In [1]:

```
import pandas as pd
import numpy as np                     # For mathematical calculations
import seaborn as sns                  # For data visualization
import matplotlib.pyplot as plt
import seaborn as sn                   # For plotting graphs
import io
%matplotlib inline
import warnings                        # To ignore any warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
filepath2 = r"C:\Users\91623\OneDrive\Desktop\projects\wine\winequalityN.csv"
df= pd.read_csv(filepath2)
print(df)
```

```
       type  fixed acidity  volatile acidity  citric acid  residual sugar  \
0     white            7.0             0.270         0.36            20.7
1     white            6.3             0.300         0.34             1.6
2     white            8.1             0.280         0.40             6.9
3     white            7.2             0.230         0.32             8.5
4     white            7.2             0.230         0.32             8.5
...     ...            ...               ...          ...             ...
6492    red            6.2             0.600         0.08             2.0
6493    red            5.9             0.550         0.10             2.2
6494    red            6.3             0.510         0.13             2.3
6495    red            5.9             0.645         0.12             2.0
6496    red            6.0             0.310         0.47             3.6

      chlorides  free sulfur dioxide  total sulfur dioxide  density    pH  \
0         0.045                 45.0                 170.0  1.00100  3.00
1         0.049                 14.0                 132.0  0.99400  3.30
2         0.050                 30.0                  97.0  0.99510  3.26
3         0.058                 47.0                 186.0  0.99560  3.19
4         0.058                 47.0                 186.0  0.99560  3.19
...         ...                  ...                   ...      ...   ...
6492      0.090                 32.0                  44.0  0.99490  3.45
6493      0.062                 39.0                  51.0  0.99512  3.52
6494      0.076                 29.0                  40.0  0.99574  3.42
6495      0.075                 32.0                  44.0  0.99547  3.57
6496      0.067                 18.0                  42.0  0.99549  3.39

      sulphates  alcohol  quality
0          0.45      8.8        6
1          0.49      9.5        6
2          0.44     10.1        6
3          0.40      9.9        6
4          0.40      9.9        6
...         ...      ...      ...
6492       0.58     10.5        5
6493        NaN     11.2        6
6494       0.75     11.0        6
6495       0.71     10.2        5
6496       0.66     11.0        6

[6497 rows x 13 columns]
```

In [3]:

```
df.head()
```

Out[3]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

In [4]:

```
df= df.dropna()
```

In [5]:

```
df.describe()
```

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	
count	6463.000000	6463.000000	6463.000000	6463.000000	6463.000000	6463.000000	6463.000000	6463.000000	6463.000000	6463.000000	6463
mean	7.217755	0.339589	0.318758	5.443958	0.056056	30.516865	115.694492	0.994698	3.218332	0.531150	10
std	1.297913	0.164639	0.145252	4.756852	0.035076	17.758815	56.526736	0.003001	0.160650	0.148913	
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000	8
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992330	3.110000	0.430000	9
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000	10
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.997000	3.320000	0.600000	11
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000	14

In [6]:

```
unique = [feature for feature in df.columns if len(df[feature].unique())>0 and len(df[feature].unique())<100]
for feature in unique:
    print("{} has {} unique values : {} {}".format(feature,len(df[feature].unique()),df[feature].unique(),"\n")
```

```
type has 2 unique values : ['white' 'red']

citric acid has 89 unique values : [0.36 0.34 0.4  0.32 0.16 0.43 0.41 0.37 0.62 0.38 0.04 0.42 0.14 0.48
 0.35 0.39 0.2  0.23 0.26 0.27 0.31 0.25 0.29 0.33 0.15 0.24 0.07 0.03
 0.13 0.28 0.46 0.3  0.61 0.63 0.66 0.54 0.5  0.   0.47 0.56 0.22 0.67
 0.18 0.45 0.44 0.88 0.21 0.08 0.59 0.49 0.58 0.7  0.6  0.51 0.1  0.19
 0.12 0.09 0.53 0.02 0.65 0.17 0.71 0.06 0.68 0.72 0.69 1.66 0.57 0.05
 0.52 1.   0.01 0.74 0.81 0.55 0.64 0.73 0.99 0.78 0.79 0.82 0.8  1.23
 0.86 0.11 0.91 0.76 0.75]

quality has 7 unique values : [6 5 7 8 4 3 9]
```

In [7]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [8]:

```
df1=df
```

In [9]:

```
list1=['type']
for i in list1:
    df1[i]=le.fit_transform(df1[i])

df1.head()
```

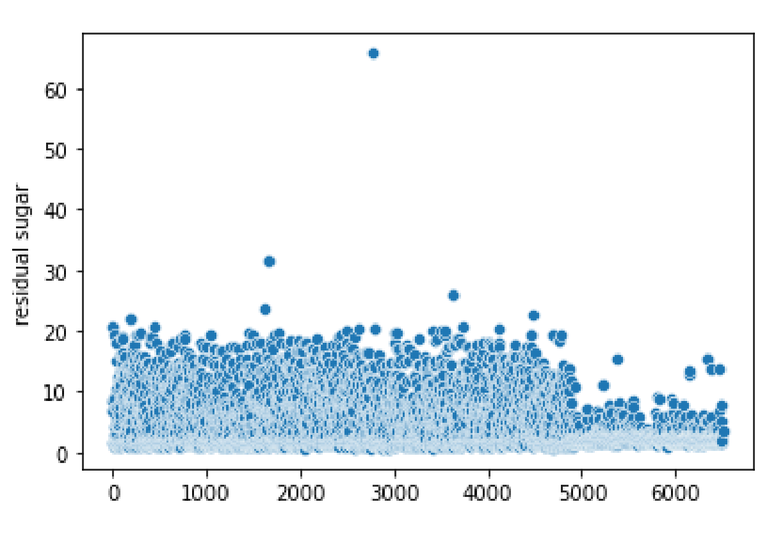
Out[9]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	1	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	1	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	1	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	1	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

In [10]:

```
sns.scatterplot(data =df1['residual sugar'])
```

Out[10]: <AxesSubplot:ylabel='residual sugar'>



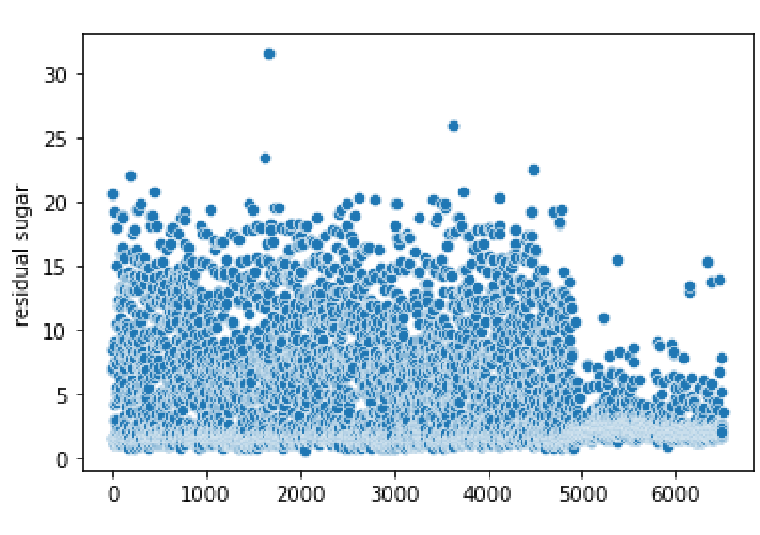
In [11]:

```
df1 = df1.drop(df1[df1['residual sugar']>40].index)
```

In [12]:

```
sns.scatterplot(data =df1['residual sugar'])
```

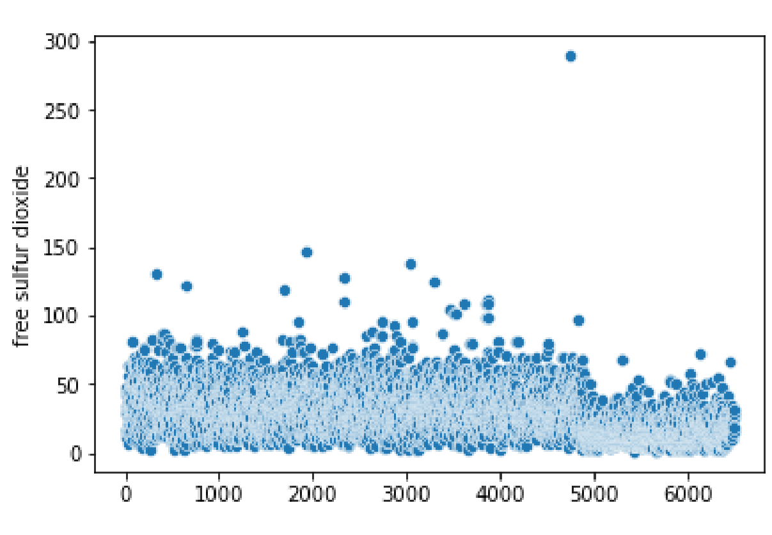
Out[12]: <AxesSubplot:ylabel='residual sugar'>



In [13]:

```
sns.scatterplot(data =df1['free sulfur dioxide'])
```

Out[13]: <AxesSubplot:ylabel='free sulfur dioxide'>



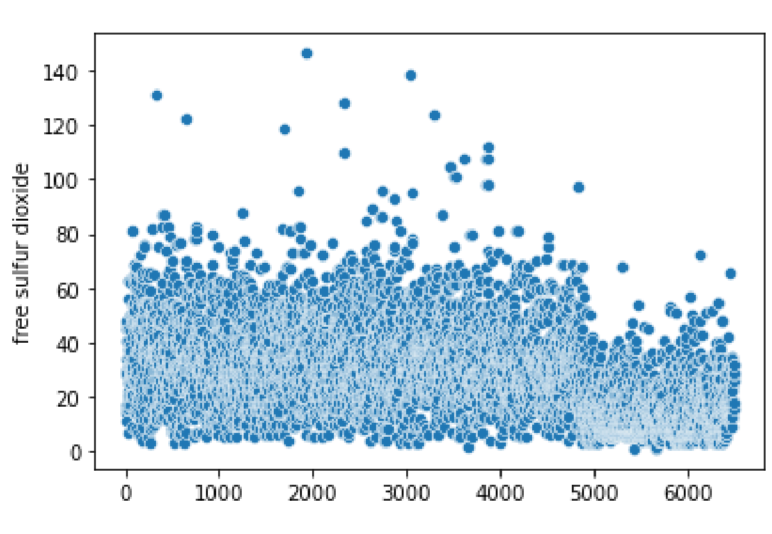
In [14]:

```
df1 = df1.drop(df1[df1['free sulfur dioxide']>200].index)
```

In [15]:

```
sns.scatterplot(data =df1['free sulfur dioxide'])
```

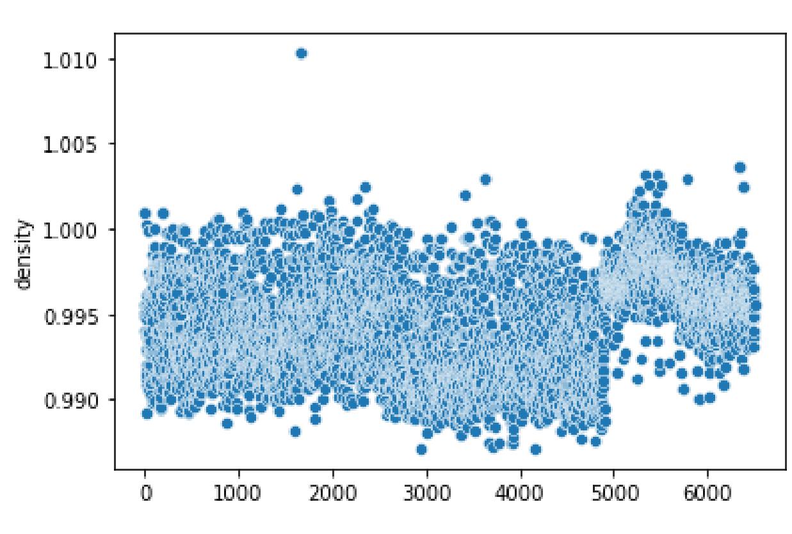
Out[15]: <AxesSubplot:ylabel='free sulfur dioxide'>



In [16]:

```
sns.scatterplot(data =df1['density'])
```

Out[16]: <AxesSubplot:ylabel='density'>



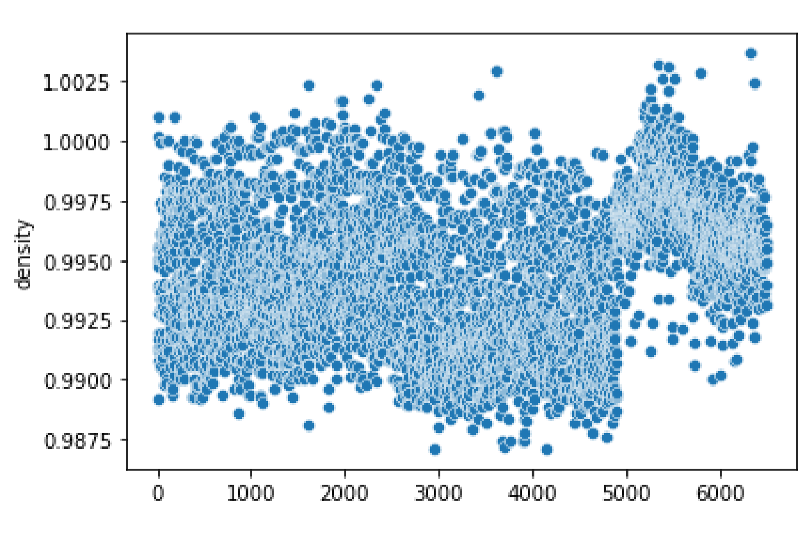
In [17]:

```
df1 = df1.drop(df1[df1['density']>1.005].index)
```

In [18]:

```
sns.scatterplot(data =df1['density'])
```

Out[18]: <AxesSubplot:ylabel='density'>



In [19]:

```
y=df1['quality']
```

In [20]:

```
y.shape
```

Out[20]:

```
(6459,)
```

In [21]:

```
df1.head()
```

Out[21]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	1	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	1	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	1	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	1	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

In [22]:

```
x=df1[["type","fixed acidity","volatile acidity","citric acid","residual sugar","chlorides","free sulfur dioxic
```

In [23]:

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
```

In [24]:

```
x=StandardScaler().fit(x).transform(x)
x
```

Out[24]:

```
array([[ 0.57216572, -0.1676547 , -0.42256361, ..., -1.35885052,
        -0.54501192, -1.41931958],
       [ 0.57216572, -0.70691924, -0.24018217, ...,  0.50862326,
        -0.2763842 , -0.83253343],
       [ 0.57216572,  0.67976101, -0.3617698 , ...,  0.25962675,
        -0.61216885, -0.32957387],
       ...,
       [-1.74774539, -0.70691924,  1.03648798, ...,  1.25561277,
         1.469696  ,  0.42486546],
       [-1.74774539, -1.01507041,  1.8572045 , ...,  2.18934965,
         1.20106828, -0.24574728],
       [-1.74774539, -0.93803262, -0.17938835, ...,  1.06886539,
         0.86528362,  0.42486546]])
```

In [25]:

```
from sklearn.model_selection import train_test_split
```

In [26]:

```
x_train,x_test,y_train,y_test =train_test_split(x,y, test_size=0.3,random_state=2529)
```

In [27]:

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[27]:

```
((4521, 12), (1938, 12), (4521,), (1938,))
```

In [28]:

```
from sklearn.linear_model import LogisticRegression
```

In [29]:

```
lr=LogisticRegression()
```

In [30]:

```
lr.fit(x_train,y_train)
```

Out[30]:

```
LogisticRegression()
```

In [31]:

```
y_pred = lr.predict(x_test)
```

In [32]:

```
from sklearn.metrics import confusion_matrix ,classification_report
```

In [33]:

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

```
[[  0   1   4   5   1   0   0]
 [  0   4  47  16   0   0   0]
 [  0   2 375 263   3   0   0]
 [  0   0 212 563  51   1   0]
 [  0   0  16 243  75   0   0]
 [  0   0   1  37  17   0   0]
 [  0   0   0   0   1   0   0]]
```

In [34]:

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

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00        11
           4       0.57      0.06      0.11        67
           5       0.57      0.58      0.58       643
           6       0.50      0.68      0.58       827
           7       0.51      0.22      0.31       334
           8       0.00      0.00      0.00        55
           9       0.00      0.00      0.00         1

    accuracy                           0.52      1938
   macro avg       0.31      0.22      0.22      1938
weighted avg       0.51      0.52      0.49      1938
```

In [35]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [36]:

```
rfc=RandomForestClassifier()
rfc
```

Out[36]:

```
RandomForestClassifier()
```

In [37]:

```
rfc.fit(x_train,y_train)
```

Out[37]:

```
RandomForestClassifier()
```

In [38]:

```
rfc_pred=rfc.predict(x_test)
```

In [47]:

```
print(confusion_matrix(y_test,rfc_pred))
```

```
[[  0   0   6   5   0   0   0]
 [  0   6  45  16   0   0   0]
 [  0   3 464 171   5   0   0]
 [  0   1 131 643  52   0   0]
 [  0   0   6 146 181   1   0]
 [  0   0   0  26  13  16   0]
 [  0   0   0   1   0   0   0]]
```

In [46]:

```
print(classification_report(y_test,rfc_pred))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00        11
           4       0.60      0.09      0.16        67
           5       0.71      0.72      0.72       643
           6       0.64      0.78      0.70       827
           7       0.72      0.54      0.62       334
           8       0.94      0.29      0.44        55
           9       0.00      0.00      0.00         1

    accuracy                           0.68      1938
   macro avg       0.52      0.35      0.38      1938
weighted avg       0.68      0.68      0.66      1938
```

In [52]:

```
from sklearn import tree
```

In [53]:

```
clf = tree.DecisionTreeClassifier()
```

In [54]:

```
clf.fit(x_train,y_train)
```

Out[54]:

```
DecisionTreeClassifier()
```

In [55]:

```
clf_pred=clf.predict(x_test)
```

In [58]:

```
print(confusion_matrix(y_test,clf_pred))
```

```
[[  0   1   4   4   2   0   0]
 [  2  10  36  15   3   1   0]
 [  5  13 403 191  30   1   0]
 [  1  16 170 526 102  11   1]
 [  0   5  17 112 184  16   0]
 [  0   0   4  17  16  18   0]
 [  0   0   0   0   1   0   0]]
```

In [59]:

```
print(classification_report(y_test,clf_pred))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00        11
           4       0.22      0.15      0.18        67
           5       0.64      0.63      0.63       643
           6       0.61      0.64      0.62       827
           7       0.54      0.55      0.55       334
           8       0.38      0.33      0.35        55
           9       0.00      0.00      0.00         1

    accuracy                           0.59      1938
   macro avg       0.34      0.33      0.33      1938
weighted avg       0.58      0.59      0.59      1938
```

In [ ]:

In [ ]:

In [41]:

```
from sklearn.svm import SVC
```

In [42]:

```
svC=SVC()
svC.fit(x_train,y_train)
```

Out[42]:

```
SVC()
```

In [43]:

```
svC_pred=svC.predict(x_test)
```

In [49]:

```
print(confusion_matrix(y_test,svC_pred))
```

```
[[  0   0   5   6   0   0   0]
 [  0   1  51  15   0   0   0]
 [  0   0 405 236   2   0   0]
 [  0   0 178 614  35   0   0]
 [  0   0   8 256  70   0   0]
 [  0   0   1  40  14   0   0]
 [  0   0   0   0   1   0   0]]
```

In [48]:

```
print(classification_report(y_test,svC_pred))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00        11
           4       1.00      0.01      0.03        67
           5       0.62      0.63      0.63       643
           6       0.53      0.74      0.62       827
           7       0.57      0.21      0.31       334
           8       0.00      0.00      0.00        55
           9       0.00      0.00      0.00         1

    accuracy                           0.56      1938
   macro avg       0.39      0.23      0.23      1938
weighted avg       0.57      0.56      0.52      1938
```

In [ ]:

In [60]:

```
from sklearn.naive_bayes import GaussianNB
```

In [61]:

```
gnb = GaussianNB()
```

In [62]:

```
gnb.fit(x_train,y_train)
```

Out[62]:

```
GaussianNB()
```

In [63]:

```
gnb_pred=gnb.predict(x_test)
```

In [64]:

```
print(confusion_matrix(y_test,gnb_pred))
```

```
[[  5   1   1   2   0   0   2]
 [  3   7  31  17   1   0   8]
 [ 11  22 328 201  23   0  58]
 [ 13  23 240 280  68   0 203]
 [  9   6  30  89  38   1 161]
 [  0   1   3   7  10   2  32]
 [  0   0   0   0   0   0   1]]
```

In [65]:

```
print(classification_report(y_test,gnb_pred))
```

```
              precision    recall  f1-score   support

           3       0.12      0.45      0.19        11
           4       0.12      0.10      0.11        67
           5       0.52      0.51      0.51       643
           6       0.47      0.34      0.39       827
           7       0.27      0.11      0.16       334
           8       0.67      0.04      0.07        55
           9       0.00      1.00      0.00         1

    accuracy                           0.34      1938
   macro avg       0.31      0.37      0.21      1938
weighted avg       0.44      0.34      0.37      1938
```

In [ ]:

In [ ]:

In [70]:

In [ ]: