Elisetty Naga Jyothi

21bce9019

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

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

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

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

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

```
df.shape
```

```
(1599, 12)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
```

```
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

```
df.quality.value_counts()
```

```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

```
df.isnull().any()
```

```
fixed acidity           False
volatile acidity        False
citric acid             False
residual sugar          False
chlorides               False
free sulfur dioxide     False
total sulfur dioxide    False
density                 False
pH                      False
sulphates               False
alcohol                 False
quality                 False
dtype: bool
```

```
df.describe()
```

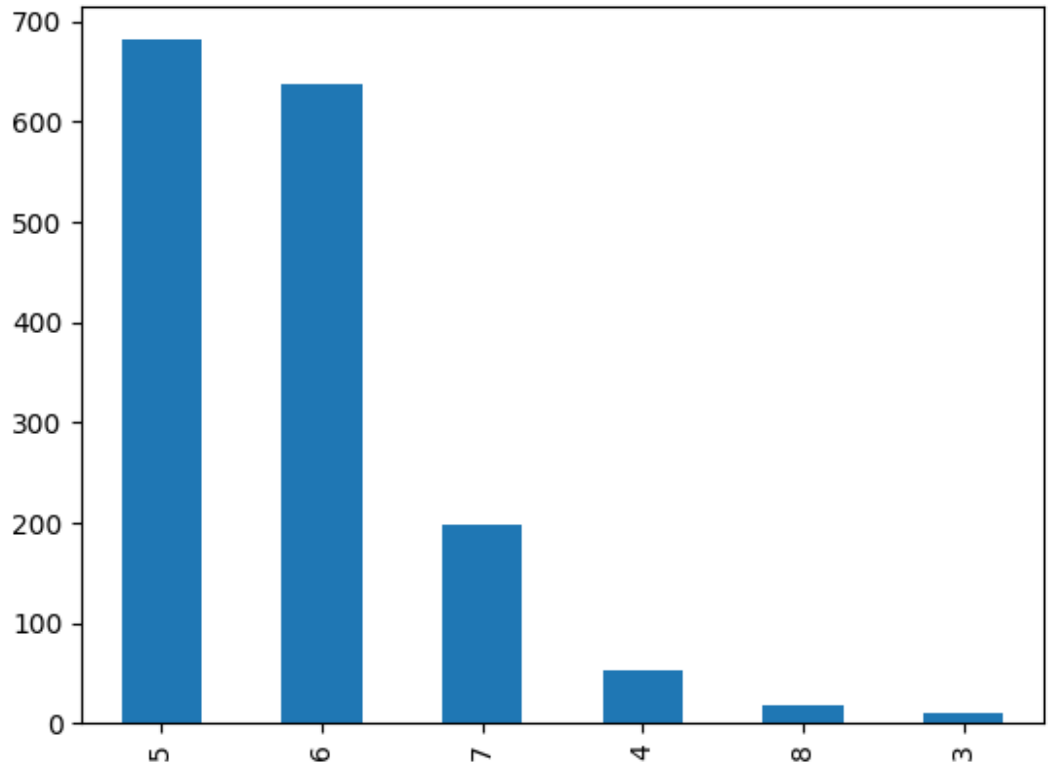
```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1599.000000       1599.000000  1599.000000     1599.000000
mean        8.319637          0.527821     0.270976        2.538806
std         1.741096          0.179060     0.194801        1.409928
min         4.600000 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

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

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

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

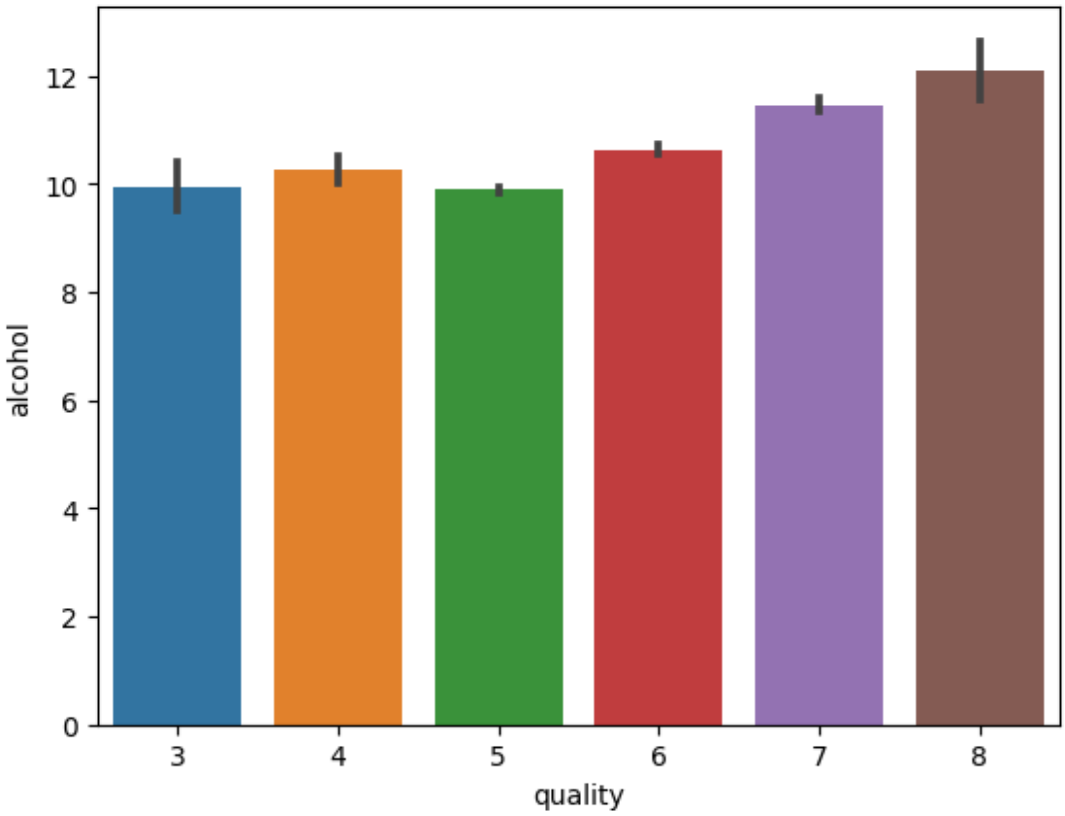
```
df["quality"].value_counts().plot(kind='bar')
```

```
<Axes: >
```




```
sns.barplot(x=df["quality"],y=df["alcohol"])
```

```
<Axes: xlabel='quality', ylabel='alcohol'>
```




```
df.corr()
```

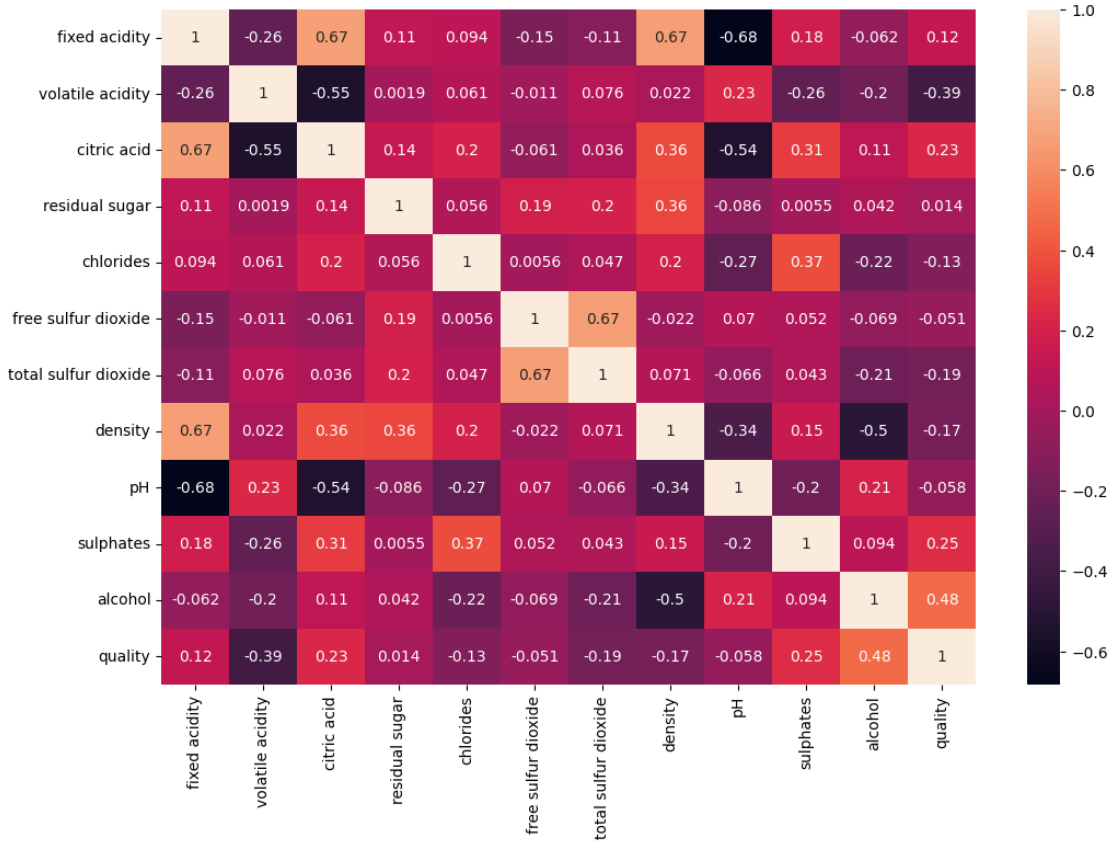
	fixed acidity	volatile acidity	citric acid	\
fixed acidity	1.000000	-0.256131	0.671703	
volatile acidity	-0.256131	1.000000	-0.552496	
citric acid	0.671703	-0.552496	1.000000	
residual sugar	0.114777	0.001918	0.143577	
chlorides	0.093705	0.061298	0.203823	
free sulfur dioxide	-0.153794	-0.010504	-0.060978	
total sulfur dioxide	-0.113181	0.076470	0.035533	
density	0.668047	0.022026	0.364947	
pH	-0.682978	0.234937	-0.541904	
sulphates	0.183006	-0.260987	0.312770	
alcohol	-0.061668	-0.202288	0.109903	
quality	0.124052	-0.390558	0.226373	

	residual sugar	chlorides	free sulfur dioxide	\
fixed acidity	0.114777	0.093705	-0.153794	
volatile acidity	0.001918	0.061298	-0.010504	
citric acid	0.143577	0.203823	-0.060978	
residual sugar	1.000000	0.055610	0.187049	
chlorides	0.055610	1.000000	0.005562	
free sulfur dioxide	0.187049	0.005562	1.000000	
total sulfur dioxide	0.203028	0.047400	0.667666	
density	0.355283	0.200632	-0.021946	

```
pH                          -0.085652  -0.265026            0.070377
sulphates                    0.005527   0.371260            0.051658
alcohol                      0.042075  -0.221141           -0.069408
quality                      0.013732  -0.128907           -0.050656

                      total sulfur dioxide   density        pH  sulphates  \
fixed acidity                    -0.113181  0.668047 -0.682978   0.183006
volatile acidity                  0.076470  0.022026  0.234937  -0.260987
citric acid                       0.035533  0.364947 -0.541904   0.312770
residual sugar                    0.203028  0.355283 -0.085652   0.005527
chlorides                         0.047400  0.200632 -0.265026   0.371260
free sulfur dioxide               0.667666 -0.021946  0.070377   0.051658
total sulfur dioxide              1.000000  0.071269 -0.066495   0.042947
density                           0.071269  1.000000 -0.341699   0.148506
pH                               -0.066495 -0.341699  1.000000  -0.196648
sulphates                         0.042947  0.148506 -0.196648   1.000000
alcohol                          -0.205654 -0.496180  0.205633   0.093595
quality                          -0.185100 -0.174919 -0.057731   0.251397

                       alcohol   quality
fixed acidity        -0.061668  0.124052
volatile acidity     -0.202288 -0.390558
citric acid           0.109903  0.226373
residual sugar        0.042075  0.013732
chlorides            -0.221141 -0.128907
free sulfur dioxide  -0.069408 -0.050656
total sulfur dioxide -0.205654 -0.185100
density              -0.496180 -0.174919
pH                    0.205633 -0.057731
sulphates             0.093595  0.251397
alcohol               1.000000  0.476166
quality               0.476166  1.000000
```

```
plt.figure(figsize=(12, 8))
cor=df.corr()
sns.heatmap(cor,annot=True)
```

```
<Axes: >
```

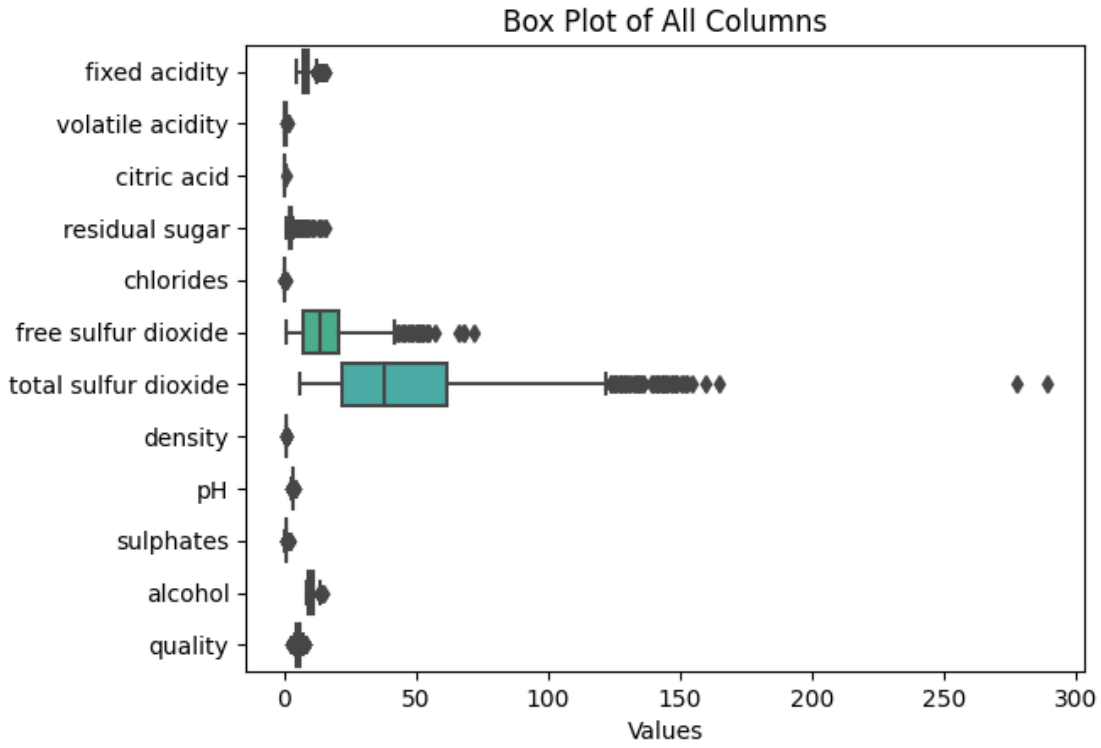



```
df.corr().quality.sort_values(ascending=False)
```

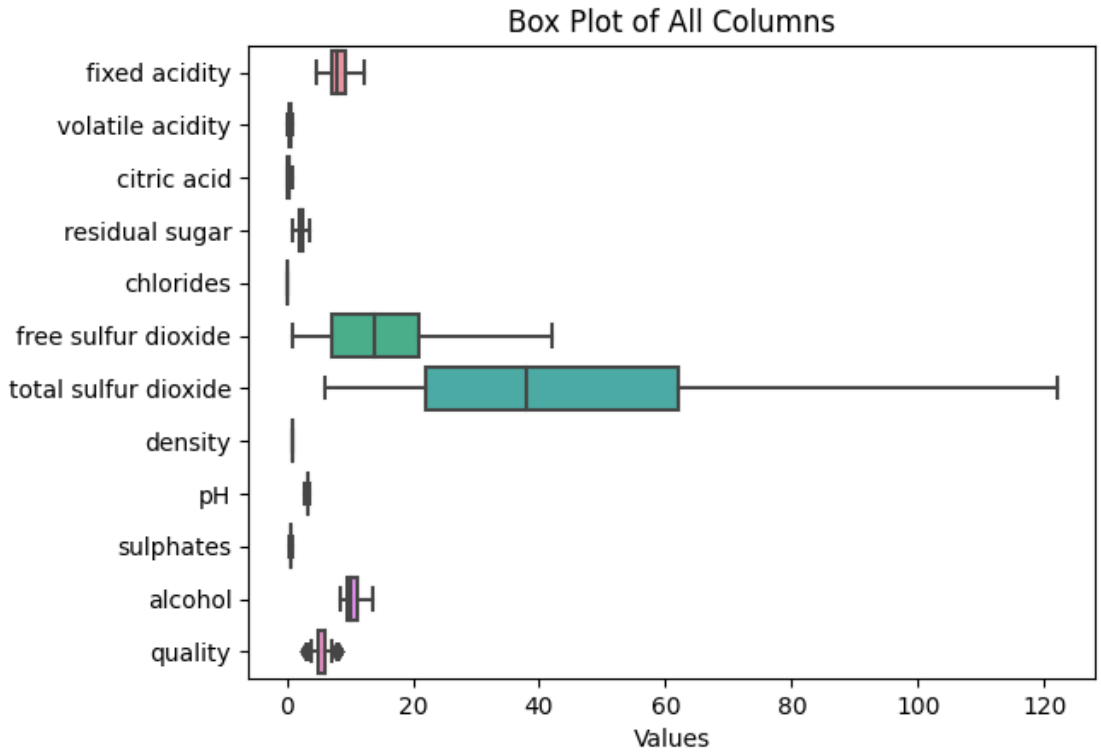
```
quality                 1.000000
alcohol                 0.476166
sulphates               0.251397
citric acid             0.226373
fixed acidity           0.124052
residual sugar          0.013732
free sulfur dioxide    -0.050656
pH                     -0.057731
chlorides              -0.128907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

```
sns.boxplot(data=df,orient='h')

plt.xlabel('Values')
plt.title('Box Plot of All Columns')
```

```
Text(0.5, 1.0, 'Box Plot of All Columns')
```




```
l=["fixed acidity","volatile acidity","citric acid","residual
sugar","chlorides","free sulfur dioxide","total sulfur
dioxide","density","pH","sulphates","alcohol"]

for i in l:
    q1=df[i].quantile(0.25)
    q3=df[i].quantile(0.75)
    iqr=q3-q1
    upperL=q3+1.5*iqr
    lowerL=q1-1.5*iqr
    df[i]=np.where(df[i]>upperL,upperL,np.where(df[i]<lowerL,lowerL,df[i]))

sns.boxplot(data=df, orient='h')

plt.xlabel('Values')
plt.title('Box Plot of All Columns')

Text(0.5, 1.0, 'Box Plot of All Columns')
```




```
x =df.drop(columns =['quality'],axis =1)

x.head()
```

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

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

```
0    5
1    5
2    5
3    6
4    5
Name: quality, dtype: int64
```

```
from sklearn.preprocessing import MinMaxScaler
scale =MinMaxScaler()
```

```
scaled_x = pd.DataFrame(scale.fit_transform(X),columns =X.columns)
scaled_x.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	0.361290	0.648045	0.000000	0.363636	0.4500	0.243902	0.241379	0.621085	0.769737	0.343284	0.196078
1	0.412903	0.849162	0.000000	0.618182	0.7250	0.585366	0.525862	0.509228	0.361842	0.522388	0.274510
2	0.412903	0.715084	0.043716	0.509091	0.6500	0.341463	0.413793	0.531600	0.440789	0.477612	0.274510
3	0.851613	0.178771	0.612022	0.363636	0.4375	0.390244	0.465517	0.643456	0.309211	0.373134	0.274510
4	0.361290	0.648045	0.000000	0.363636	0.4500	0.243902	0.241379	0.621085	0.769737	0.343284	0.196078

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(scaled_x,y,test_size =
0.2,random_state = 0)
```

```
x_train.shape
```

```
(1279, 11)
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

```
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```

```
/usr/local/lib/python3.10/dist-
packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs
failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
from sklearn.metrics import accuracy_score,
confusion_matrix,classification_report,roc_auc_score,roc_curve
```

```
accuracy_score(y_test,y_pred)
```

```
0.61875
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[  0,   0,   2,   0,   0,   0],
       [  0,   0,   7,   4,   0,   0],
       [  0,   0, 102,  33,   0,   0],
       [  0,   0,  40,  88,  14,   0],
       [  0,   0,   2,  17,   8,   0],
       [  0,   0,   0,   1,   2,   0]])
```

```
pd.crosstab(y_test,y_pred)
```

```
col_0      5   6   7
quality
3          2   0   0
4          7   4   0
5        102  33   0
6         40  88  14
7          2  17   8
8          0   1   2
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.00      0.00      0.00        11
           5       0.67      0.76      0.71       135
           6       0.62      0.62      0.62       142
           7       0.33      0.30      0.31        27
```

```
           8       0.00      0.00      0.00         3

    accuracy                           0.62       320
   macro avg       0.27      0.28      0.27       320
weighted avg       0.58      0.62      0.60       320


/usr/local/lib/python3.10/dist-
packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-
packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-
packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
sample_check=[[6.5, 0.6, 0.5, 2.2, 0.07, 15.0, 80.0, 0.996, 3.4, 0.7, 9.5],
             [8.0, 0.4, 0.9, 4.8, 0.945, 75.0, 55.0, 0.998, 7.2, 0.95, 15.2]]

for i in sample_check:
    x=model.predict([i])
    if(x>=6):
        print(x, "--> Good" )
    elif(x<6):
        print(x, "--> Not Good")
```

```
[5] --> Not Good
[5] --> Not Good

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
```