In [2]:
```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
Quality = pd.read_csv('WineQT.csv')
```

In [4]:
```
Quality
```

Out[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...	...	...
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82	9.5
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2

1143 rows × 13 columns

In [5]:
```
Quality.head()
```

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	qua
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	

In [6]:

```
Quality.tail()
```

Out[6]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82	9.5
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2

In [7]:

```
Quality.sample(5)
```

Out[7]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
894	7.2	0.57	0.05	2.3	0.081	16.0	36.0	0.99564	3.38	0.60	10.3
815	10.0	0.35	0.47	2.0	0.061	6.0	11.0	0.99585	3.23	0.52	12.0
832	6.5	0.88	0.03	5.6	0.079	23.0	47.0	0.99572	3.58	0.50	11.2
79	10.1	0.31	0.44	2.3	0.080	22.0	46.0	0.99880	3.32	0.67	9.7
172	7.3	0.66	0.00	2.0	0.084	6.0	23.0	0.99830	3.61	0.96	9.9

In [9]:

```
Quality.shape
```

Out[9]:

```
(1143, 13)
```

In [10]:

```
Quality.describe()
```

Out[10]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	
count	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	11
mean	8.311111	0.531339	0.268364	2.532152	0.086933	15.615486	45.914698	
std	1.747595	0.179633	0.196686	1.355917	0.047267	10.250486	32.782130	
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	
25%	7.100000	0.392500	0.090000	1.900000	0.070000	7.000000	21.000000	
50%	7.900000	0.520000	0.250000	2.200000	0.079000	13.000000	37.000000	
75%	9.100000	0.640000	0.420000	2.600000	0.090000	21.000000	61.000000	
max	15.900000	1.580000	1.000000	15.500000	0.611000	68.000000	289.000000	

In [11]:

```
Quality.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64  
 12  Id                    1143 non-null   int64  
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

In [12]:

```
Quality.isnull()
```

Out[12]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
1138	False	False	False	False	False	False	False	False	False	False	False
1139	False	False	False	False	False	False	False	False	False	False	False
1140	False	False	False	False	False	False	False	False	False	False	False
1141	False	False	False	False	False	False	False	False	False	False	False
1142	False	False	False	False	False	False	False	False	False	False	False

1143 rows × 13 columns

In [13]:

```
Quality.sum()
```

Out[13]:

```
fixed acidity             9499.600000
volatile acidity           607.320000
citric acid                306.740000
residual sugar            2894.250000
chlorides                   99.364000
free sulfur dioxide      17848.500000
total sulfur dioxide     52480.500000
density                   1139.262860
pH                        3784.490000
sulphates                  751.760000
alcohol                  11935.333333
quality                   6466.000000
Id                      920080.000000
dtype: float64
```

In [14]:

```
Quality1 = pd.get_dummies(Quality,drop_first=True)
```

In [15]:

```
Quality1
```

Out[15]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...	...	...
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82	9.5
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2

1143 rows × 13 columns

In [16]:

```
Quality1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64  
 12  Id                    1143 non-null   int64  
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

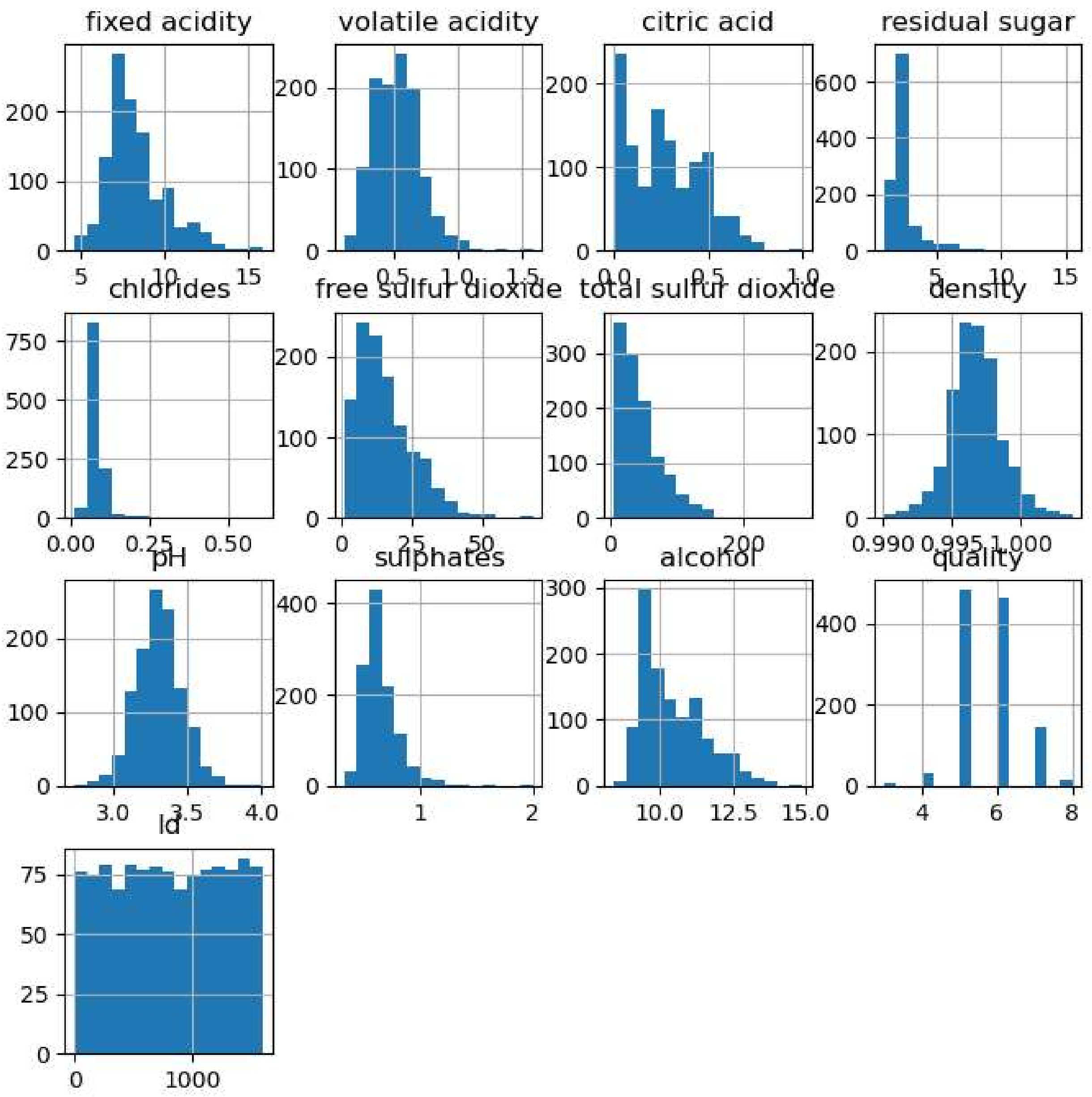
In [19]:

```
for col in Quality1.columns:
    if Quality1[col].isnull().sum() > 0:
        Quality1[col] = Quality1[col].fillna(Quality1[col].mean())
Quality1.isnull().sum().sum()
```

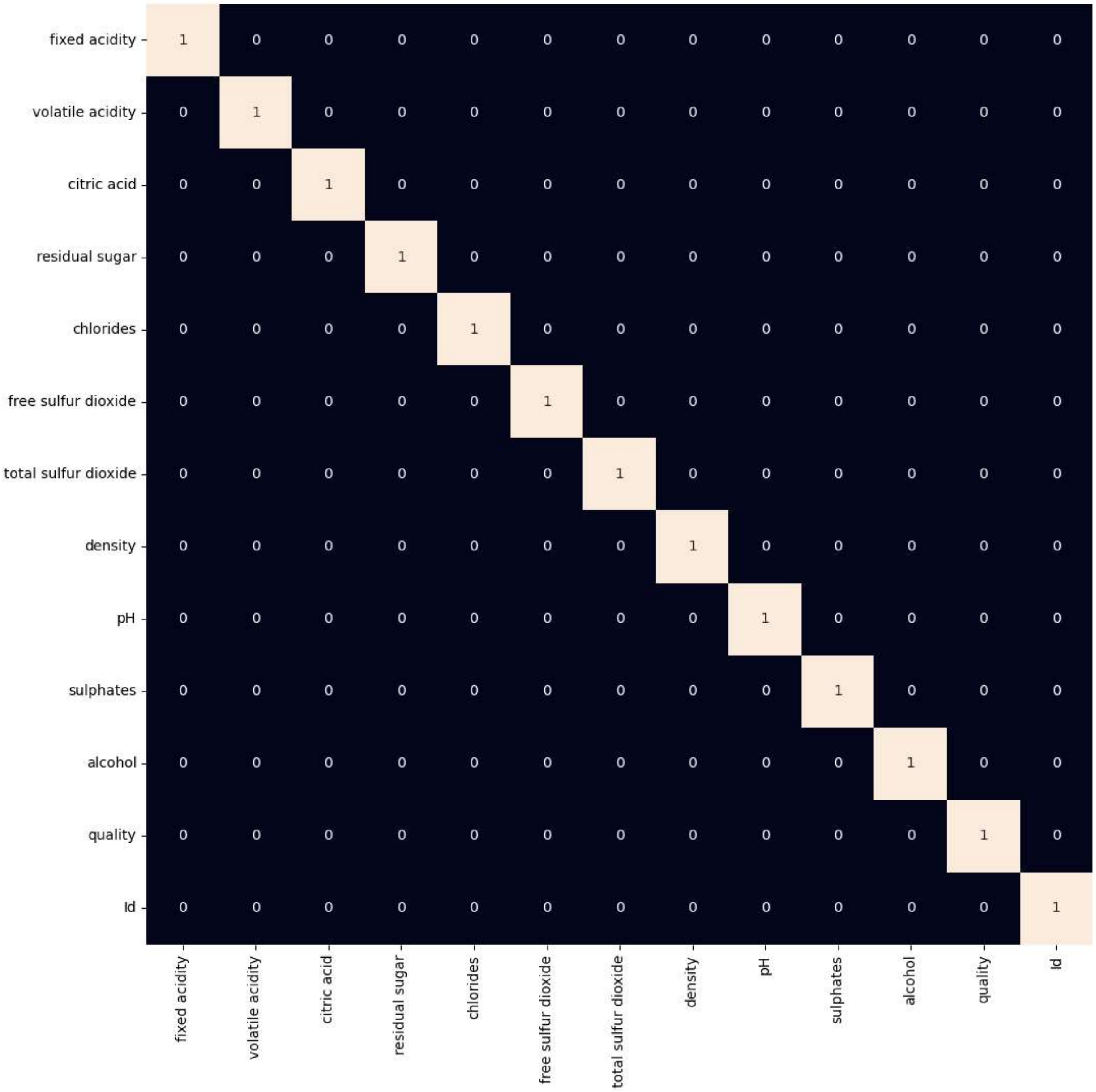
Out[19]:

```
0
```

In [20]:
```
Quality1.hist(bins=15, figsize=(8, 8))
plt.show()
```



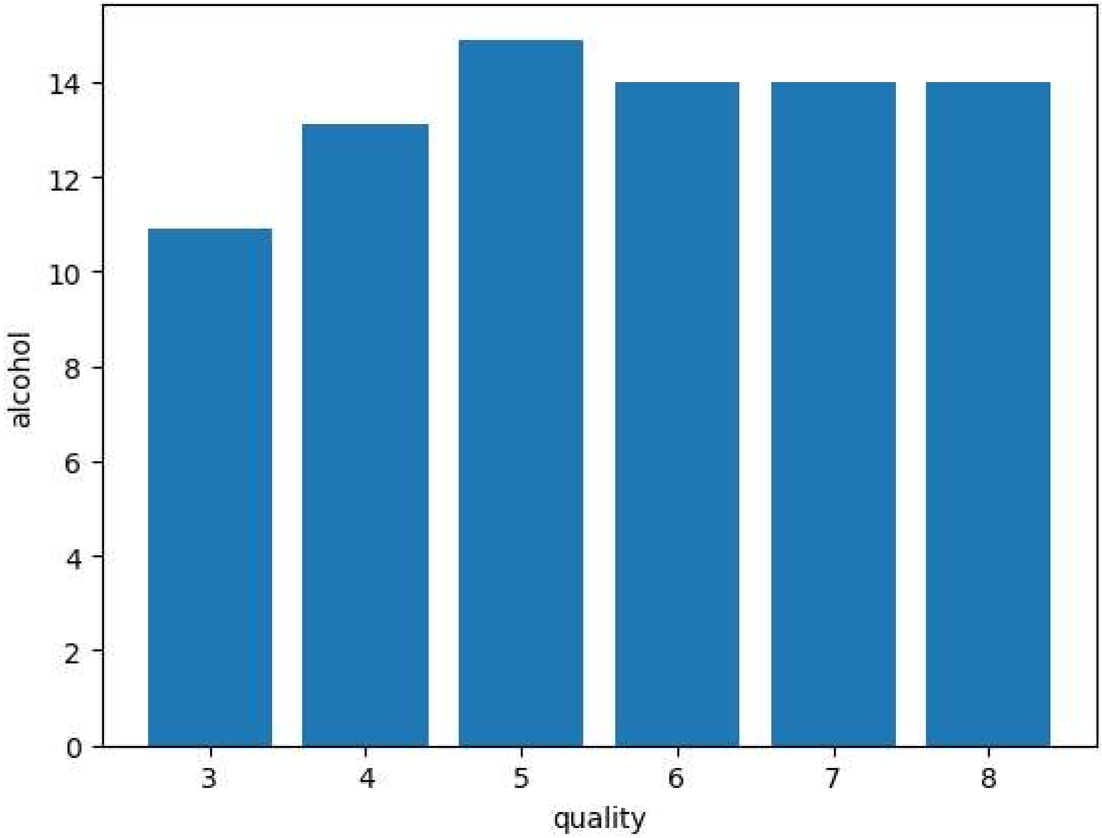
In [24]:
```
plt.figure(figsize=(12, 12))
sb.heatmap(Quality1.corr() > 0.7, annot=True, cbar=False)
plt.show()
```



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
fixed acidity	1	0	0	0	0	0	0	0	0	0	0	0	0
volatile acidity	0	1	0	0	0	0	0	0	0	0	0	0	0
citric acid	0	0	1	0	0	0	0	0	0	0	0	0	0
residual sugar	0	0	0	1	0	0	0	0	0	0	0	0	0
chlorides	0	0	0	0	1	0	0	0	0	0	0	0	0
free sulfur dioxide	0	0	0	0	0	1	0	0	0	0	0	0	0
total sulfur dioxide	0	0	0	0	0	0	1	0	0	0	0	0	0
density	0	0	0	0	0	0	0	1	0	0	0	0	0
pH	0	0	0	0	0	0	0	0	1	0	0	0	0
sulphates	0	0	0	0	0	0	0	0	0	1	0	0	0
alcohol	0	0	0	0	0	0	0	0	0	0	1	0	0
quality	0	0	0	0	0	0	0	0	0	0	0	1	0
Id	0	0	0	0	0	0	0	0	0	0	0	0	1

In [23]:

```
plt.bar(Quality1['quality'], Quality1['alcohol'])
plt.xlabel('quality')
plt.ylabel('alcohol')
plt.show()
```



```
In [26]: Quality = Quality1.drop('total sulfur dioxide', axis=1)
```

```
In [27]: Quality['best quality'] = [1 if x > 5 else 0 for x in Quality.quality]
```

```
In [28]: Quality.replace({'white': 1, 'red': 0}, inplace=True)
```

```
In [29]: features = Quality.drop(['quality', 'best quality'], axis=1)
         target = Quality['best quality']
```

```
In [31]: xtrain, xtest, ytrain, ytest = train_test_split(features, target, test_size=0.2, rando
```

```
In [34]: xtrain.shape

Out[34]: (914, 11)
```

```
In [33]: xtest.shape

Out[33]: (229, 11)
```

```
In [35]: ytrain.shape

Out[35]: (914,)
```

```
In [36]: ytest.shape

Out[36]: (229,)
```

```
In [39]: xtest
```

Out[39]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	density	pH	sulphates	alcohol	Id
471	7.2	0.57	0.06	1.6	0.076	9.0	0.99720	3.36	0.70	9.6	662
192	11.5	0.18	0.51	4.0	0.104	4.0	0.99960	3.28	0.97	10.1	269
1035	6.5	0.90	0.00	1.6	0.052	9.0	0.99467	3.50	0.63	10.9	1455
476	8.2	0.73	0.21	1.7	0.074	5.0	0.99680	3.20	0.52	9.5	671
512	8.4	0.56	0.04	2.0	0.082	10.0	0.99760	3.22	0.44	9.6	720
...	...	...	...	...	...	...	...	...	...	...	...
281	7.7	0.69	0.05	2.7	0.075	15.0	0.99740	3.26	0.61	9.1	404
176	7.1	0.60	0.00	1.8	0.074	16.0	0.99720	3.47	0.70	9.9	251
537	8.3	0.65	0.10	2.9	0.089	17.0	0.99803	3.29	0.55	9.5	753
1043	7.3	0.48	0.32	2.1	0.062	31.0	0.99728	3.30	0.65	10.0	1466
801	8.5	0.28	0.35	1.7	0.061	6.0	0.99524	3.30	0.74	11.8	1134

229 rows × 11 columns

In [40]:

```
ytest
```

Out[40]:

```
471     1
192     1
1035    1
476     0
512     0
       ..
281     0
176     1
537     0
1043    1
801     1
Name: best quality, Length: 229, dtype: int64
```

In [41]:

```
norm = MinMaxScaler()
xtrain = norm.fit_transform(xtrain)
xtest = norm.transform(xtest)
```

In [50]:

```
models = [LogisticRegression(),SVC(kernel='rbf')]
for i in range(2):
 models[i].fit(xtrain, ytrain)
 print(f'models[i] : ')
 print('Training Accuracy : ', metrics.roc_auc_score(ytrain, models[i].predict(xtrain)
 print('Validation Accuracy : ', metrics.roc_auc_score(
 ytest, models[i].predict(xtest)))
 print()
```

```
models[i] :
Training Accuracy :  0.7546950559364851
Validation Accuracy :  0.7255154639175256

models[i] :
Training Accuracy :  0.7648213641284736
Validation Accuracy :  0.7358247422680412
```

In [51]:

```
print(metrics.classification_report(ytest,
 models[1].predict(xtest)))
```

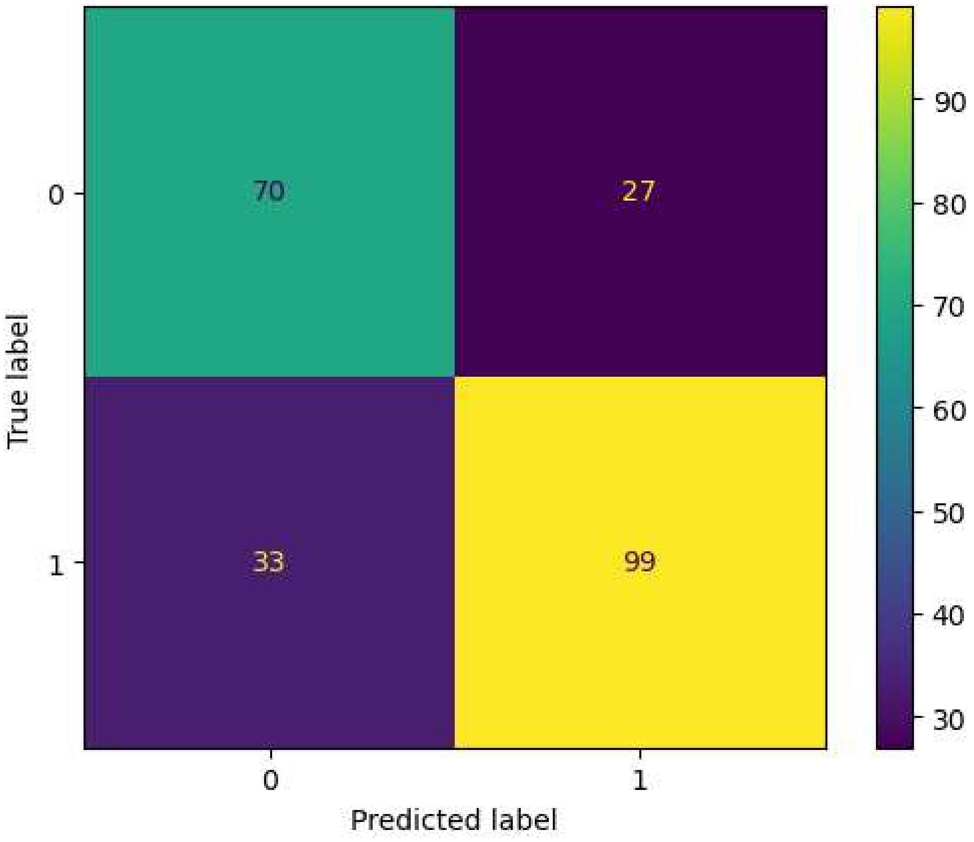
```
              precision    recall  f1-score   support

           0       0.68      0.72      0.70        97
           1       0.79      0.75      0.77       132

    accuracy                           0.74       229
   macro avg       0.73      0.74      0.73       229
weighted avg       0.74      0.74      0.74       229
```

In [52]:

```
metrics.plot_confusion_matrix(models[1], xtest, ytest)
plt.show()
```



In [68]:

```
for a in range(len(Quality1.corr().columns)):
    for b in range(a):
        if abs(Quality.corr().iloc[a,b]) >0.7:
            name = Quality.corr().columns[a]
            print(name)
```

```
best quality
```

In [ ]:

In [ ]: