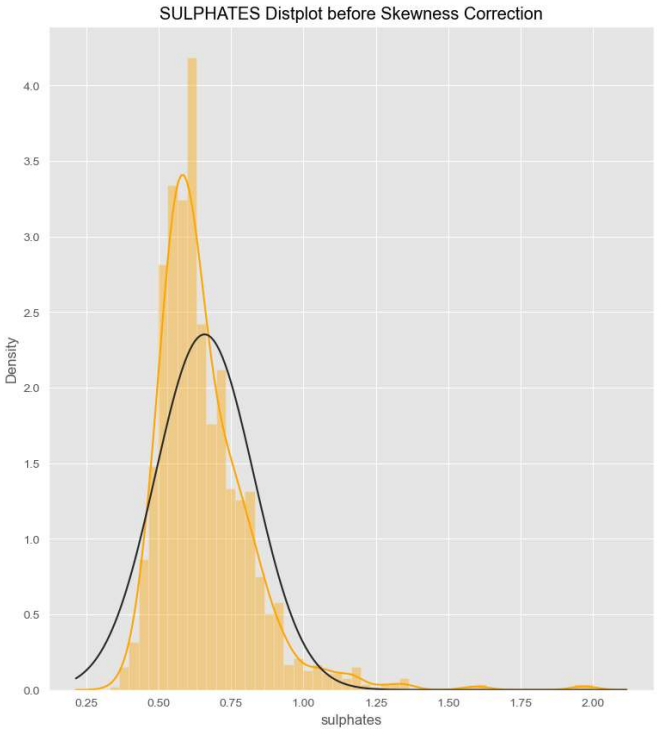


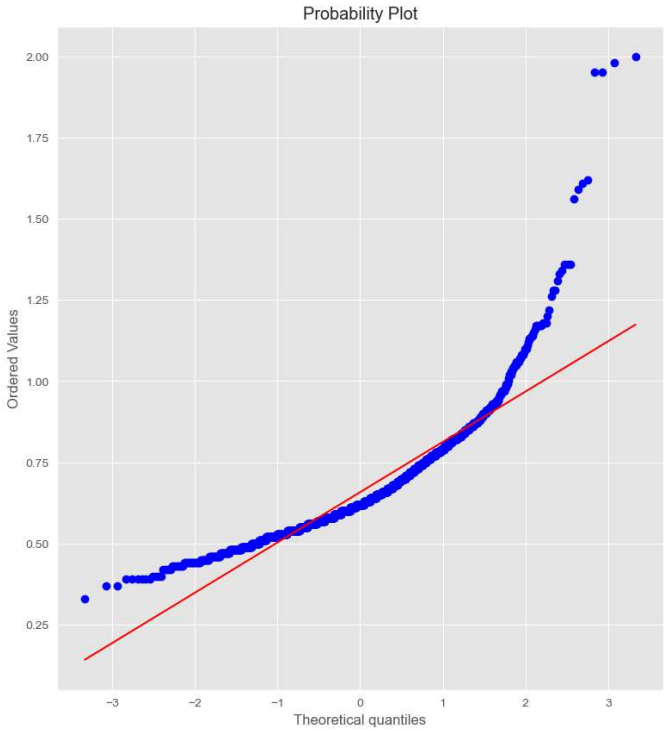

```
C:\Users\rohit\AppData\Local\Temp\ipykernel_5500\442355379.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy
  dataset[columnName], lam_fixed_acidity = boxcox(
After Correcting
Mu after correcting TOTAL SULFUR DIOXIDE : 46.43107769423559, Sigma after correcting TOTAL SULFU
R DIOXIDE : 32.88276546693362
```



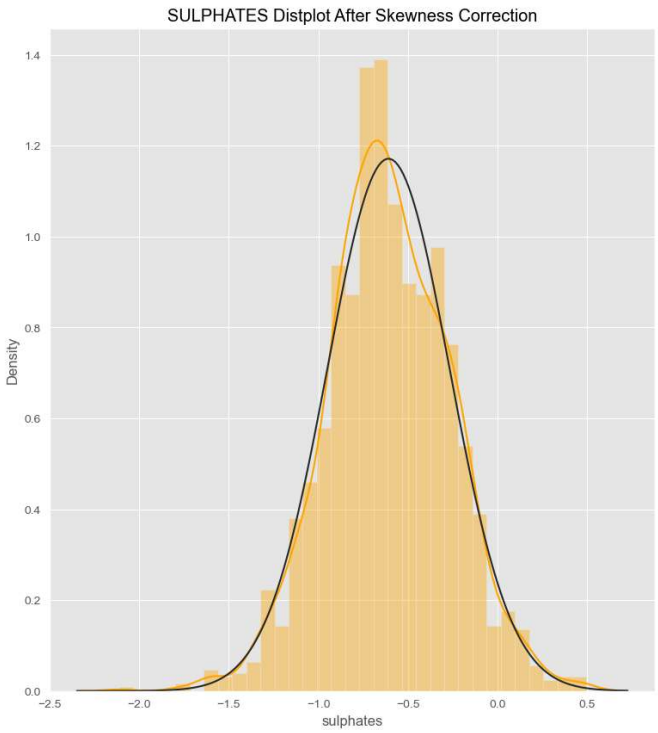


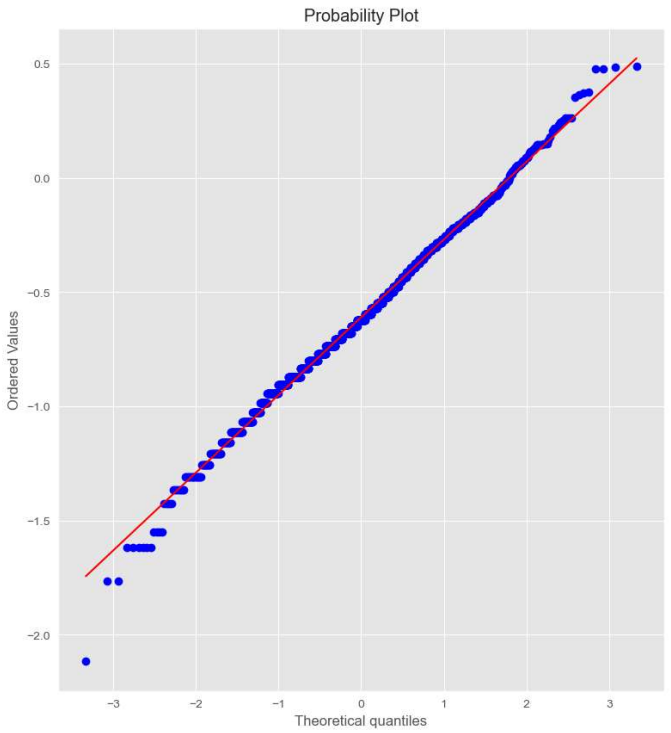

```
Before Correcting
Mu before correcting SULPHATES : 0.6581892230576439, Sigma before correcting SULPHATES : 0.16953
43520513218
```






```
C:\Users\rohit\AppData\Local\Temp\ipykernel_5500\442355379.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy
  dataset[columnName], lam_fixed_acidity = boxcox(
After Correcting
Mu after correcting SULPHATES : 0.6581892230576439, Sigma after correcting SULPHATES : 0.1695343
520513218
```





Outlier Correction

We have detected several outliers in our dataset here we will try to correct them.

In [29]:
```
def detect_outliers(columns):
    outlier_indices = []

    for column in columns:
        # 1st quartile
        Q1 = np.percentile(dataset[column], 25)
        # 3st quartile
        Q3 = np.percentile(dataset[column], 75)
        # IQR
        IQR = Q3 - Q1
        # Outlier Step
        outlier_step = IQR * 1.5
        # detect outlier and their indeces
        outlier_list_col = dataset[(dataset[column] < Q1 - outlier_step)
                                   | (dataset[column] > Q3 + outlier_step)].index
        # store indeces
        outlier_indices.extend(outlier_list_col)

    outlier_indices = Counter(outlier_indices)
    multiple_outliers = list(i for i, v in outlier_indices.items() if v > 1.5)

    return multiple_outliers
```

In [30]:
```
print("number of outliers detected --> ",
      len(dataset.loc[detect_outliers(dataset.columns[:-1])]))
```

```
number of outliers detected -->  72
```

In [31]:
```
dataset.loc[detect_outliers(dataset.columns[:-1])]
```

Out[31]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
45	0.958601	0.520	0.15	0.494470	0.054	2.428544	4.638743	0.9934	3.90	-0.801214	13.1	4.0
94	0.987994	1.020	0.04	0.277736	0.045	4.925770	4.971034	0.9938	3.75	-1.111103	10.5	4.0
95	0.966339	0.600	0.17	0.530431	0.058	3.506925	5.247860	0.9932	3.85	-0.678115	12.9	6.0
442	1.261597	0.685	0.76	0.666711	0.100	2.047186	4.135524	1.0032	3.52	-0.476557	11.2	7.0
553	0.987994	1.040	0.24	0.360819	0.050	4.507782	5.123243	0.9934	3.52	-0.622697	11.5	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...
65	1.098997	0.725	0.05	0.709533	0.086	1.536103	2.546557	0.9962	3.41	-1.617327	10.9	5.0
86	1.144076	0.490	0.28	0.450185	0.110	3.755351	5.563998	0.9972	2.93	0.478421	9.9	6.0
91	1.144076	0.490	0.28	0.450185	0.110	3.755351	5.563998	0.9972	2.93	0.478421	9.9	6.0
92	1.144076	0.490	0.29	0.473533	0.110	3.676309	5.535543	0.9972	2.93	0.485875	9.8	5.0
170	1.123200	0.885	0.03	0.423995	0.058	1.536103	2.190640	0.9972	3.36	-2.113843	9.1	4.0

72 rows × 12 columns

Dropping Outliers

In [32]:
```
dataset = dataset.drop(detect_outliers(dataset.columns[:-1]),axis = 0).reset_index(drop = True)
```

2) Using Pandas Profiling

In [33]:
```
ProfileReport(dataset)
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

Overview

Dataset statistics

Number of variables	12
Number of observations	1524
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	209
Duplicate rows (%)	13.7%
Total size in memory	143.0 KiB
Average record size in memory	96.1 B

Variable types

Numeric	12

Alerts

Dataset has 209 (13.7%) duplicate rows	**Duplicates**
`fixed acidity` is highly overall correlated with `citric acid` and 1 other fields (citric acid, density)	**High correlation**
`volatile acidity` is highly overall correlated with `citric acid`	**High correlation**
`citric acid` is highly overall correlated with `fixed acidity` and 1	**High correlation**

Out[33]:

Correlation and Causation

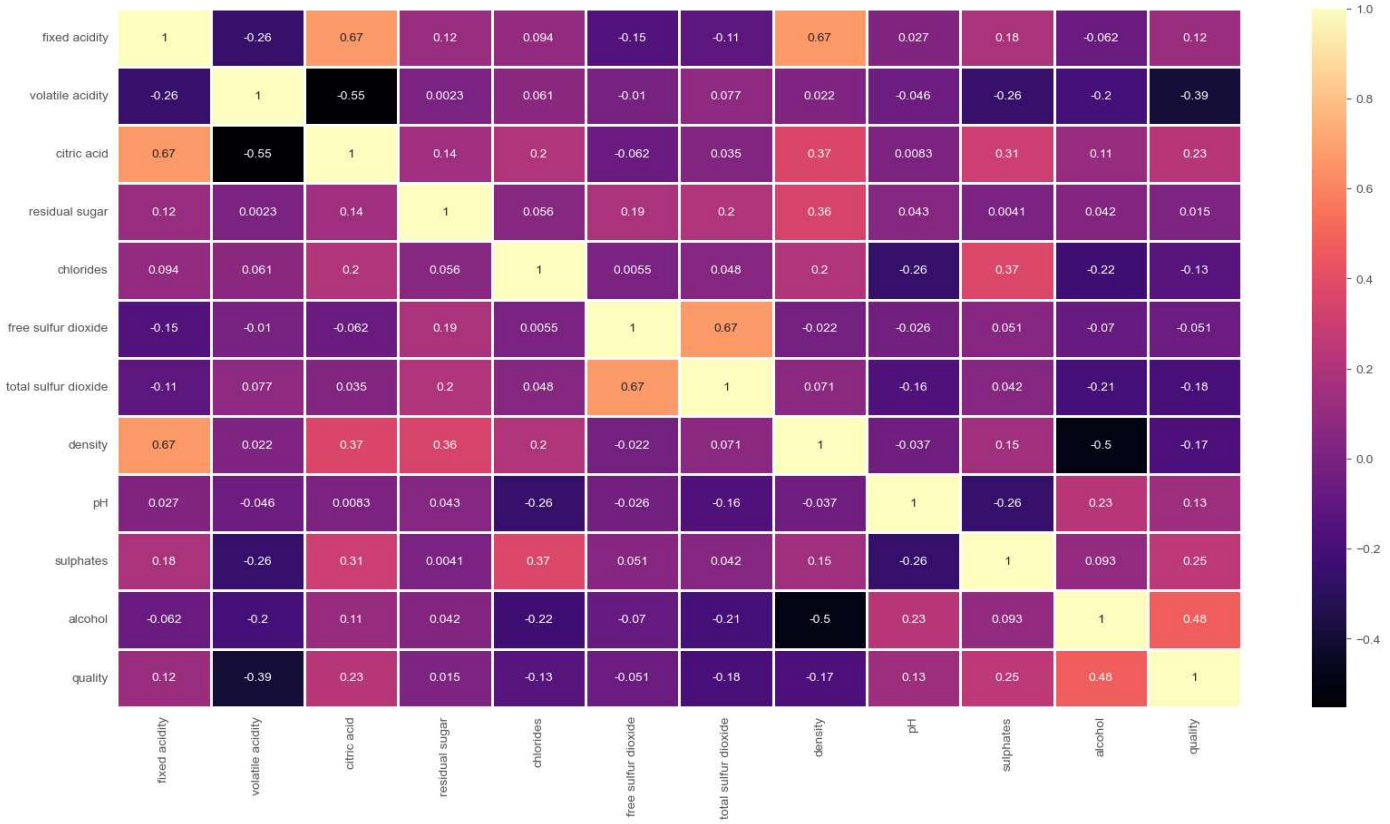
Two or more variables considered to be related, in a statistical context, if their values change so that as the value of one variable increases or decreases so does the value of the other variable (although it may be in the opposite direction). For example, for the two variables "hours worked" and "income earned" there is a relationship between the two if the increase in hours worked is associated with an increase in income earned. If we consider the two variables "price" and "purchasing power", as the price of goods increases a person's ability to buy these goods decreases (assuming a constant income).

Correlation is a statistical measure (expressed as a number) that describes the size and direction of a relationship between two or more variables. A correlation between variables, however, does not automatically mean that the change in one variable is the cause of the change in the values of the other variable.

Causation indicates that one event is the result of the occurrence of the other event; i.e. there is a causal relationship between the two events. This is also referred to as cause and effect.

Theoretically, the difference between the two types of relationships are easy to identify — an action or occurrence can cause another (e.g. smoking causes an increase in the risk of developing lung cancer), or it can correlate with another (e.g. smoking is correlated with alcoholism, but it does not cause alcoholism). In practice, however, it remains difficult to clearly establish cause and effect, compared with establishing correlation.

In [34]:
```
plt.figure(figsize = [20, 10], facecolor = 'white')
sns.heatmap(df.corr(), annot = True, linewidths = 2, cmap = "magma");
```



IMPORTANT NOTE! There is 'multicollinearity' problem

Here we see that there is relatively high (0.67, positive) correlation between 'free sulfur dioxide' and 'total_sulfur_dioxide' variables. There is relatively high (-0.68, negative) correlation between "pH" and "fixed_acidity" variables. And there is about 0.5 correlation between some of other variables. That's why we must consider when build Machine Learning models

* Look at Dataset

In [35]:
```
dataset.head().style.background_gradient(cmap = "Purples_r")
```

Out[35]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	1.106301	0.700000	0.000000	0.450185	0.076000	2.869746	3.854084	0.997800	3.510000	-0.801214	9.400000
1	1.119965	0.880000	0.000000	0.573178	0.098000	4.106167	4.676060	0.996800	3.200000	-0.476557	9.800000
2	1.119965	0.760000	0.040000	0.530431	0.092000	3.319568	4.411663	0.997000	3.260000	-0.546151	9.800000
3	1.201974	0.280000	0.560000	0.450185	0.075000	3.506925	4.540451	0.998000	3.160000	-0.737477	9.800000
4	1.106301	0.700000	0.000000	0.450185	0.076000	2.869746	3.854084	0.997800	3.510000	-0.801214	9.400000

Divide quality range into 2 parts

In this dataset quality range is between 3 and 8 We will divide quality range into two parts:

- High quality wine: 6 - 8
- Low quality wine: 3 - 5

In [36]:
```
df["quality"] = np.where(df["quality"] > 5, 1, 0)
df["quality"]
```

Out[36]:
```
0       0
1       0
2       0
3       1
4       0
      ..
1594    0
1595    1
1596    1
1597    0
1598    1
Name: quality, Length: 1596, dtype: int32
```

Look at Dataset (with changed 'quality' variable)

In [37]:
```
df.head().style.background_gradient(cmap = "Reds")
```

Out[37]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohc
0	7.400000	0.700000	0.000000	1.900000	0.076000	11.000000	34.000000	0.997800	3.510000	0.560000	9.40000
1	7.800000	0.880000	0.000000	2.600000	0.098000	25.000000	67.000000	0.996800	3.200000	0.680000	9.80000
2	7.800000	0.760000	0.040000	2.300000	0.092000	15.000000	54.000000	0.997000	3.260000	0.650000	9.80000
3	11.200000	0.280000	0.560000	1.900000	0.075000	17.000000	60.000000	0.998000	3.160000	0.580000	9.80000
4	7.400000	0.700000	0.000000	1.900000	0.076000	11.000000	34.000000	0.997800	3.510000	0.560000	9.40000

Data Preprocessing

- If quality value is less than or eqaul to 6 then it will be in class 0
- If quality value is greater than 6 then it will be in class 1

In [38]:
```
dataset['quality'] = np.where(dataset['quality'] > 6, 1, 0)
dataset['quality'].value_counts()
```

Out[38]:
```
0    1319
1     205
Name: quality, dtype: int64
```

In [39]:
```
X = dataset.iloc[:, 0:-1].values
y = dataset.iloc[:, -1].values
```

In [40]:
```
X
```

Out[40]:
```
array([[ 1.10630075,  0.7       ,  0.        , ...,  3.51      ,
         -0.80121374,  9.4       ],
       [ 1.11996501,  0.88      ,  0.        , ...,  3.2       ,
         -0.476557  ,  9.8       ],
       [ 1.11996501,  0.76      ,  0.04      , ...,  3.26      ,
         -0.54615126,  9.8       ],
       ...,
       [ 1.06143542,  0.51      ,  0.13      , ...,  3.52      ,
         -0.33643239, 11.        ],
       [ 1.04172459,  0.645     ,  0.12      , ...,  3.52      ,
         -0.41301622, 10.2       ],
       [ 1.04685707,  0.31      ,  0.47      , ...,  3.52      ,
         -0.52222854, 11.        ]])
```

In [41]:
```
y
```

Out[41]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [42]:
```
X.shape
```

Out[42]:
```
(1524, 11)
```

In [43]:
```
y.shape
```

Out[43]:
```
(1524,)
```

Splitting Dataset into Training Set and Test Set

In [44]:
```
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, random_state=1)
```

Standardizing Independent Variables

In [45]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Training Classifiers on Training Set and drawing Inference

In [46]:
```
accuracy_scores = {}
def predictor(predictor, params):
    global accuracy_scores
    if predictor == 'lr':
        print('Training Logistic Regression on Training Set')
        from sklearn.linear_model import LogisticRegression
        classifier = LogisticRegression(**params)

    elif predictor == 'svm':
        print('Training Support Vector Machine on Training Set')
        from sklearn.svm import SVC
        classifier = SVC(**params)

    elif predictor == 'knn':
        print('Training K-Nearest Neighbours on Training Set')
        from sklearn.neighbors import KNeighborsClassifier
        classifier = KNeighborsClassifier(**params)

    elif predictor == 'dt':
        print('Training Decision Tree Classifier on Training Set')
        from sklearn.tree import DecisionTreeClassifier
        classifier = DecisionTreeClassifier(**params)

    elif predictor == 'nb':
        print('Training Naive Bayes Classifier on Training Set')
        from sklearn.naive_bayes import GaussianNB
        classifier = GaussianNB(**params)

    elif predictor == 'rfc':
        print('Training Random Forest Classifier on Training Set')
        from sklearn.ensemble import RandomForestClassifier
        classifier = RandomForestClassifier(**params)

    classifier.fit(X_train, y_train)

    print('''Predicting Single Cell Result''')
    single_predict = classifier.predict(sc.transform([[
        7.4, 0.7, 0.0, 1.9, 0.076, 11.0, 34.0, 0.9978, 3.51, 0.56, 9.4
    ]]))
    if single_predict > 0 :
        print('High Quality Wine')
    else:
        print('Low Quality Wine')
    print('''Prediciting Test Set Result''')
    y_pred = classifier.predict(X_test)

    result = np.concatenate((y_pred.reshape(len(y_pred), 1),
                             y_test.reshape(len(y_test), 1)),1)
    print(result, '\n')
    print('''Making Confusion Matrix''')
    from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
    y_pred = classifier.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)
    print(cm, '\n')
    plot_confusion_matrix(classifier, X_test, y_test, cmap="pink")
    print('True Positives :', cm[0][0])
    print('False Positives :', cm[0][1])
    print('False Negatives :', cm[1][0])
```

```
    print('True Negatives :', cm[0][1], '\n')

    print('''Classification Report''')
    print(classification_report(y_test, y_pred,
          target_names=['0', '1'], zero_division=1))

    print('''Evaluating Model Performance''')
    accuracy = accuracy_score(y_test, y_pred)
    print(accuracy, '\n')

    print('''Applying K-Fold Cross validation''')
    from sklearn.model_selection import cross_val_score
    accuracies = cross_val_score(
        estimator=classifier, X=X_train, y=y_train, cv=10)
    print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
    accuracy_scores[classifier] = accuracies.mean()*100
    print("Standard Deviation: {:.2f} %".format(accuracies.std()*100), '\n')
```

In [47]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
```

Training Logistic Regression on Training Set

In [48]:
```
predictor('lr', {'penalty': 'l1', 'solver': 'liblinear'})
```

```
Training Logistic Regression on Training Set
Predicting Single Cell Result
High Quality Wine
Prediciting Test Set Result
[[0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
```

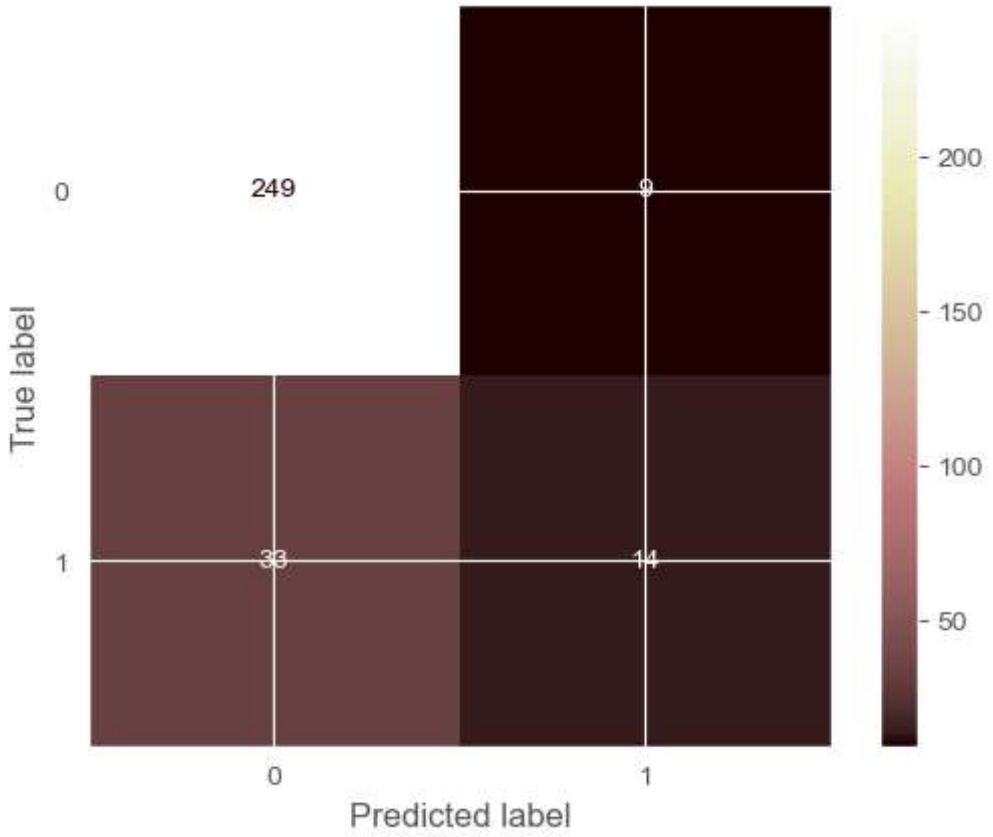
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]

[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[1 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[1 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 1]

[0 0]
[0 0]
[0 0]
[1 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 1]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[1 1]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]

```
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [1 0]
 [0 0]
 [0 1]
 [0 0]
 [0 1]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]]

Making Confusion Matrix
[[249   9]
 [ 33  14]]

True Positives : 249
False Positives : 9
False Negatives : 33
True Negatives : 9

Classification Report
```

```
              precision    recall  f1-score   support

           0       0.88      0.97      0.92       258
           1       0.61      0.30      0.40        47

    accuracy                           0.86       305
   macro avg       0.75      0.63      0.66       305
weighted avg       0.84      0.86      0.84       305

Evaluating Model Performance
0.8622950819672132

Applying K-Fold Cross validation
Accuracy: 88.84 %
Standard Deviation: 1.22 %
```



Training SVM on Training Set

In [49]:
```
predictor('svm', {'C': .5, 'gamma': 0.8,
          'kernel': 'linear', 'random_state': 0})
```

```
Training Support Vector Machine on Training Set
Predicting Single Cell Result
High Quality Wine
Prediciting Test Set Result
[[0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
```

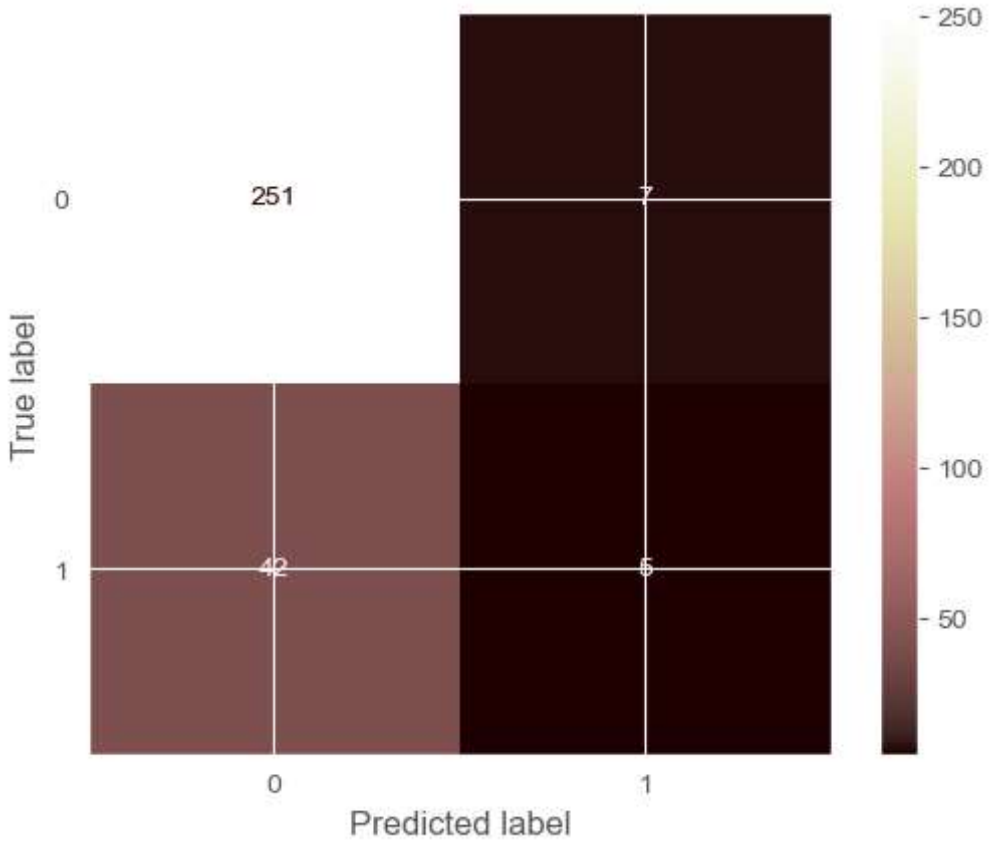
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]

[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 1]

[0 0]
[0 0]
[0 0]
[1 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 1]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[1 1]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]

```
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 1]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]]

Making Confusion Matrix
[[251   7]
 [ 42   5]]

True Positives : 251
False Positives : 7
False Negatives : 42
True Negatives : 7

Classification Report
```

```
              precision    recall  f1-score   support

           0       0.86      0.97      0.91       258
           1       0.42      0.11      0.17        47

    accuracy                           0.84       305
   macro avg       0.64      0.54      0.54       305
weighted avg       0.79      0.84      0.80       305

Evaluating Model Performance
0.839344262295082

Applying K-Fold Cross validation
Accuracy: 88.11 %
Standard Deviation: 0.90 %
```



Training Kernel SVM on Training Set

In [50]:
```
predictor('svm', {'C': .25, 'gamma': 0.1, 'kernel': 'rbf', 'random_state': 0})
```

```
Training Support Vector Machine on Training Set
Predicting Single Cell Result
Low Quality Wine
Prediciting Test Set Result
[[0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
```

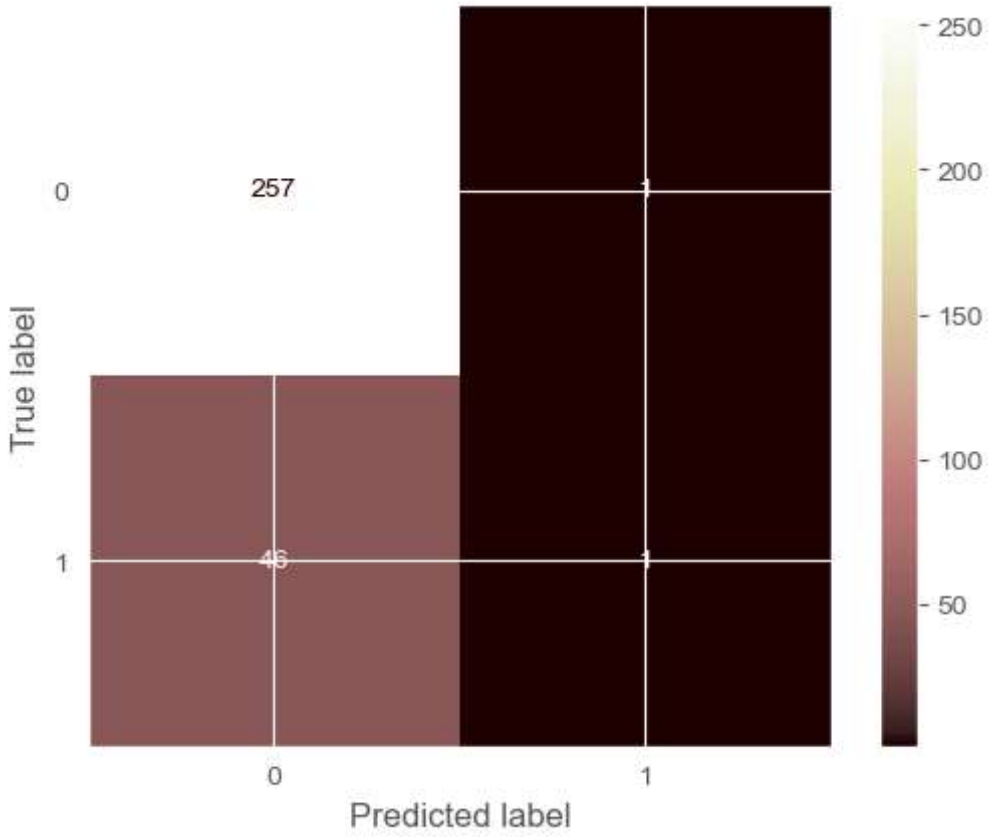
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]

[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 1]

[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 1]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 1]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]

```
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 1]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]]

Making Confusion Matrix
[[257   1]
 [ 46   1]]

True Positives : 257
False Positives : 1
False Negatives : 46
True Negatives : 1

Classification Report
```

```
              precision    recall  f1-score   support

           0       0.85      1.00      0.92       258
           1       0.50      0.02      0.04        47

    accuracy                           0.85       305
   macro avg       0.67      0.51      0.48       305
weighted avg       0.79      0.85      0.78       305

Evaluating Model Performance
0.8459016393442623

Applying K-Fold Cross validation
Accuracy: 88.27 %
Standard Deviation: 1.08 %
```



Training K-Nearest Neighbours on Training Set

In [51]:
```
predictor('knn', {'algorithm': 'auto', 'n_jobs': 1,
          'n_neighbors': 8, 'weights': 'distance'})
```

```
Training K-Nearest Neighbours on Training Set
Predicting Single Cell Result
Low Quality Wine
Prediciting Test Set Result
[[0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 0]
 [0 0]
 [1 0]
 [1 0]
 [1 1]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
```

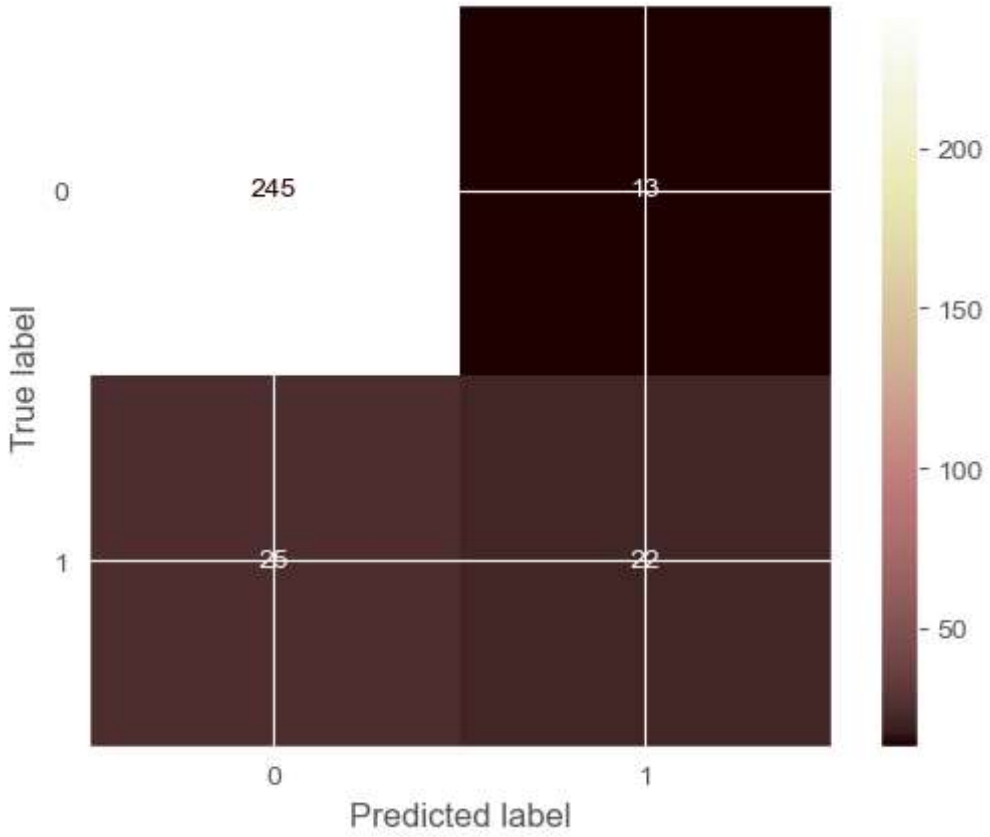
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]

[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[1 1]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 1]

[0 0]
[0 0]
[0 0]
[1 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[1 1]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[1 0]
[0 0]
[1 0]
[0 0]
[1 0]
[0 1]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[1 1]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]

```
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 1]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]]

Making Confusion Matrix
[[245  13]
 [ 25  22]]

True Positives : 245
False Positives : 13
False Negatives : 25
True Negatives : 13

Classification Report
```

```
              precision    recall  f1-score   support

           0       0.91      0.95      0.93       258
           1       0.63      0.47      0.54        47

    accuracy                           0.88       305
   macro avg       0.77      0.71      0.73       305
weighted avg       0.86      0.88      0.87       305

Evaluating Model Performance
0.8754098360655738

Applying K-Fold Cross validation
Accuracy: 90.65 %
Standard Deviation: 1.87 %
```



Training Decision Tree on Training Set

In [52]:
```
predictor('dt', {'criterion': 'entropy', 'max_features': 'auto',
          'splitter': 'best', 'random_state': 0})
```

```
Training Decision Tree Classifier on Training Set
Predicting Single Cell Result
Low Quality Wine
Prediciting Test Set Result
[[0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [1 0]
 [1 1]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
```

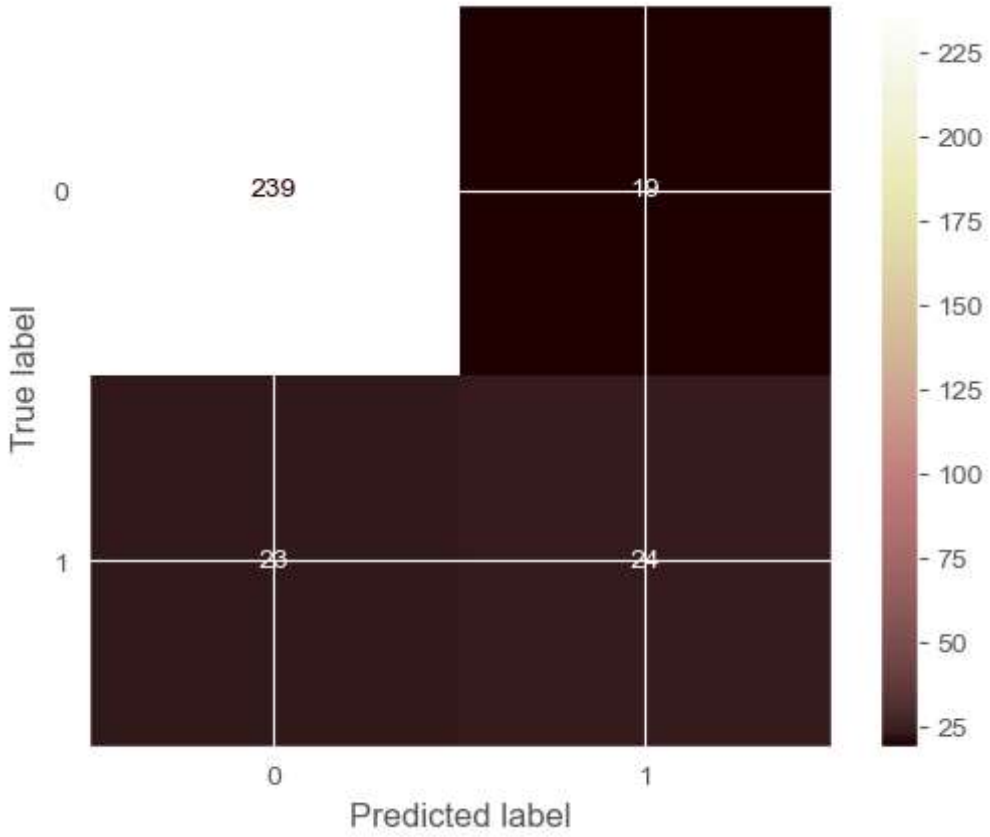
[0 0]
[0 0]
[0 0]
[0 0]
[1 0]
[0 1]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[1 0]

[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[1 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[1 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 1]

[0 0]
[0 0]
[0 0]
[1 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[1 1]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[1 0]
[1 0]
[1 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[1 1]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]

```
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 1]
 [0 0]
 [1 1]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]]

Making Confusion Matrix
[[239  19]
 [ 23  24]]

True Positives : 239
False Positives : 19
False Negatives : 23
True Negatives : 19

Classification Report
```

```
              precision    recall  f1-score   support

           0       0.91      0.93      0.92       258
           1       0.56      0.51      0.53        47

    accuracy                           0.86       305
   macro avg       0.74      0.72      0.73       305
weighted avg       0.86      0.86      0.86       305

Evaluating Model Performance
0.8622950819672132

Applying K-Fold Cross validation
Accuracy: 87.45 %
Standard Deviation: 2.32 %
```



Training Naive Bayes on Training Set

In [53]:
```
predictor('nb', {})
```

```
Training Naive Bayes Classifier on Training Set
Predicting Single Cell Result
High Quality Wine
Prediciting Test Set Result
[[0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 0]
 [0 0]
 [1 0]
 [1 0]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
```

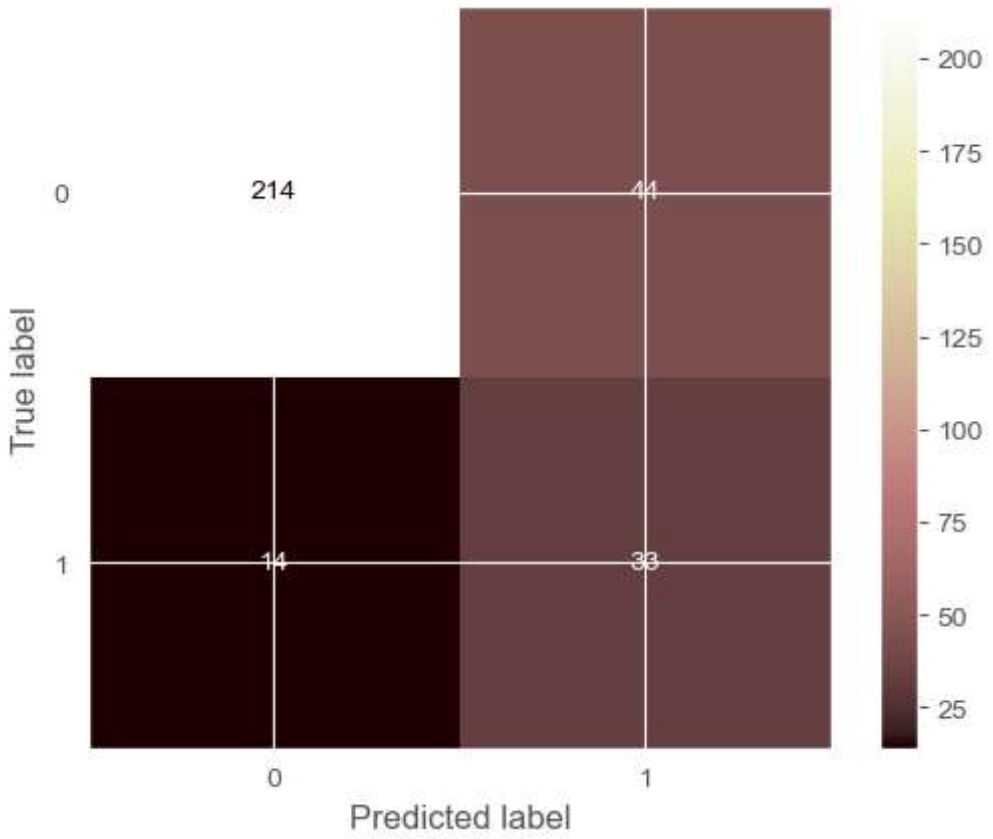
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[1 0]
[0 1]
[0 0]
[1 0]
[0 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[1 0]
[1 1]
[0 0]
[1 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[1 0]
[0 0]
[1 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[1 0]
[0 0]
[0 0]
[1 0]
[0 0]
[0 0]
[1 1]
[0 0]
[1 0]

[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[1 1]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 1]
[0 0]
[0 0]
[0 0]
[0 1]
[1 0]
[0 1]

[0 0]
[0 0]
[0 0]
[1 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 1]
[1 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[1 0]
[0 0]
[0 0]
[1 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[1 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[1 0]
[0 0]
[1 0]
[1 0]
[1 0]
[1 1]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[1 1]
[0 0]
[1 1]
[1 0]
[0 0]
[0 0]
[0 0]
[1 0]
[1 1]
[0 0]
[0 0]
[0 0]
[1 0]
[0 0]

```
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [1 0]
 [0 0]
 [1 1]
 [0 0]
 [1 1]
 [0 0]
 [1 1]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [1 0]]

Making Confusion Matrix
[[214  44]
 [ 14  33]]

True Positives : 214
False Positives : 44
False Negatives : 14
True Negatives : 44

Classification Report
```

```
              precision    recall  f1-score   support

           0       0.94      0.83      0.88       258
           1       0.43      0.70      0.53        47

    accuracy                           0.81       305
   macro avg       0.68      0.77      0.71       305
weighted avg       0.86      0.81      0.83       305

Evaluating Model Performance
0.8098360655737705

Applying K-Fold Cross validation
Accuracy: 82.12 %
Standard Deviation: 2.00 %
```



Training Random Forest Classifier on Training Set

In [54]:
```
predictor('rfc', {'criterion': 'gini', 'max_features': 'log2', 'n_estimators': 100,'random_state
```

```
Training Random Forest Classifier on Training Set
Predicting Single Cell Result
Low Quality Wine
Prediciting Test Set Result
[[0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [1 0]
 [1 1]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
```

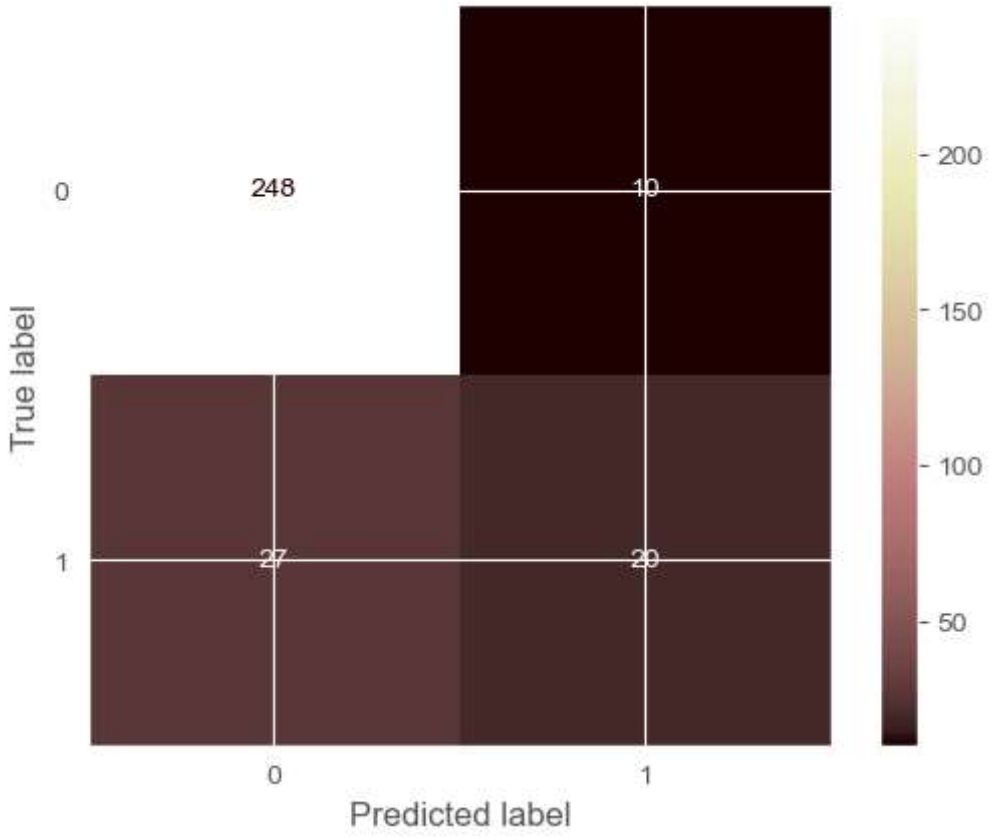
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]

[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 1]

[0 0]
[0 0]
[0 0]
[0 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[1 1]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[1 0]
[0 0]
[0 0]
[0 0]
[1 0]
[0 1]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[1 1]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]

```
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 1]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]]

Making Confusion Matrix
[[248  10]
 [ 27  20]]

True Positives : 248
False Positives : 10
False Negatives : 27
True Negatives : 10

Classification Report
```

```
              precision    recall  f1-score   support

           0       0.90      0.96      0.93       258
           1       0.67      0.43      0.52        47

    accuracy                           0.88       305
   macro avg       0.78      0.69      0.73       305
weighted avg       0.87      0.88      0.87       305

Evaluating Model Performance
0.8786885245901639

Applying K-Fold Cross validation
Accuracy: 91.31 %
Standard Deviation: 1.60 %
```



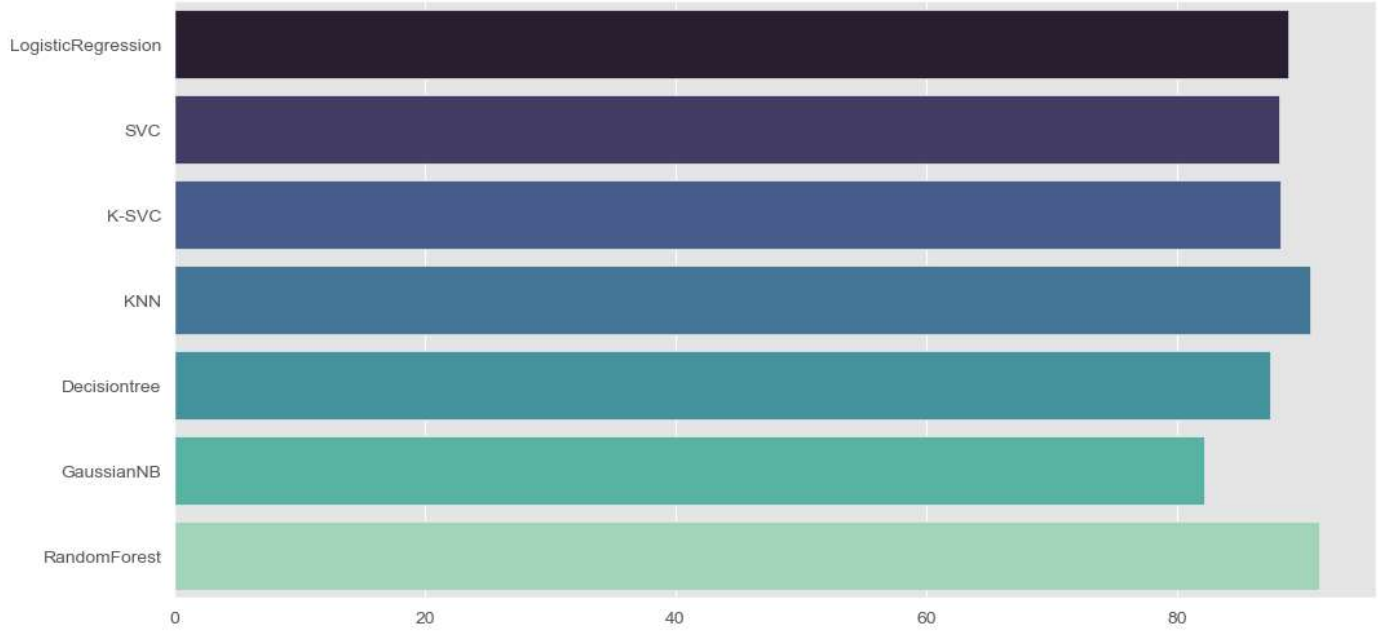
Finding which Classifier performed best

In [55]:
```
maxKey = max(accuracy_scores, key=lambda x: accuracy_scores[x])
print('The model with highest K-Fold Validation Accuracy score is  {0} with an accuracy of  {1:.
    maxKey, accuracy_scores[maxKey]))
```

```
The model with highest K-Fold Validation Accuracy score is  RandomForestClassifier(max_features
='log2', random_state=0) with an accuracy of  91.31
```

Plotting Bar Chart for Accuracies of different classifiers

In [56]:
```
plt.figure(figsize=(12, 6))
model_accuracies = list(accuracy_scores.values())
model_names = ['LogisticRegression', 'SVC',
               'K-SVC', 'KNN', 'Decisiontree', 'GaussianNB', 'RandomForest']
sns.barplot(x=model_accuracies, y=model_names, palette='mako');
```



Summary

- Random Forest Classifier performed best on this data set with an accuracy of 90.81%
- K-Nearest Classifier was just behind with an accuracy of an accuracy of 90.56%

In [59]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the data into a pandas DataFrame
#data = pd.read_csv('WQ data.csv')
#df = data.dropna()


# Split the data into features (X) and target variable (y)
X = df.drop('quality', axis=1)
y = df['quality']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Train the decision tree classifier on the training data
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)

# Make predictions on the test data
y_pred = clf.predict(X_test)

# Calculate the accuracy of the model on the test data
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy: ', accuracy)
```

```
Accuracy:  0.64375
```

In [ ]:

In [ ]:

In [ ]:

thankyou

Submitted by Rohit Varathe

In [ ]: