assignment-4

September 18, 2023

0.0.1 Dataset Inspection

[2]:
```
import pandas as pd
import numpy as np
from sklearn import preprocessing
```

[3]:
```
df = pd.read_csv("data.csv")
```

[19]:
```
df.shape
```

[19]:
```
(1599, 12)
```

[14]:
```
df.head(20)
```

[14]:
```
    fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0             7.4             0.700         0.00             1.9      0.076
1             7.8             0.880         0.00             2.6      0.098
2             7.8             0.760         0.04             2.3      0.092
3            11.2             0.280         0.56             1.9      0.075
4             7.4             0.700         0.00             1.9      0.076
5             7.4             0.660         0.00             1.8      0.075
6             7.9             0.600         0.06             1.6      0.069
7             7.3             0.650         0.00             1.2      0.065
8             7.8             0.580         0.02             2.0      0.073
9             7.5             0.500         0.36             6.1      0.071
10            6.7             0.580         0.08             1.8      0.097
11            7.5             0.500         0.36             6.1      0.071
12            5.6             0.615         0.00             1.6      0.089
13            7.8             0.610         0.29             1.6      0.114
14            8.9             0.620         0.18             3.8      0.176
15            8.9             0.620         0.19             3.9      0.170
16            8.5             0.280         0.56             1.8      0.092
17            8.1             0.560         0.28             1.7      0.368
18            7.4             0.590         0.08             4.4      0.086
19            7.9             0.320         0.51             1.8      0.341

    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
5                  13.0                  40.0   0.9978  3.51       0.56
6                  15.0                  59.0   0.9964  3.30       0.46
7                  15.0                  21.0   0.9946  3.39       0.47
8                   9.0                  18.0   0.9968  3.36       0.57
9                  17.0                 102.0   0.9978  3.35       0.80
10                 15.0                  65.0   0.9959  3.28       0.54
11                 17.0                 102.0   0.9978  3.35       0.80
12                 16.0                  59.0   0.9943  3.58       0.52
13                  9.0                  29.0   0.9974  3.26       1.56
14                 52.0                 145.0   0.9986  3.16       0.88
15                 51.0                 148.0   0.9986  3.17       0.93
16                 35.0                 103.0   0.9969  3.30       0.75
17                 16.0                  56.0   0.9968  3.11       1.28
18                  6.0                  29.0   0.9974  3.38       0.50
19                 17.0                  56.0   0.9969  3.04       1.08

    alcohol  quality
0       9.4        5
1       9.8        5
2       9.8        5
3       9.8        6
4       9.4        5
5       9.4        5
6       9.4        5
7      10.0        7
8       9.5        7
9      10.5        5
10      9.2        5
11     10.5        5
12      9.9        5
13      9.1        5
14      9.2        5
15      9.2        5
16     10.5        7
17      9.3        5
18      9.0        4
19      9.2        6
```

[20]:
```python
df.describe()
```

[20]:
```
       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
```

[6]:
```
df.columns
```

[6]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

[11]:
```
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

0.0.2 Data Pre-processing and Visualisation

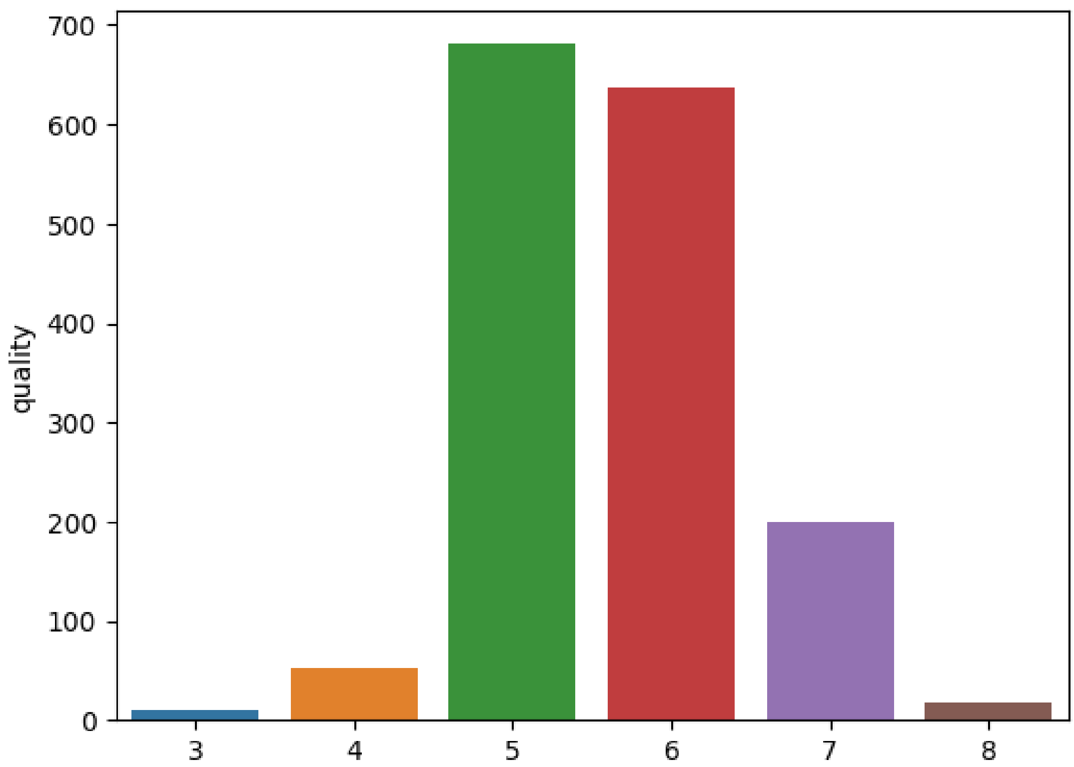
[15]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

[9]:
```
df.isnull().values.any()
```

[9]: False

[26]:
```
sns.barplot(x=df['quality'].value_counts().index, y=df['quality'].
 ↪value_counts())
```

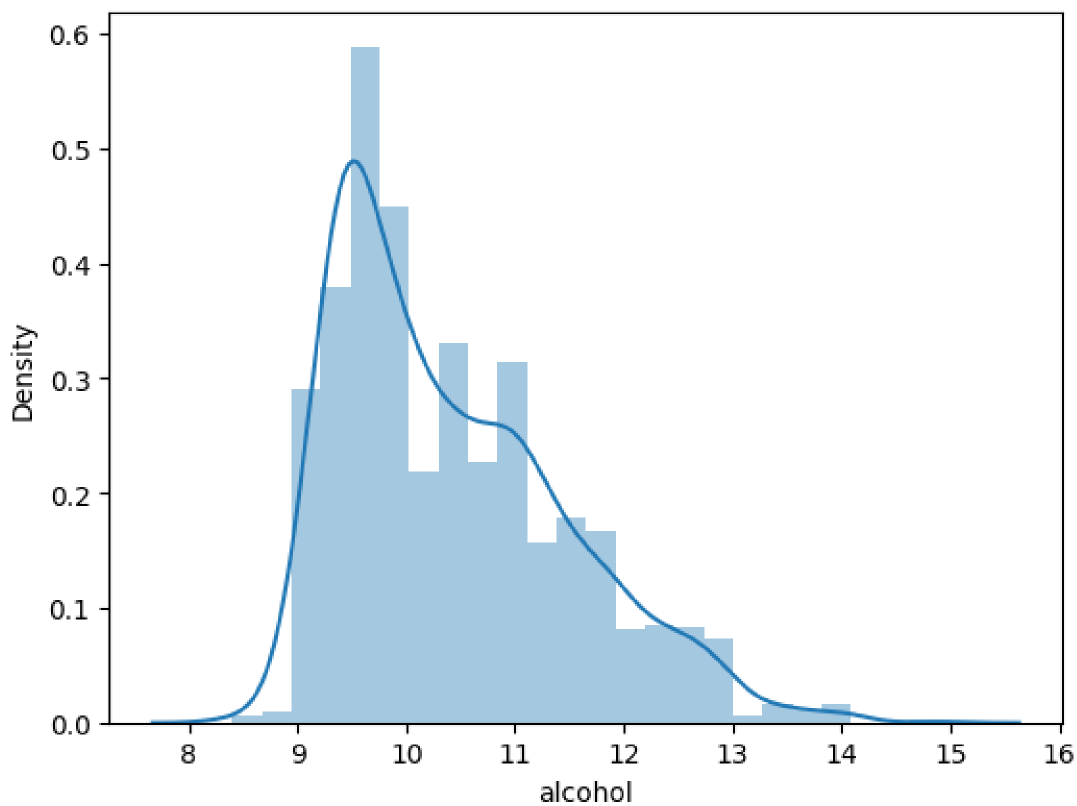
[26]: <Axes: ylabel='quality'>



[28]:
```
sns.distplot(df['alcohol'])
```

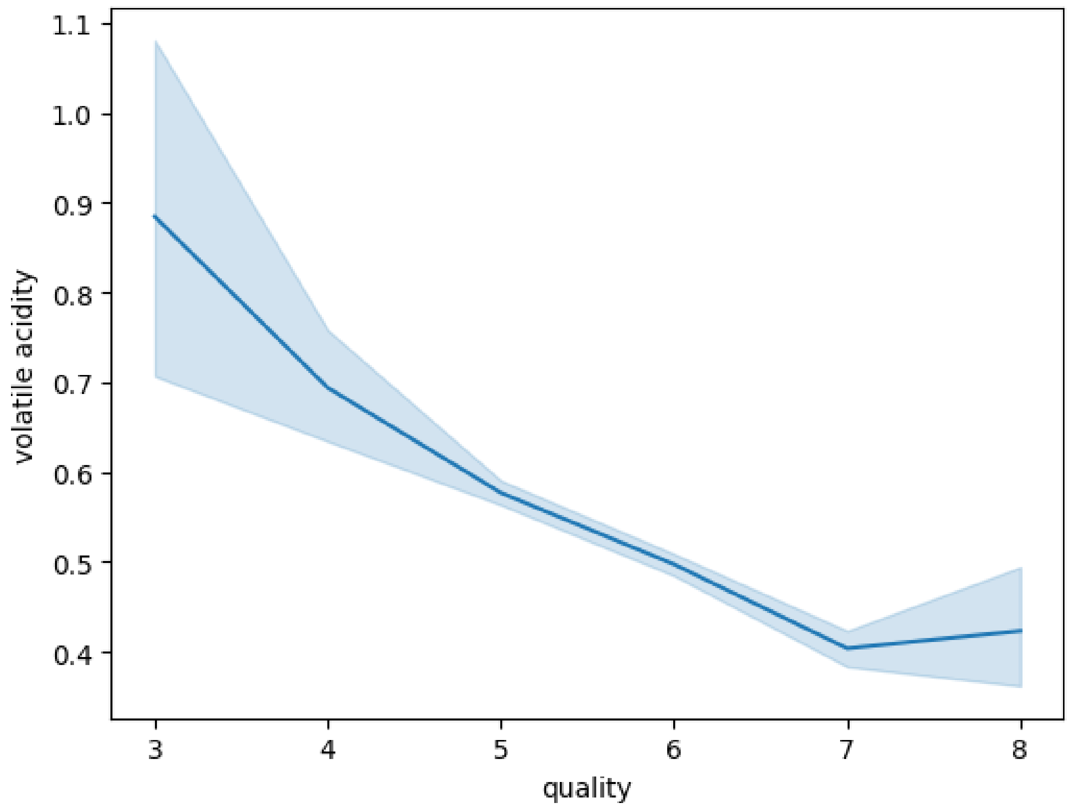
```
<ipython-input-28-570de8ff0310>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['alcohol'])
```

[28]:
```
<Axes: xlabel='alcohol', ylabel='Density'>
```



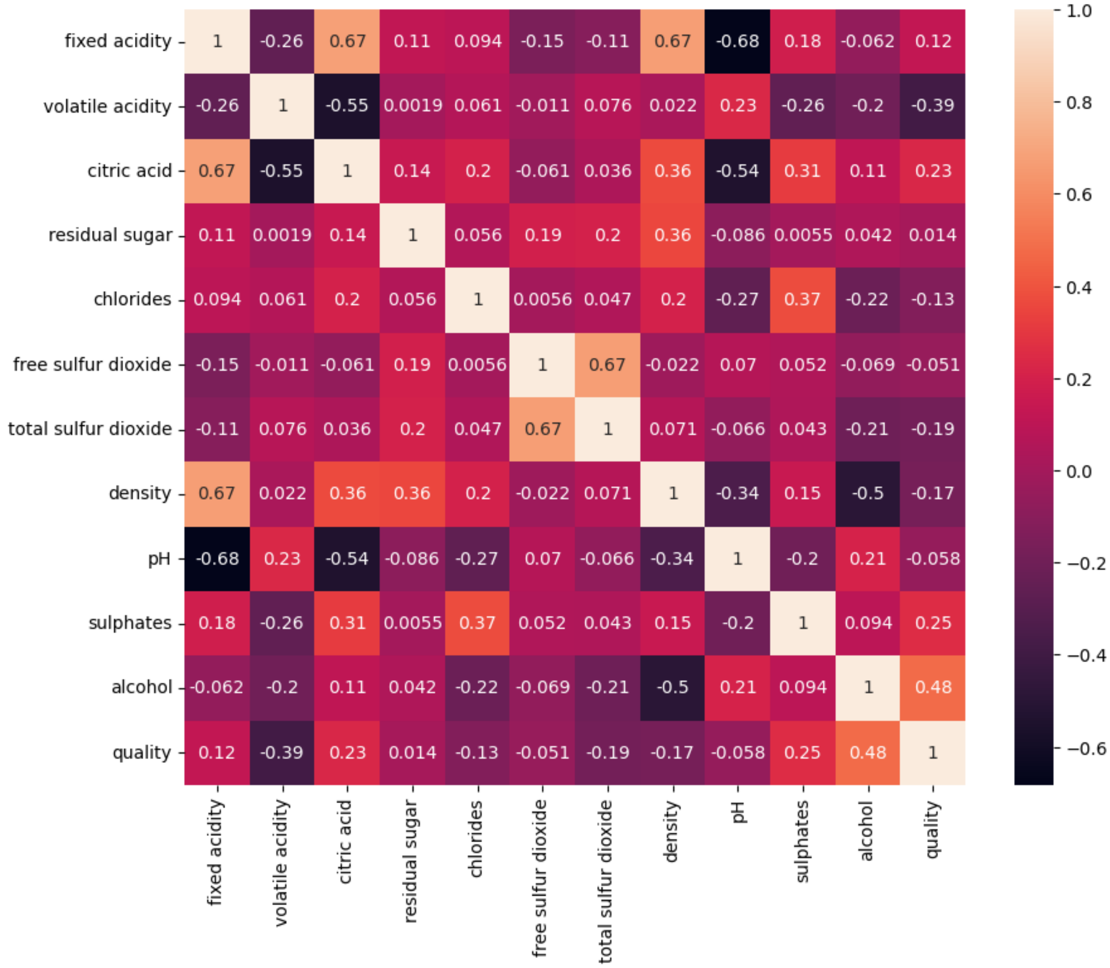
[29]:
```
sns.lineplot(x=df['quality'],y=df['volatile acidity'])
```

[29]:
```
<Axes: xlabel='quality', ylabel='volatile acidity'>
```



[31]:
```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(), annot=True)
```

[31]: <Axes: >



0.0.3 Data Splitting

[34]:
```
y = df['quality']
y.head()
```

[34]:
```
0    5
1    5
2    5
3    6
4    5
Name: quality, dtype: int64
```

[40]:
```
x = df.drop(columns=['quality'], axis=1)
x.head()
```

[40]:

```
   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
0      9.4
1      9.8
2      9.8
3      9.8
4      9.4
```

0.0.4 Scaling The Data

[41]:

```python
from sklearn.preprocessing import MinMaxScaler
```

[43]:

```python
x_scaled = pd.DataFrame(MinMaxScaler().fit_transform(x), columns=x.columns)
x_scaled.head()
```

[43]:

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0       0.247788          0.397260         0.00        0.068493   0.106845
1       0.283186          0.520548         0.00        0.116438   0.143573
2       0.283186          0.438356         0.04        0.095890   0.133556
3       0.584071          0.109589         0.56        0.068493   0.105175
4       0.247788          0.397260         0.00        0.068493   0.106845

   free sulfur dioxide  total sulfur dioxide   density        pH  sulphates  \
0             0.140845              0.098940  0.567548  0.606299   0.137725
1             0.338028              0.215548  0.494126  0.362205   0.209581
2             0.197183              0.169611  0.508811  0.409449   0.191617
3             0.225352              0.190813  0.582232  0.330709   0.149701
4             0.140845              0.098940  0.567548  0.606299   0.137725

    alcohol
0  0.153846
1  0.215385
2  0.215385
3  0.215385
```

```
4   0.153846
```

0.0.5 Train and Test Split

```
[44]: from sklearn.model_selection import train_test_split
```

```
[62]: x_train, x_test,y_train, y_test = train_test_split(x_scaled, y, test_size=0.4,␣
      ↪random_state=0)
```

```
[63]: x_train.shape
```

```
[63]: (959, 11)
```

```
[64]: y_train.shape
```

```
[64]: (959,)
```

```
[65]: x_test.shape
```

```
[65]: (640, 11)
```

```
[66]: y_test.shape
```

```
[66]: (640,)
```

0.0.6 Logistic Model

```
[67]: from sklearn.linear_model import LogisticRegression
      model1 = LogisticRegression()
```

```
[68]: model1.fit(x_train,y_train)
```

```
[68]: LogisticRegression()
```

```
[69]: y_pred1 = model1.predict(x_test)
      y_pred1
```

```
[69]: array([6, 5, 6, 5, 6, 5, 5, 6, 5, 5, 5, 5, 6, 5, 6, 6, 7, 6, 6, 5, 6, 5,
             6, 6, 5, 5, 5, 6, 5, 6, 6, 6, 6, 5, 6, 6, 5, 6, 6, 6, 5, 6, 7, 7,
             6, 5, 5, 6, 6, 6, 5, 5, 6, 6, 6, 5, 5, 5, 7, 5, 5, 6, 6, 6, 5, 6,
             5, 6, 6, 6, 5, 5, 5, 5, 5, 6, 5, 5, 5, 6, 6, 5, 6, 6, 5, 5, 6, 5,
             5, 5, 5, 5, 6, 5, 6, 5, 6, 5, 5, 6, 7, 6, 6, 7, 6, 5, 6, 5, 6, 5,
             6, 5, 6, 5, 6, 6, 6, 7, 6, 6, 5, 6, 5, 5, 6, 6, 5, 5, 6, 6, 5, 5,
             6, 6, 6, 5, 6, 5, 6, 5, 6, 5, 5, 5, 6, 6, 6, 6, 6, 5, 6, 6, 5, 6,
             6, 5, 5, 5, 6, 6, 6, 6, 6, 5, 6, 5, 6, 7, 5, 6, 6, 5, 5, 7, 6, 6,
             6, 7, 6, 5, 5, 7, 5, 6, 7, 5, 5, 6, 5, 6, 6, 6, 5, 5, 5, 5, 5, 5,
```

```
       5, 5, 5, 6, 5, 5, 5, 5, 5, 6, 6, 5, 6, 5, 5, 7, 5, 5, 6, 6, 6, 5,
       5, 6, 6, 6, 5, 6, 7, 6, 5, 5, 5, 6, 5, 6, 6, 6, 6, 7, 7, 6, 5, 5,
       5, 5, 6, 5, 5, 5, 5, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 5, 7, 5, 5,
       5, 5, 5, 5, 6, 7, 5, 6, 6, 6, 6, 6, 6, 5, 7, 6, 5, 7, 6, 6, 6, 5,
       5, 5, 6, 6, 6, 6, 6, 5, 5, 6, 5, 5, 5, 5, 6, 5, 5, 5, 6, 6, 5, 5,
       5, 5, 6, 6, 5, 5, 5, 7, 6, 6, 5, 7, 5, 5, 6, 6, 6, 5, 7, 5, 5, 5,
       7, 5, 5, 6, 5, 6, 6, 5, 5, 5, 5, 5, 5, 6, 6, 5, 5, 5, 5, 6, 6, 6,
       5, 6, 7, 5, 6, 6, 6, 6, 5, 6, 5, 6, 6, 5, 6, 6, 5, 5, 6, 6, 5, 6,
       5, 5, 6, 6, 6, 6, 6, 6, 5, 5, 5, 6, 6, 5, 6, 5, 7, 5, 5, 7, 5, 6,
       5, 6, 6, 5, 5, 5, 5, 5, 6, 5, 6, 6, 5, 6, 5, 5, 5, 6, 6, 5, 6, 5,
       5, 6, 6, 6, 7, 6, 5, 6, 6, 6, 5, 5, 5, 6, 5, 6, 6, 7, 6, 6, 5, 5,
       6, 5, 6, 5, 6, 5, 5, 7, 5, 6, 6, 5, 6, 7, 6, 5, 6, 5, 6, 5, 5, 7,
       5, 5, 5, 5, 6, 5, 5, 5, 6, 5, 5, 5, 6, 5, 5, 5, 6, 5, 6, 7, 5, 6,
       6, 7, 6, 5, 6, 5, 5, 6, 5, 6, 6, 6, 5, 5, 6, 5, 6, 5, 5, 5, 5, 6,
       5, 5, 6, 6, 6, 6, 5, 5, 7, 6, 6, 5, 5, 5, 5, 5, 6, 5, 5, 5, 6, 5,
       6, 6, 6, 5, 6, 7, 5, 6, 6, 6, 6, 6, 6, 5, 5, 6, 6, 6, 7, 6, 7, 5,
       6, 6, 5, 6, 6, 5, 6, 7, 6, 5, 6, 6, 5, 5, 5, 6, 6, 7, 6, 5, 6, 5,
       5, 6, 6, 6, 5, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 5, 6, 7, 6, 5, 6, 6,
       6, 6, 6, 6, 5, 6, 6, 5, 7, 6, 6, 5, 6, 6, 5, 6, 5, 5, 6, 5, 6, 6,
       6, 5, 5, 5, 6, 5, 6, 6, 6, 6, 5, 5, 5, 5, 6, 5, 6, 6, 5, 6, 5, 5,
       6, 6])
```

0.0.7 Evaluation

[70]:
```python
from sklearn.metrics import accuracy_score,classification_report
```

[71]:
```python
acc1 = accuracy_score(y_test,y_pred1)
acc1
```

[71]: 0.615625

[73]:
```python
pd.crosstab(y_test, y_pred1)
```

[73]:

col_0	5	6	7
quality			
3	4	0	0
4	11	12	0
5	208	70	1
6	76	170	22
7	3	41	16
8	0	3	3

[74]:
```python
print(classification_report(y_test,y_pred1))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         4
```

```
           4       0.00      0.00      0.00        23
           5       0.69      0.75      0.72       279
           6       0.57      0.63      0.60       268
           7       0.38      0.27      0.31        60
           8       0.00      0.00      0.00         6

    accuracy                           0.62       640
   macro avg       0.27      0.27      0.27       640
weighted avg       0.58      0.62      0.59       640

/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))
```