Task2-Wine Quality Prediction

In [2]:
```
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]:
```
df = pd.read_csv("WineQT.csv")
print(df.head())
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlori
des  \
0            7.4              0.70         0.00             1.9      0.
076
1            7.8              0.88         0.00             2.6      0.
098
2            7.8              0.76         0.04             2.3      0.
092
3           11.2              0.28         0.56             1.9      0.
075
4            7.4              0.70         0.00             1.9      0.
076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates
\
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
```

In [4]:
```
df.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 [5]:
```
df.describe().T
```

Out[5]:

	count	mean	std	min	25%	50%	75%	
fixed acidity	1143.0	8.311111	1.747595	4.60000	7.10000	7.90000	9.100000	15.
volatile acidity	1143.0	0.531339	0.179633	0.12000	0.39250	0.52000	0.640000	1.
citric acid	1143.0	0.268364	0.196686	0.00000	0.09000	0.25000	0.420000	1.
residual sugar	1143.0	2.532152	1.355917	0.90000	1.90000	2.20000	2.600000	15.
chlorides	1143.0	0.086933	0.047267	0.01200	0.07000	0.07900	0.090000	0.
free sulfur dioxide	1143.0	15.615486	10.250486	1.00000	7.00000	13.00000	21.000000	68.
total sulfur dioxide	1143.0	45.914698	32.782130	6.00000	21.00000	37.00000	61.000000	289.
density	1143.0	0.996730	0.001925	0.99007	0.99557	0.99668	0.997845	1.
pH	1143.0	3.311015	0.156664	2.74000	3.20500	3.31000	3.400000	4.
sulphates	1143.0	0.657708	0.170399	0.33000	0.55000	0.62000	0.730000	2.
alcohol	1143.0	10.442111	1.082196	8.40000	9.50000	10.20000	11.100000	14.
quality	1143.0	5.657043	0.805824	3.00000	5.00000	6.00000	6.000000	8.
Id	1143.0	804.969379	463.997116	0.00000	411.00000	794.00000	1209.500000	1597.

EDTA-Analysis

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
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
Id                      0
dtype: int64
```

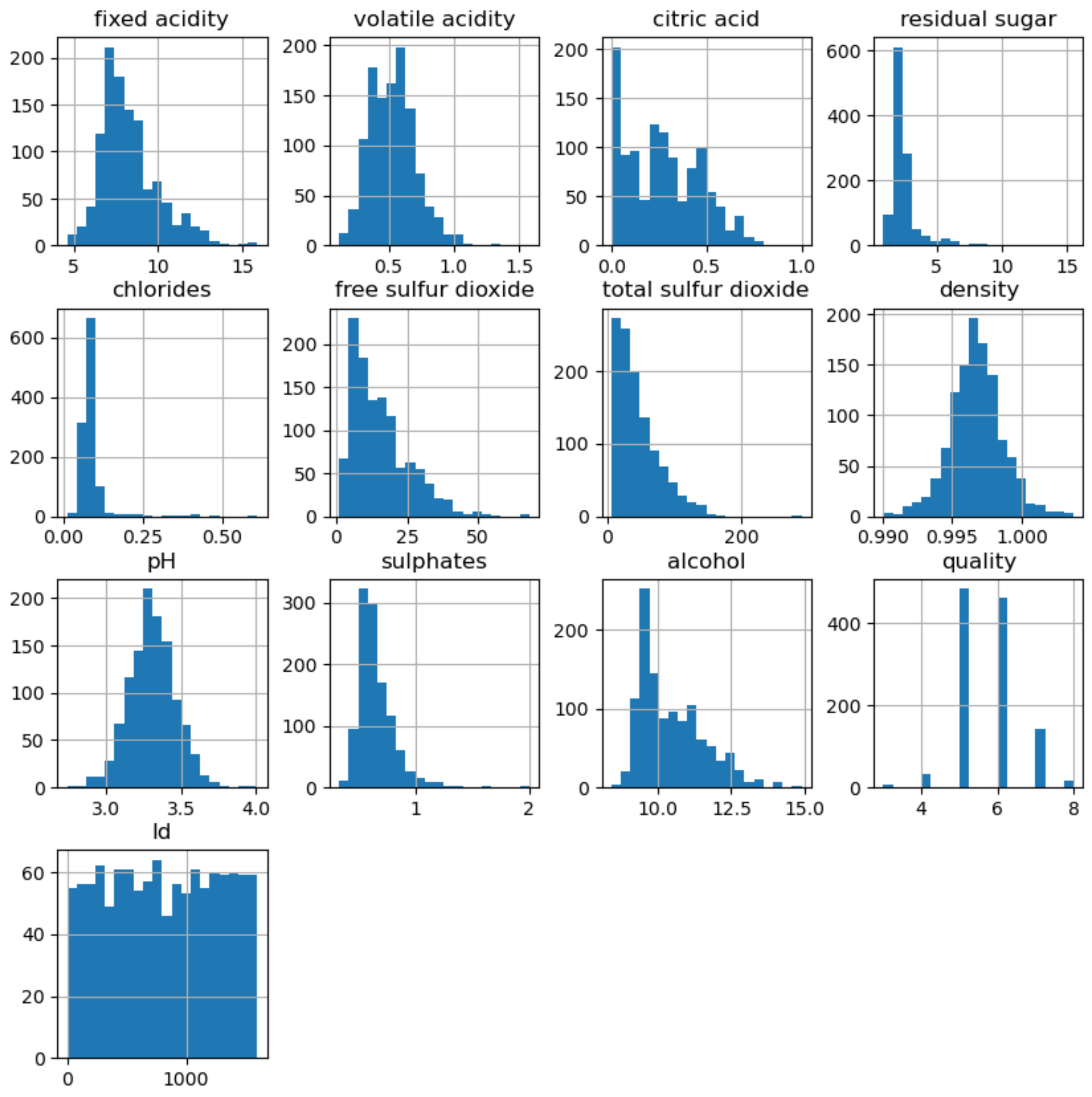
In [7]:
```
for col in df.columns:
  if df[col].isnull().sum() > 0:
    df[col] = df[col].fillna(df[col].mean())

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

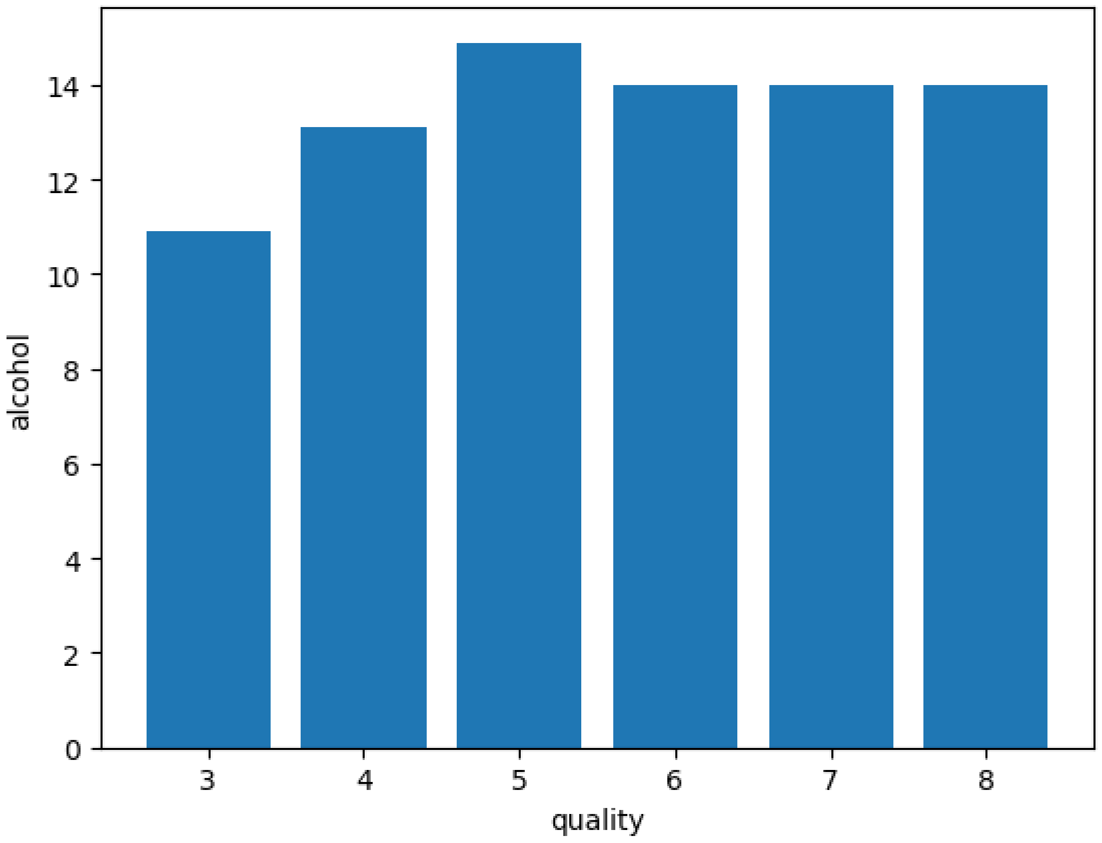
Out[7]:
```
0
```

Histogram

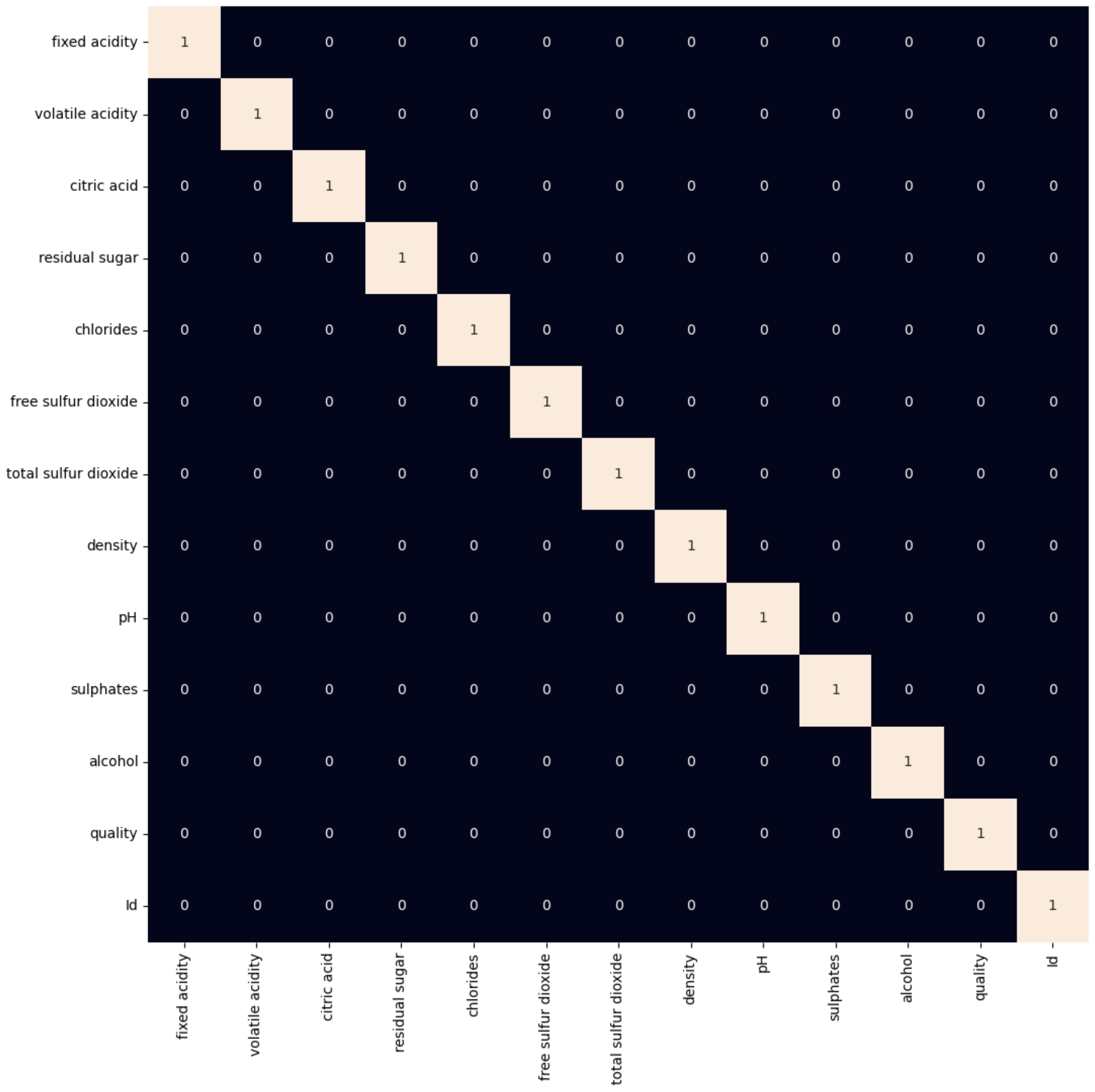
In [8]:
```
df.hist(bins=20, figsize=(10, 10))
plt.show()
```



In [9]:
```
plt.bar(df['quality'], df['alcohol'])
plt.xlabel('quality')
plt.ylabel('alcohol')
plt.show()
```



In [10]:
```
plt.figure(figsize=(12, 12))
sb.heatmap(df.corr() > 0.7, annot=True, cbar=False)
plt.show()
```



In [11]:
```
df = df.drop('total sulfur dioxide', axis=1)
```

In [12]:
```
df['best quality'] = [1 if x > 5 else 0 for x in df.quality]
```

In [13]:
```
df.replace({'white': 1, 'red': 0}, inplace=True)
```

In [14]:
```
features = df.drop(['quality', 'best quality'], axis=1)
target = df['best quality']

xtrain, xtest, ytrain, ytest = train_test_split(
    features, target, test_size=0.2, random_state=40)

xtrain.shape, xtest.shape
```

Out[14]:
```
((914, 11), (229, 11))
```

In [15]:
```
#Normalizing the data
norm = MinMaxScaler()
xtrain = norm.fit_transform(xtrain)
xtest = norm.transform(xtest)
```

In [17]:
```
models = [LogisticRegression(),SVC(kernel='rbf')]

for i in range(3):
    models[i].fit(xtrain, ytrain)

    print(f'{models[i]} : ')
    print('Training Accuracy : ', metrics.roc_auc_score(ytrain, models[i].p
    print('Validation Accuracy : ', metrics.roc_auc_score(
        ytest, models[i].predict(xtest)))
    print()
```

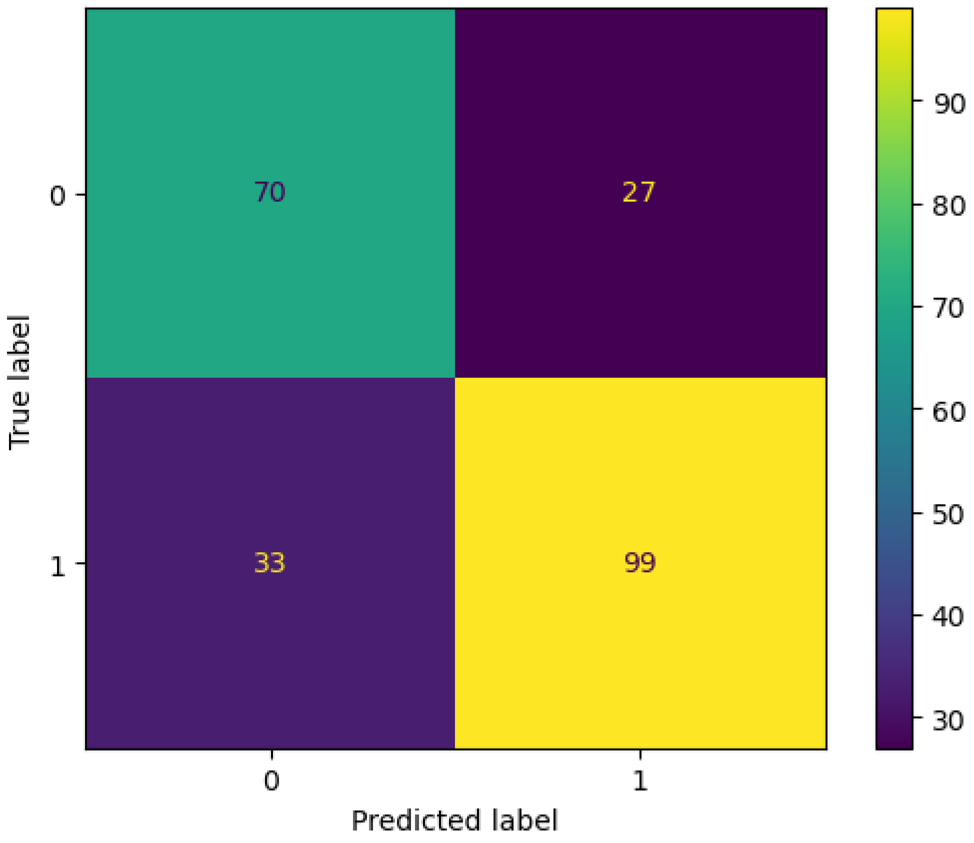
```
LogisticRegression() :
Training Accuracy :  0.7546950559364851
Validation Accuracy :  0.7255154639175256

SVC() :
Training Accuracy :  0.7648213641284736
Validation Accuracy :  0.7358247422680412


--------------------------------------------------------------------------
-
IndexError                                Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_23416\723548108.py in <module>
      2 
      3 for i in range(3):
----> 4     models[i].fit(xtrain, ytrain)
      5 
      6     print(f'{models[i]} : ')

IndexError: list index out of range
```

In [18]:
```
metrics.plot_confusion_matrix(models[1], xtest, ytest)
plt.show()
```



In [19]:
```
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 [ ]: