In [1]:
```
import warnings
warnings.simplefilter("ignore")
import joblib
```

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import scipy.stats as stats
from scipy.stats import zscore
```

In [3]:
```
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier
```

In [4]:
```
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_curve
```

In [5]:
```
df = pd.read_csv("winequality-red.csv")
```

In [6]:
```
df
```

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	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
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

1599 rows × 12 columns

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

Out[7]:
```
(1599, 12)
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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 [9]:
```
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
```

In [10]:
```
df.describe()
```

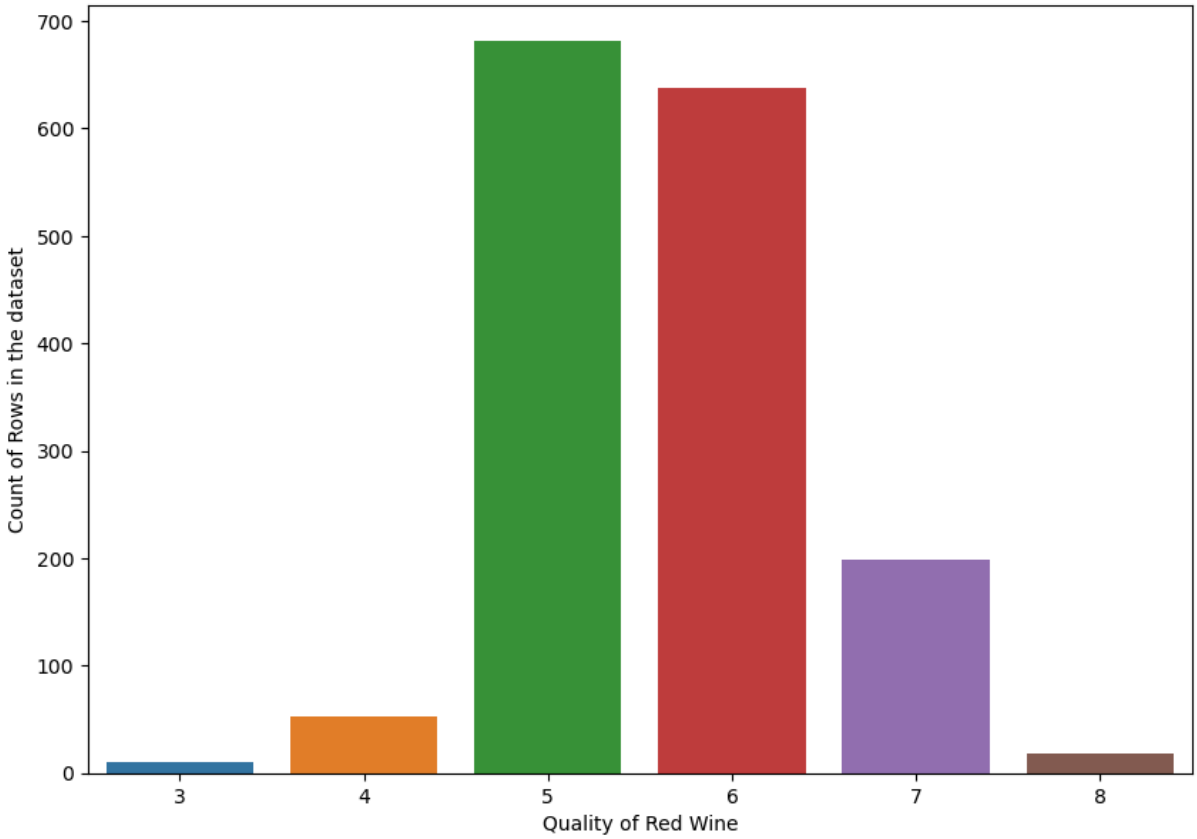
Out[10]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	dens
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.0000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.9967
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.0018
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.9900
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.9956
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.9967
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.9978
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.0036

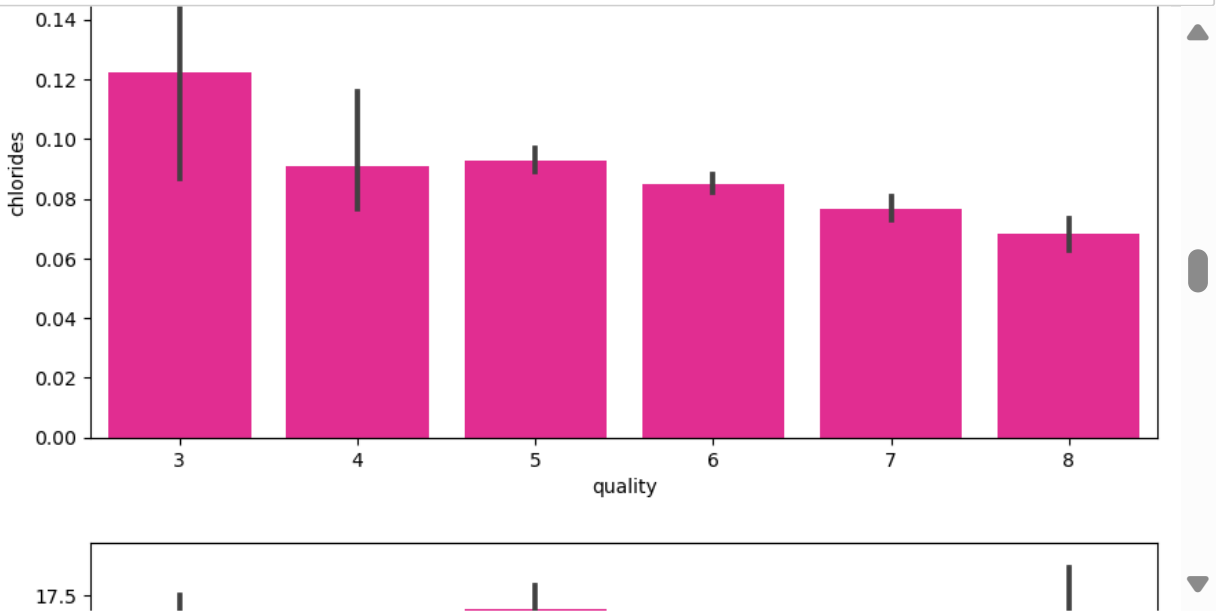
In [11]:
```
df.skew()
```

Out[11]:
```
fixed acidity           0.982751
volatile acidity        0.671593
citric acid             0.318337
residual sugar          4.540655
chlorides               5.680347
free sulfur dioxide     1.250567
total sulfur dioxide    1.515531
density                 0.071288
pH                      0.193683
sulphates               2.428672
alcohol                 0.860829
quality                 0.217802
dtype: float64
```

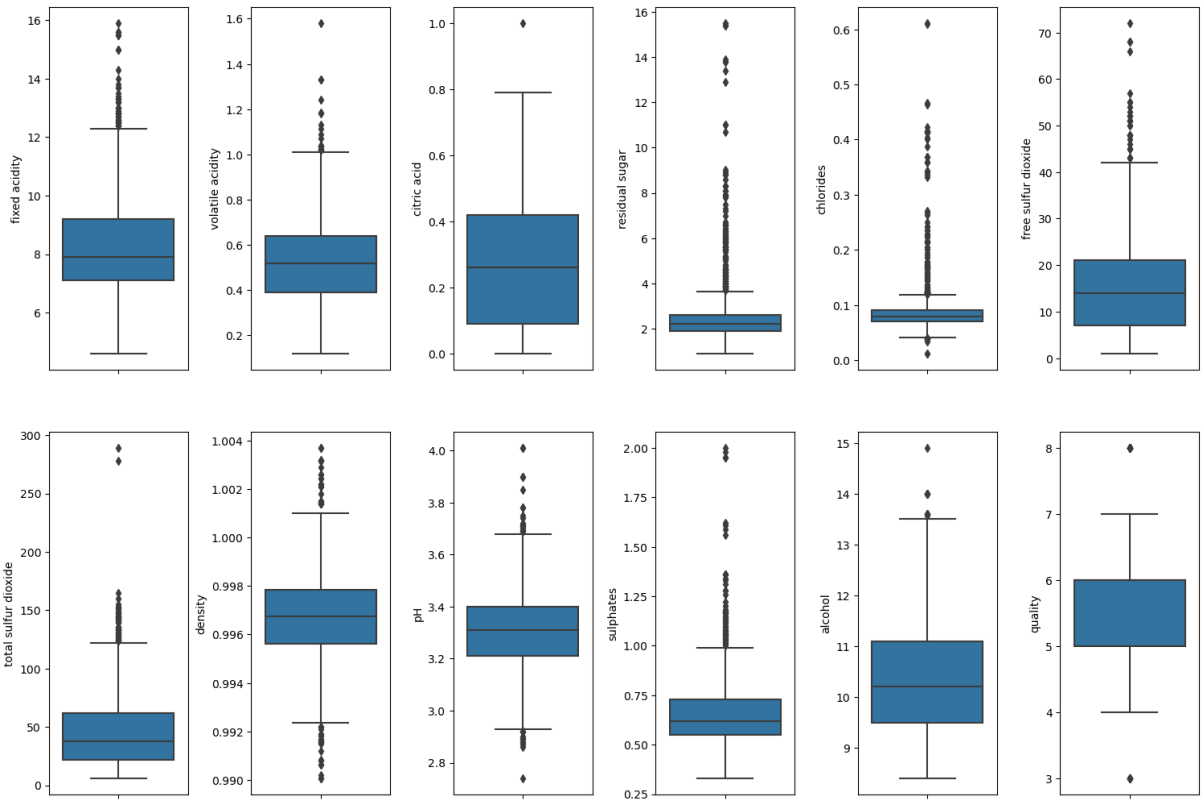
In [12]:
```
plt.figure(figsize=(10,7))
sns.countplot(x ='quality', data = df)
plt.xlabel('Quality of Red Wine')
plt.ylabel('Count of Rows in the dataset')
plt.show()
```



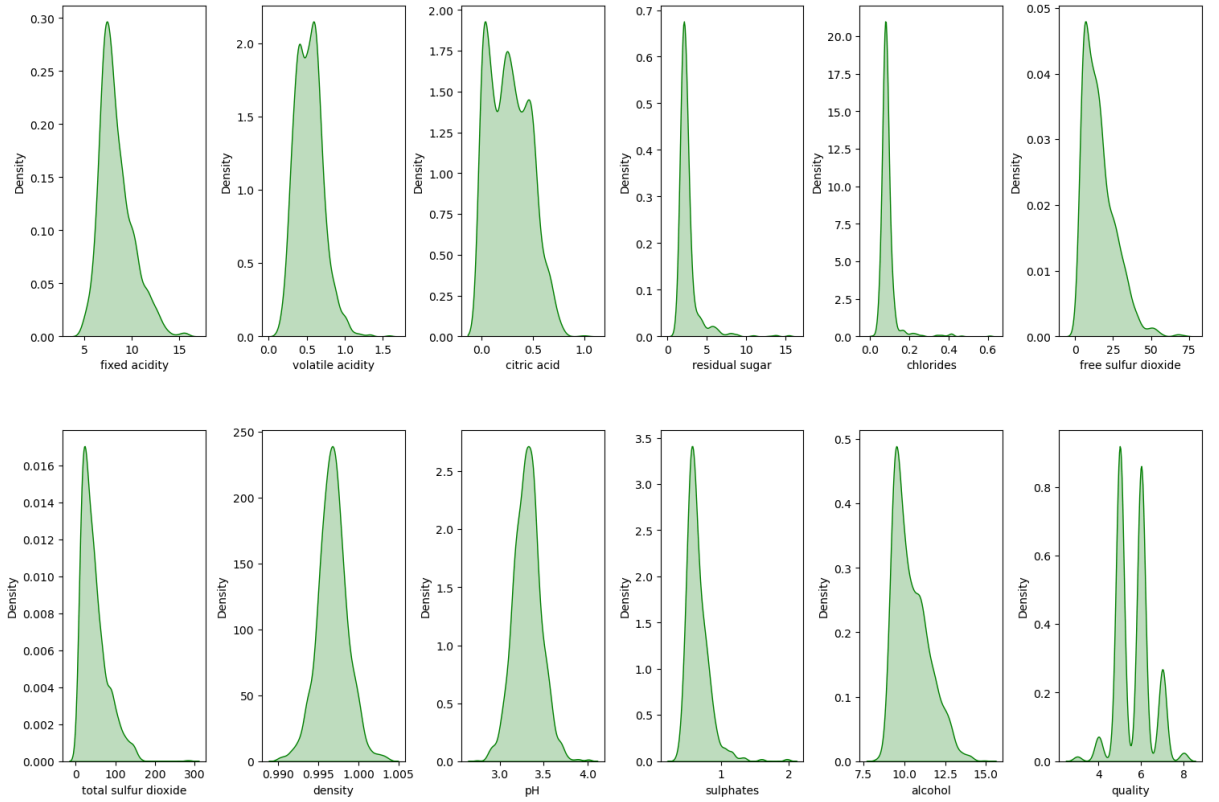
In [13]:
```
index=0
labels = df['quality']
features = df.drop('quality', axis=1)
for col in features.items():
    plt.figure(figsize=(10,5))
    sns.barplot(x=labels, y=col[index], data=df, color="deeppink")
plt.tight_layout()
plt.show()
```



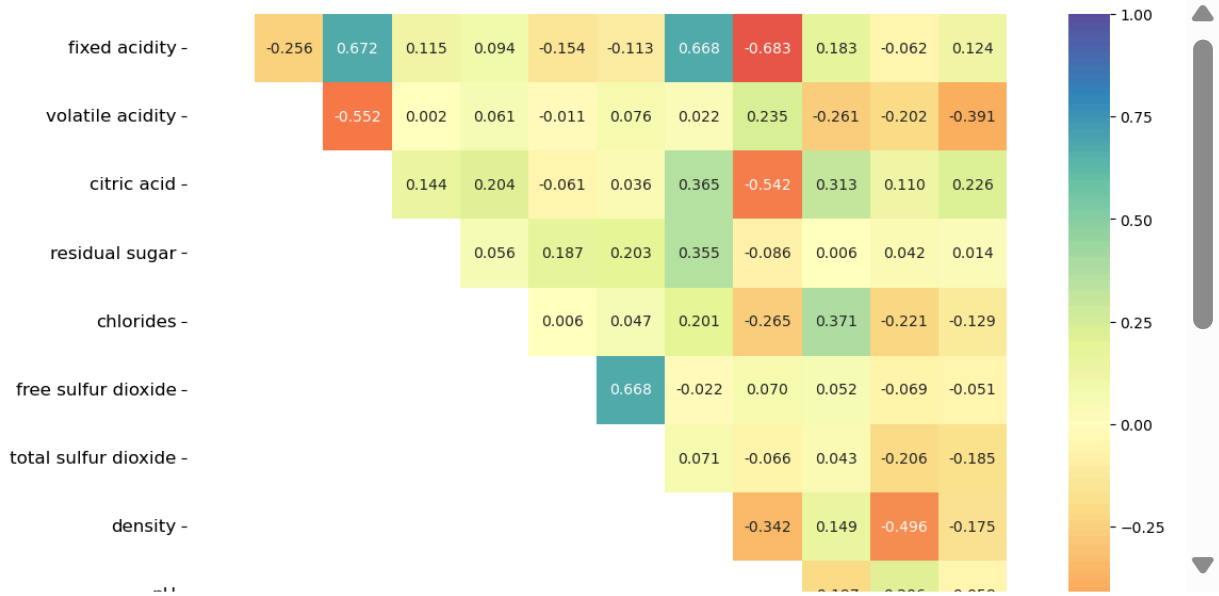
In [14]:
```
fig, ax = plt.subplots(ncols=6, nrows=2, figsize=(15,10))
index = 0
ax = ax.flatten()
for col, value in df.items():
    sns.boxplot(y=col, data=df, ax=ax[index])
    index += 1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
plt.show()
```



In [15]:
```
fig, ax = plt.subplots(ncols=6, nrows=2, figsize=(15,10))
index = 0
ax = ax.flatten()
for col, value in df.items():
    sns.distplot(value, ax=ax[index], hist=False, color="g", kde_kws={"shade": True})
    index += 1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
plt.show()
```



In [16]:
```
lower_triangle = np.tril(df.corr())
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(), vmin=-1, vmax=1, annot=True, square=True, fmt='0.3f',
            annot_kws={'size':10}, cmap="Spectral", mask=lower_triangle)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```



In [17]:
```
df = df.drop('free sulfur dioxide', axis=1)
df
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	42.0	0.99549	3.39	0.66	11.0	6

In [18]:
```
df.shape
```

Out[18]:
```
(1599, 11)
```

In [19]:
```
z=np.abs(zscore(df))
threshold=3
np.where(z>3)
```

Out[19]:
```
(array([  13,   15,   17,   17,   19,   33,   38,   42,   43,   45,   81,
          81,   83,   86,   88,   91,   92,   95,  106,  106,  109,  120,
         126,  127,  142,  144,  147,  151,  151,  151,  151,  163,  164,
         169,  169,  181,  199,  226,  226,  240,  243,  244,  258,  258,
         274,  281,  291,  324,  325,  339,  340,  347,  354,  374,  381,
         391,  396,  400,  442,  442,  451,  459,  467,  480,  480,  494,
         515,  517,  544,  554,  554,  555,  555,  557,  557,  568,  588,
         591,  595,  608,  614,  636,  639,  649,  649,  651,  652,  652,
         652,  672,  672,  684,  690,  690,  692,  692,  695,  723,  724,
         730,  754,  776,  777,  795,  821,  832,  836,  837,  889,  899,
         911,  917,  923, 1017, 1018, 1043, 1051, 1051, 1071, 1074, 1079,
        1079, 1081, 1081, 1111, 1114, 1165, 1186, 1235, 1244, 1244, 1260,
        1269, 1269, 1270, 1270, 1288, 1289, 1299, 1299, 1300, 1312, 1316,
        1319, 1319, 1321, 1367, 1370, 1370, 1372, 1372, 1374, 1374, 1434,
        1434, 1435, 1435, 1469, 1474, 1474, 1476, 1476, 1478, 1493, 1496,
        1505, 1558, 1570, 1574, 1589], dtype=int64),
 array([ 8,  5,  4,  8,  4,  3,  1,  4,  8,  7,  4,  8,  4,  8,  8,  8,  8,
         7,  4,  8,  5,  1,  1,  1,  9,  9,  4,  2,  4,  7,  8,  3,  3,  4,
         8,  4,  1,  4,  8,  4,  0,  0,  4,  8,  3,  4,  4,  3,  3,  8,  8,
         0,  5,  0,  0,  0,  3,  3,  0,  6,  4, 10,  9,  3,  6,  3,  5, 10,
         0,  0,  6,  0,  6,  0,  6,  4,  9,  5,  3,  6,  8,  5,  8,  3,  5,
         5,  0,  3,  9,  1,  5,  5,  1, 10,  4,  8,  7,  8,  1,  4,  4,  4,
         4,  8,  9, 10,  6,  6,  6, 10,  3,  3,  3,  6,  6,  3,  4,  8,  3,
         3,  3,  5,  3,  5,  7,  6,  4,  3,  3,  3,  5,  4,  6,  9,  6,  9,
         8,  8,  1, 10,  7,  1,  7,  4,  8,  7,  8,  4,  8,  4,  8,  4, 10,
         3,  6,  3,  6, 10,  3,  6,  3,  6, 10,  5,  5, 10,  4,  4,  3,  3],
       dtype=int64))
```

In [20]:
```
df=df[(z<3).all(axis=1)]
df
```

Out[20]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	42.0	0.99549	3.39	0.66	11.0	6

1464 rows × 11 columns

In [21]:
```
df.shape
```

Out[21]:
```
(1464, 11)
```

In [22]:
```
X = df.drop('quality', axis=1)
Y = df['quality']
```

In [23]:
```
Y.value_counts()
```

Out[23]:
```
5    624
6    590
7    187
4     47
8     16
Name: quality, dtype: int64
```

In [24]:
```
oversample = SMOTE()
X, Y = oversample.fit_resample(X, Y)
```

In [25]:
```
Y.value_counts()
```

Out[25]:
```
5    624
6    624
7    624
4    624
8    624
Name: quality, dtype: int64
```

In [26]:
```
Y = Y.apply(lambda y_value:1 if y_value>=7 else 0)
Y
```

Out[26]:
```
0       0
1       0
2       0
3       0
4       0
       ..
3115    1
3116    1
3117    1
3118    1
3119    1
Name: quality, Length: 3120, dtype: int64
```

In [27]:
```
scaler = StandardScaler()
X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
X
```

Out[27]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	total sulfur dioxide	density	pH	sulphates	alcol
0	-0.674572	1.058550	-1.509054	-0.632486	-0.143104	-0.109382	0.826963	1.480905	-0.795906	-1.2499
1	-0.424371	2.083079	-1.509054	0.132699	1.090815	1.146013	0.222033	-0.746055	0.113128	-0.8978
2	-0.424371	1.400060	-1.305310	-0.195238	0.754292	0.651463	0.343019	-0.315030	-0.114130	-0.8978
3	1.702331	-1.332019	1.343363	-0.632486	-0.199192	0.879717	0.947949	-1.033405	-0.644400	-0.8978
4	-0.674572	1.058550	-1.509054	-0.632486	-0.143104	-0.109382	0.826963	1.480905	-0.795906	-1.2499
...	...	...	...	...	...	...	...	...	...	
3115	0.921982	-0.644861	0.932171	0.103514	-0.389781	-0.783987	-0.319306	-1.009344	0.027444	0.6519
3116	0.804813	-0.756219	0.873094	-0.256611	-0.529566	-0.861887	-0.085374	-0.465629	0.020764	0.4327
3117	0.314984	-1.197535	1.043960	0.268330	0.113565	0.729990	-0.345116	-1.038017	0.578954	1.2893
3118	2.278778	-1.169774	2.036494	-0.206492	-0.300370	-0.367845	1.163385	-2.754867	1.286924	-0.6477
3119	0.290088	-0.801285	1.001373	0.251241	0.012776	-0.756101	-0.767638	-0.995623	1.468821	1.1507

3120 rows × 10 columns

In [28]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=21
```

In [29]:
```
def classify(model, X, Y):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_stat
    model.fit(X_train, Y_train)
    pred = model.predict(X_test)
    acc_score = (accuracy_score(Y_test, pred))*100
    print("Accuracy Score:", acc_score)
    class_report = classification_report(Y_test, pred)
    print("\nClassification Report:\n", class_report)
    cv_score = (cross_val_score(model, X, Y, cv=5).mean())*100
    print("Cross Validation Score:", cv_score)
    result = acc_score - cv_score
    print("\nAccuracy Score - Cross Validation Score is", result)
```

In [30]:
```
model=LogisticRegression()
classify(model, X, Y)
```

```
Accuracy Score: 90.06410256410257

Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.90      0.92       391
           1       0.84      0.90      0.87       233

    accuracy                           0.90       624
   macro avg       0.89      0.90      0.90       624
weighted avg       0.90      0.90      0.90       624

Cross Validation Score: 87.88461538461539

Accuracy Score - Cross Validation Score is 2.1794871794871824
```

In [31]:
```
model=SVC(C=1.0, kernel='rbf', gamma='auto', random_state=42)
classify(model, X, Y)
```

```
Accuracy Score: 91.98717948717949

Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.93      0.94       391
           1       0.89      0.90      0.89       233

    accuracy                           0.92       624
   macro avg       0.91      0.92      0.91       624
weighted avg       0.92      0.92      0.92       624

Cross Validation Score: 90.28846153846153

Accuracy Score - Cross Validation Score is 1.698717948717956
```

In [32]:
```
model=DecisionTreeClassifier(random_state=21, max_depth=15)
classify(model, X, Y)
```

```
Accuracy Score: 91.82692307692307

Classification Report:
               precision    recall  f1-score   support

           0       0.95      0.92      0.93       391
           1       0.88      0.91      0.89       233

    accuracy                           0.92       624
   macro avg       0.91      0.92      0.91       624
weighted avg       0.92      0.92      0.92       624

Cross Validation Score: 88.58974358974359

Accuracy Score - Cross Validation Score is 3.237179487179475
```

In [33]:
```
model=RandomForestClassifier(max_depth=15, random_state=111)
classify(model, X, Y)
```

```
Accuracy Score: 95.67307692307693

Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.96      0.97       391
           1       0.93      0.96      0.94       233

    accuracy                           0.96       624
   macro avg       0.95      0.96      0.95       624
weighted avg       0.96      0.96      0.96       624

Cross Validation Score: 92.56410256410255

Accuracy Score - Cross Validation Score is 3.108974358974379
```

In [34]:
```
model=KNeighborsClassifier(n_neighbors=15)
classify(model, X, Y)
```

```
Accuracy Score: 91.34615384615384

Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.90      0.93       391
           1       0.85      0.93      0.89       233

    accuracy                           0.91       624
   macro avg       0.90      0.92      0.91       624
weighted avg       0.92      0.91      0.91       624

Cross Validation Score: 88.46153846153847

Accuracy Score - Cross Validation Score is 2.8846153846153726
```

In [35]:
```
model=ExtraTreesClassifier()
classify(model, X, Y)
```

```
Accuracy Score: 95.83333333333334

Classification Report:
               precision    recall  f1-score   support

           0       0.98      0.96      0.97       391
           1       0.93      0.96      0.95       233

    accuracy                           0.96       624
   macro avg       0.95      0.96      0.96       624
weighted avg       0.96      0.96      0.96       624

Cross Validation Score: 93.65384615384616

Accuracy Score - Cross Validation Score is 2.1794871794871824
```

In [36]:
```
Final_Model = SVC(decision_function_shape='ovo', gamma='scale', kernel='rbf', probabilit
Classifier = Final_Model.fit(X_train, Y_train)
fmod_pred = Final_Model.predict(X_test)
fmod_acc = (accuracy_score(Y_test, fmod_pred))*100
print("Accuracy score for the Best Model is:", fmod_acc)
```

```
[LibSVM]Accuracy score for the Best Model is: 91.98717948717949
```

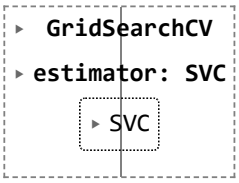
In [37]:
```
svc_param = {'kernel' : ['poly', 'sigmoid', 'rbf'],
             'gamma' : ['scale', 'auto'],
             'shrinking' : [True, False],
             'random_state' : [21,42,104],
             'probability' : [True, False],
             'decision_function_shape' : ['ovo', 'ovr'],
             'verbose' : [True, False]}
```

In [38]:
```
GSCV = GridSearchCV(SVC(), svc_param, cv=5)
```

In [39]:
```
GSCV.fit(X_train,Y_train)
```

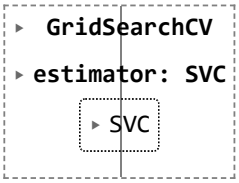
[LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSV
M][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibS
VM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Lib
SVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Li
bSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][L
ibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM]
[LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSV
M][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibS
VM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Lib
SVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Li
bSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][L
ibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM]
[LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSV
M][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibS
VM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Lib
SVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Li
bSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][L
ibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM]
[LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSV
M][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibS
VM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Lib
SVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Li
bSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][L
ibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM]
[LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSV
M][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibS
VM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Lib
SVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Li
bSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][L
ibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM]
[LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSV
M][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibS
VM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Lib
SVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Li
bSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][L
ibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM]
[LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSV
M][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibS
VM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Lib
SVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Li
bSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][L
ibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM]
[LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSV
M][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibS
VM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Lib
SVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Li
bSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][L
ibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM]
[LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSV
M][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibS
VM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Lib
SVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Li
bSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][L
ibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM]
[LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSV
M][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibS
VM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Lib
SVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Li
bSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][L
ibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM]
[LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSV
M][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibS
VM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Lib
SVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Li
bSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][L
ibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM]
[LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM]

Out[39]:



In [40]:
```
GridSearchCV(cv=5, estimator=SVC(),
             param_grid={'decision_function_shape': ['ovo', 'ovr'],
                         'gamma': ['scale', 'auto'],
                         'kernel': ['poly', 'sigmoid', 'rbf'],
                         'probability': [True, False],
                         'random_state': [21, 42, 104],
                         'shrinking': [True, False], 'verbose': [True, False]})
```

Out[40]:



In [41]:
```
GSCV.best_params_
```

Out[41]:
```
{'decision_function_shape': 'ovo',
 'gamma': 'scale',
 'kernel': 'rbf',
 'probability': True,
 'random_state': 21,
 'shrinking': True,
 'verbose': True}
```

In [42]:
```
Final_Model = SVC(decision_function_shape='ovo', gamma='scale', kernel='rbf', probabilit
Classifier = Final_Model.fit(X_train, Y_train)
fmod_pred = Final_Model.predict(X_test)
fmod_acc = (accuracy_score(Y_test, fmod_pred))*100
print("Accuracy score for the Best Model is:", fmod_acc)
```

```
[LibSVM]Accuracy score for the Best Model is: 91.98717948717949
```

In [44]:
```
!pip install scikit-plot
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: scikit-plot in c:\users\admin\appdata\roaming\python\p
ython311\site-packages (0.3.7)
Requirement already satisfied: matplotlib>=1.4.0 in c:\programdata\anaconda3\lib\site
-packages (from scikit-plot) (3.7.1)
Requirement already satisfied: scikit-learn>=0.18 in c:\programdata\anaconda3\lib\sit
e-packages (from scikit-plot) (1.2.2)
Requirement already satisfied: scipy>=0.9 in c:\programdata\anaconda3\lib\site-packag
es (from scikit-plot) (1.10.1)
Requirement already satisfied: joblib>=0.10 in c:\programdata\anaconda3\lib\site-pack
ages (from scikit-plot) (1.2.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\anaconda3\lib\site-
packages (from matplotlib>=1.4.0->scikit-plot) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-pack
ages (from matplotlib>=1.4.0->scikit-plot) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3\lib\site
-packages (from matplotlib>=1.4.0->scikit-plot) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site
-packages (from matplotlib>=1.4.0->scikit-plot) (1.4.4)
```

In [68]:
```
from sklearn.metrics import roc_curve
```

In [66]:
```
disp = metrics.plot_roc_curve(Final_Model, X_test, Y_test)
disp.figure_.suptitle("ROC Curve")
plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[66], line 1
----> 1 disp = metrics.plot_roc_curve(Final_Model, X_test, Y_test)
      2 disp.figure_.suptitle("ROC Curve")
      3 plt.show()

AttributeError: module 'sklearn.metrics' has no attribute 'plot_roc_curve'
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: