Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
import re
import nltk
nltk.download('stopwords')
from nltk.util import pr
stemmer = nltk.SnowballStemmer("english")
from nltk.corpus import stopwords
import string
stopword = set(stopwords.words("english"))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\mshiv\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Loading Dataset

In [3]:
```
df = pd.read_csv(r"E:\Data Science Projects\Project Files\Capestone 1 - Hate Spee
```

In [4]:
```
df.head()
```

Out[4]:

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	4	6	0	6	0	1	!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...

In [5]:
```
df.tail()
```

Out[5]:

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
24778	25291	3	0	2	1	1	you's a muthaf***in lie “@LifeAsKing: @2...
24779	25292	3	0	1	2	2	you've gone and broke the wrong heart baby, an...
24780	25294	3	0	3	0	1	young buck wanna eat!!.. dat nigguh like I ain...
24781	25295	6	0	6	0	1	youu got wild bitches tellin you lies
24782	25296	3	0	0	3	2	~~Ruffled | Ntac Eileen Dahlia - Beautiful col...

Data Preparation

In [6]:
```
df['labels'] = df['class'].map({0:"Hate Speech Detected",1:"Offensive language de
```

In [7]:
```
df.head()
```

Out[7]:

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet	labels
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...	NO hate and offensive speech
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...	Offensive language detected
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...	Offensive language detected
3	3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...	Offensive language detected
4	4	6	0	6	0	1	!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...	Offensive language detected

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24783 entries, 0 to 24782
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Unnamed: 0          24783 non-null  int64 
 1   count               24783 non-null  int64 
 2   hate_speech         24783 non-null  int64 
 3   offensive_language  24783 non-null  int64 
 4   neither             24783 non-null  int64 
 5   class               24783 non-null  int64 
 6   tweet               24783 non-null  object
 7   labels              24783 non-null  object
dtypes: int64(6), object(2)
memory usage: 1.5+ MB
```

In [9]:
```
num_data=[x for x in df.columns if df[x].dtypes!='object']
```

In [10]:
```
cat_data=[x for x in df.columns if df[x].dtypes=='object']
```

In [11]:
```
num_data
```

Out[11]:
```
['Unnamed: 0',
 'count',
 'hate_speech',
 'offensive_language',
 'neither',
 'class']
```

In [12]:
```
num_data.remove('Unnamed: 0')
```

In [13]:
```
num_data
```

Out[13]:
```
['count', 'hate_speech', 'offensive_language', 'neither', 'class']
```

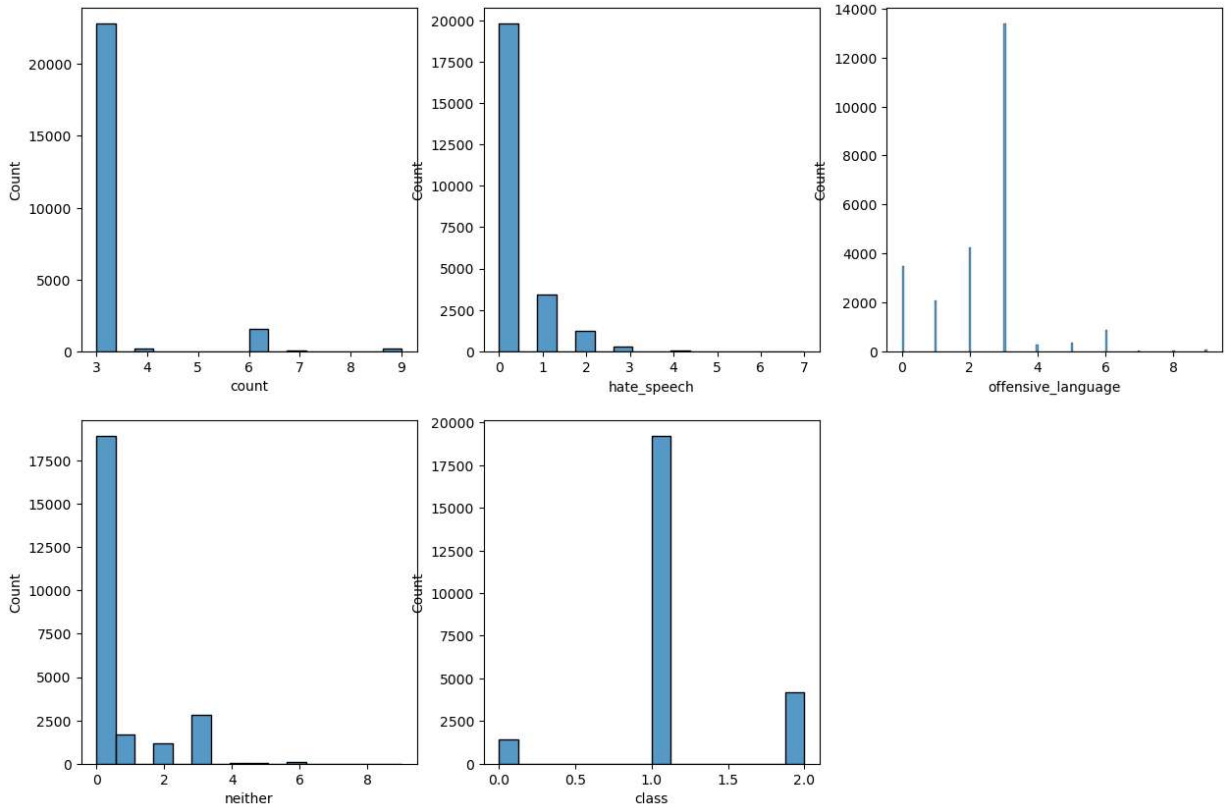
In [14]:
```
cat_data
```

Out[14]:
```
['tweet', 'labels']
```

Visualization

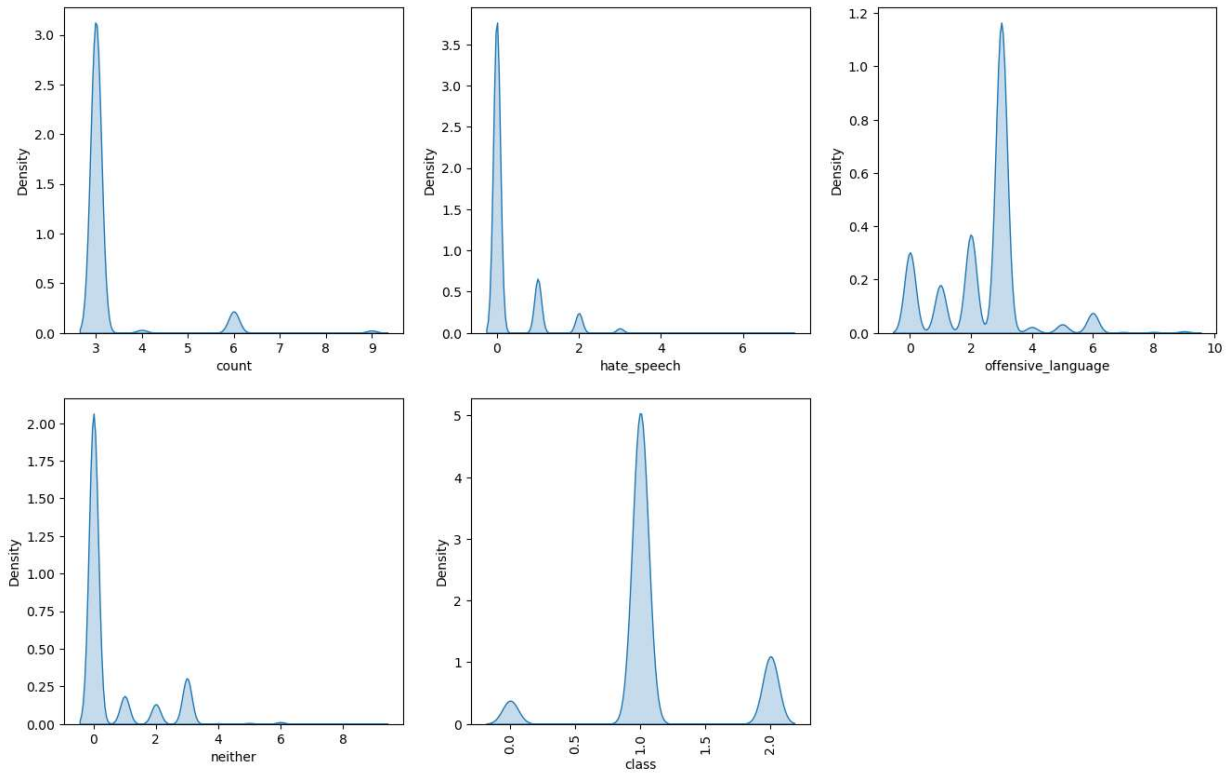
In [15]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [16]:
```
plt.figure(figsize=(15,10))

for i , col in enumerate(num_data):
    plt.subplot(2,3,i+1)
    sns.histplot(data=df,x=col)
plt.show()
```

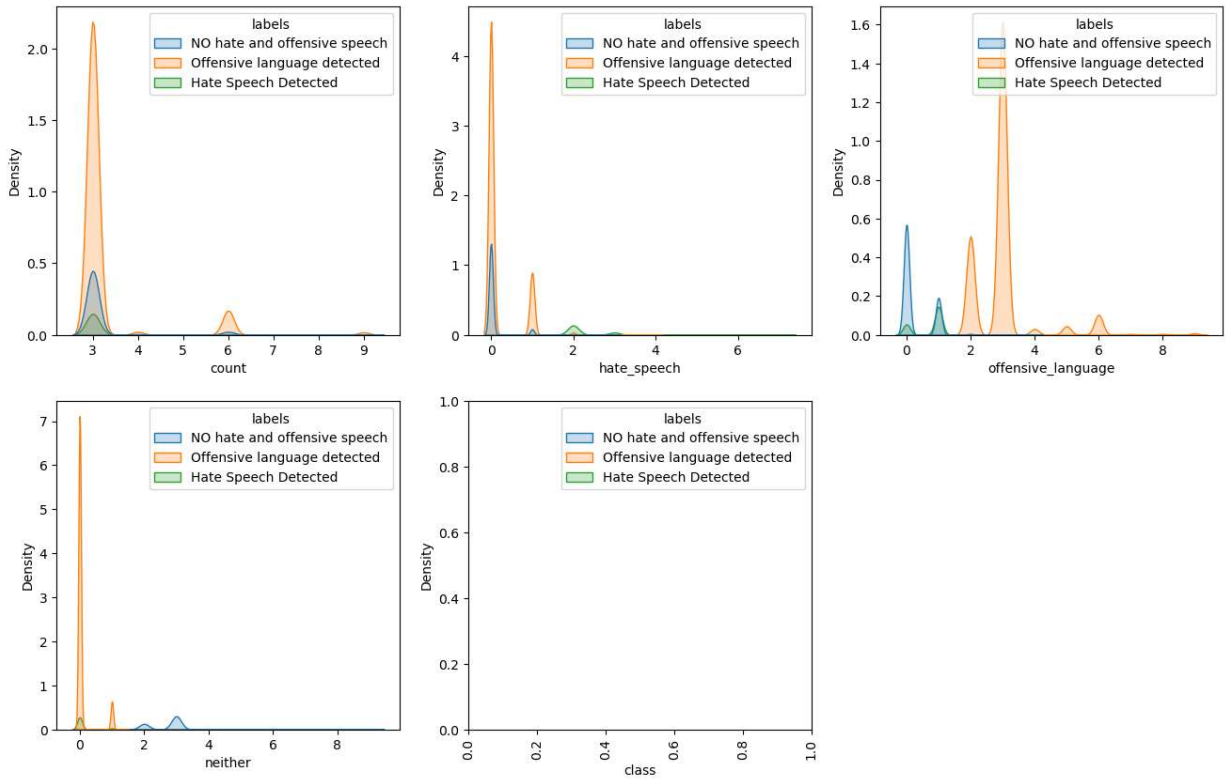


KDE Plot

In [17]:
```
plt.figure(figsize=(16,10))

for i , col in enumerate(num_data):
    plt.subplot(2,3,i+1)
    sns.kdeplot(data=df,x=col,fill=True)
plt.xticks(rotation = 90)
plt.show()
```

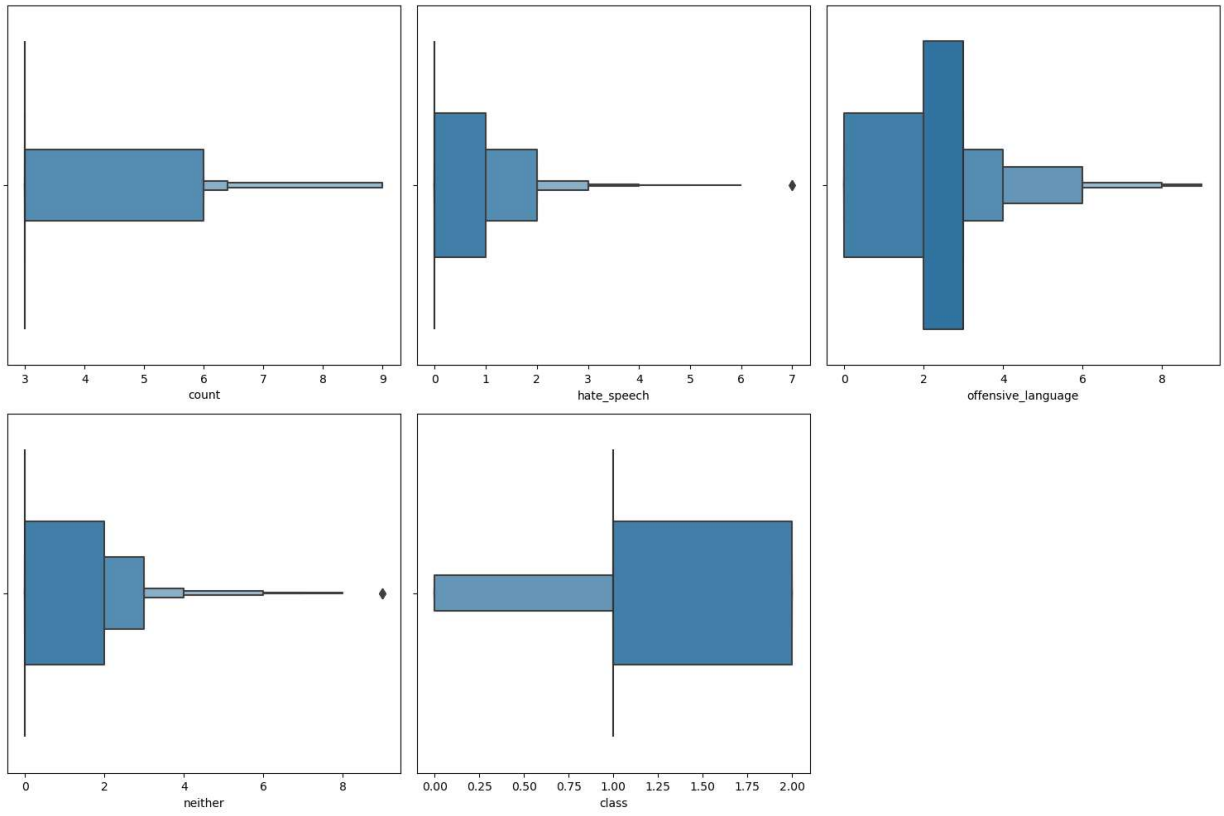


In [84]:
```
plt.figure(figsize=(16,10))

for i , col in enumerate(num_data):
    plt.subplot(2,3,i+1)
    sns.kdeplot(data=df,x=col,hue=df['labels'],fill=True)
plt.xticks(rotation = 90)
plt.show()
```



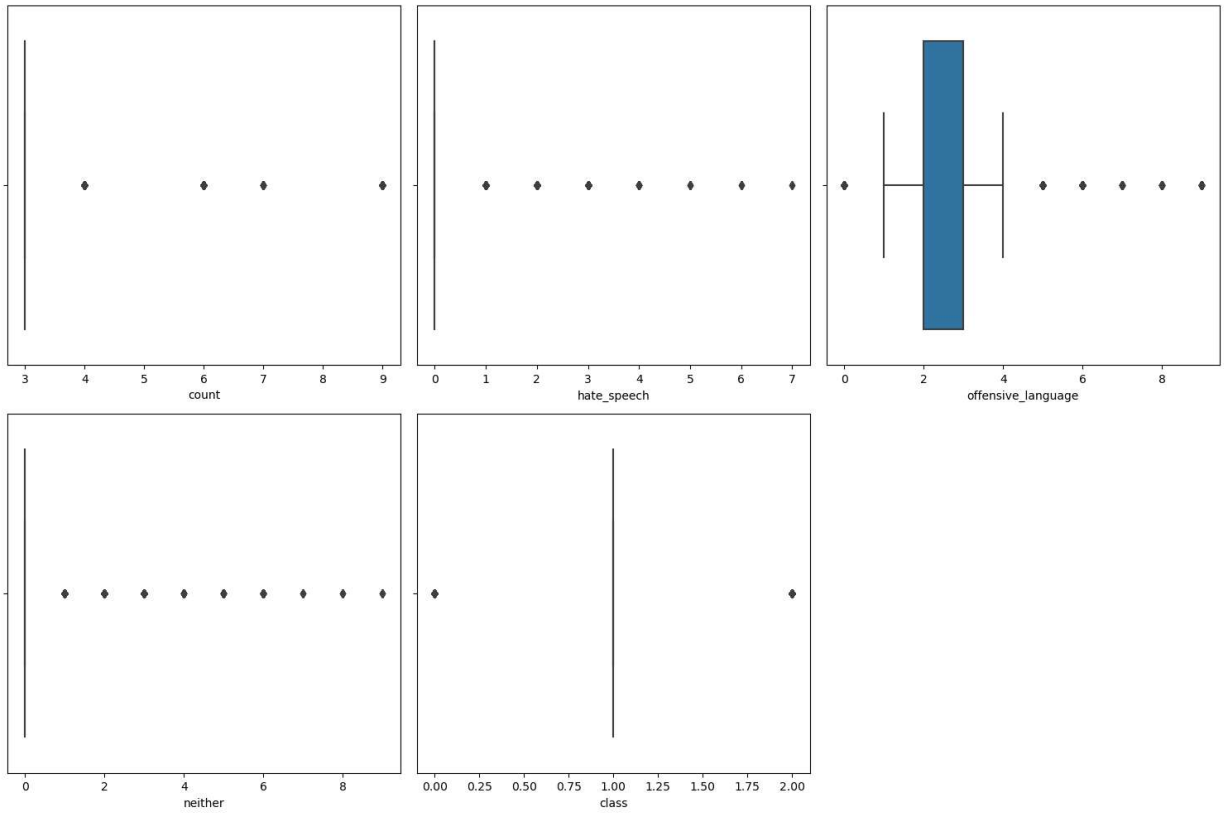
Boxen Plot

In [18]:
```
plt.figure(figsize=(15,10))


for i , col in enumerate(num_data):
    plt.subplot(2,3,i+1)
    sns.boxenplot(data=df,x=col)
plt.tight_layout()
plt.show()
```



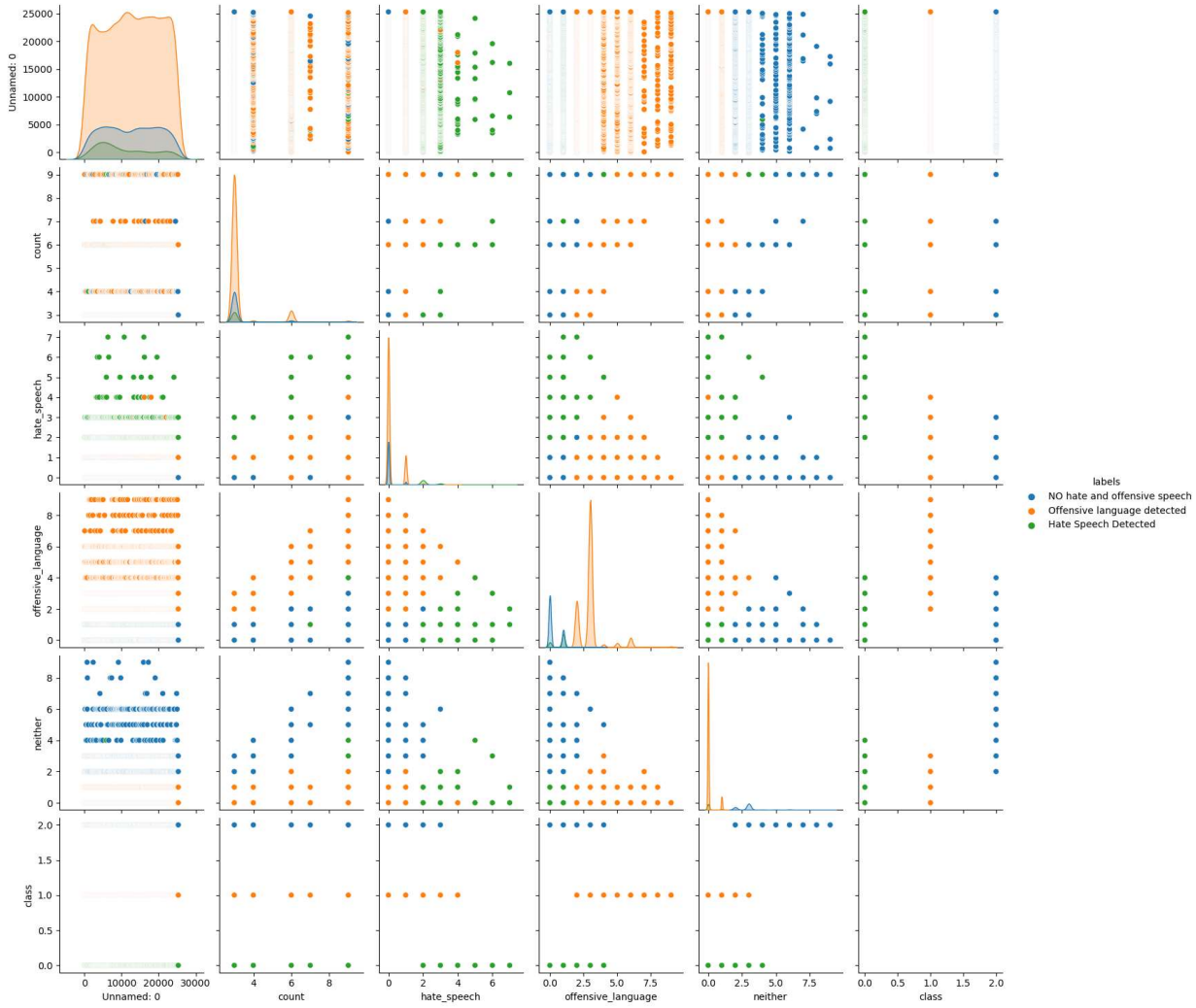
Box Plot

In [19]:
```
plt.figure(figsize=(15,10))


for i , col in enumerate(num_data):
    plt.subplot(2,3,i+1)
    sns.boxplot(data=df,x=col)
plt.tight_layout()
plt.show()
```

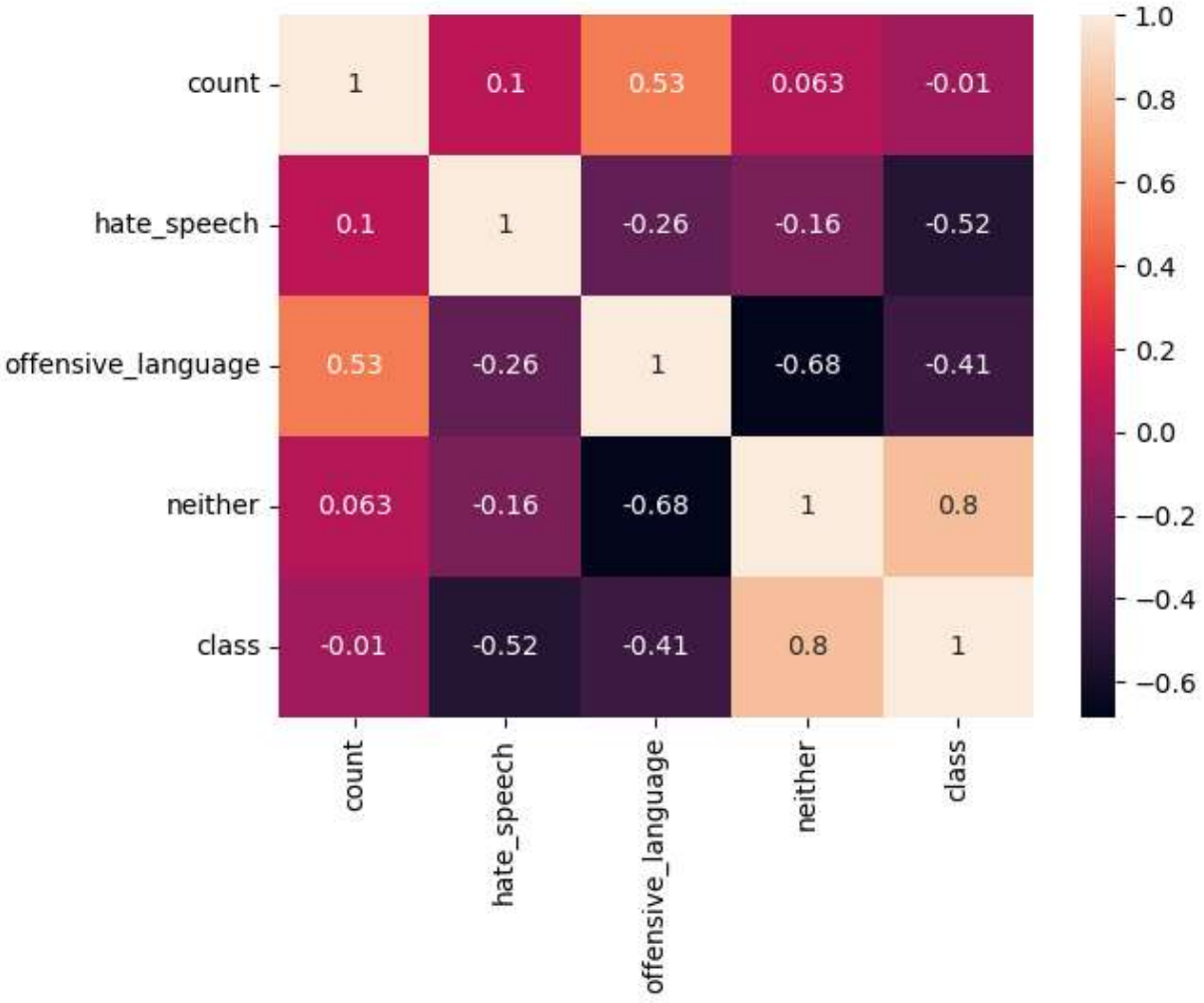


Pair Plot

In [20]:
```
sns.pairplot(df,hue='labels')
plt.show()
```

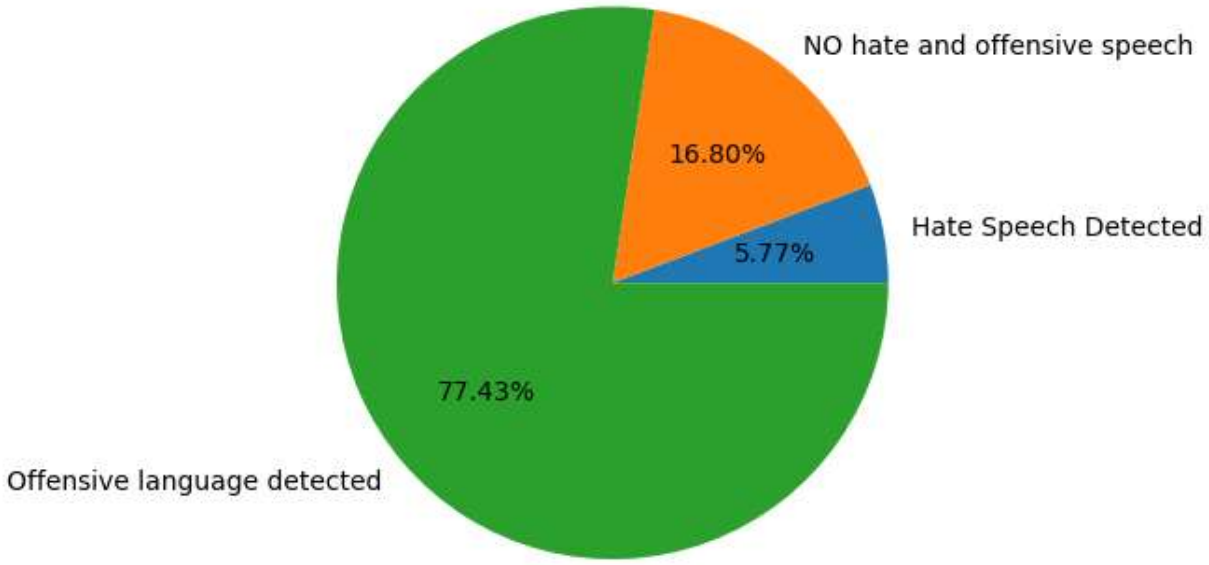


In [21]:
```
sns.heatmap(df[num_data].corr(),annot=True)
plt.show()
```



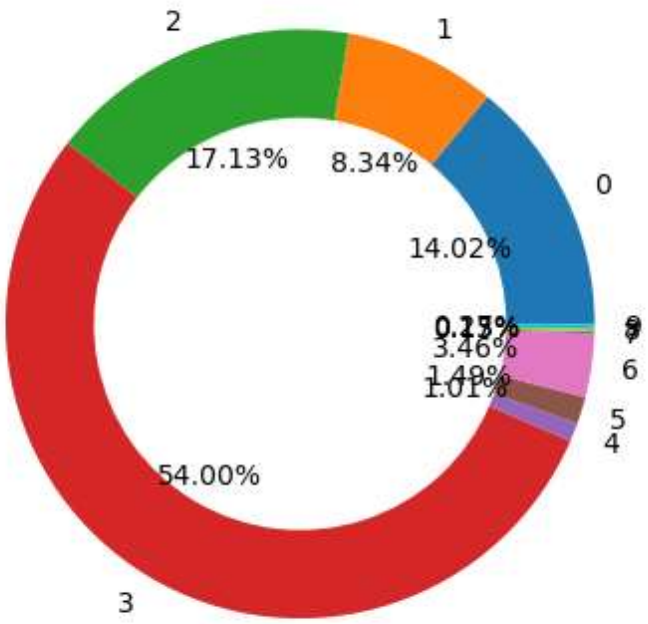
Pie Chart

In [29]:
```
h=df.groupby('labels')['labels'].count()
plt.pie(h, labels=h.index, autopct='%.2f%%')
plt.show()
```



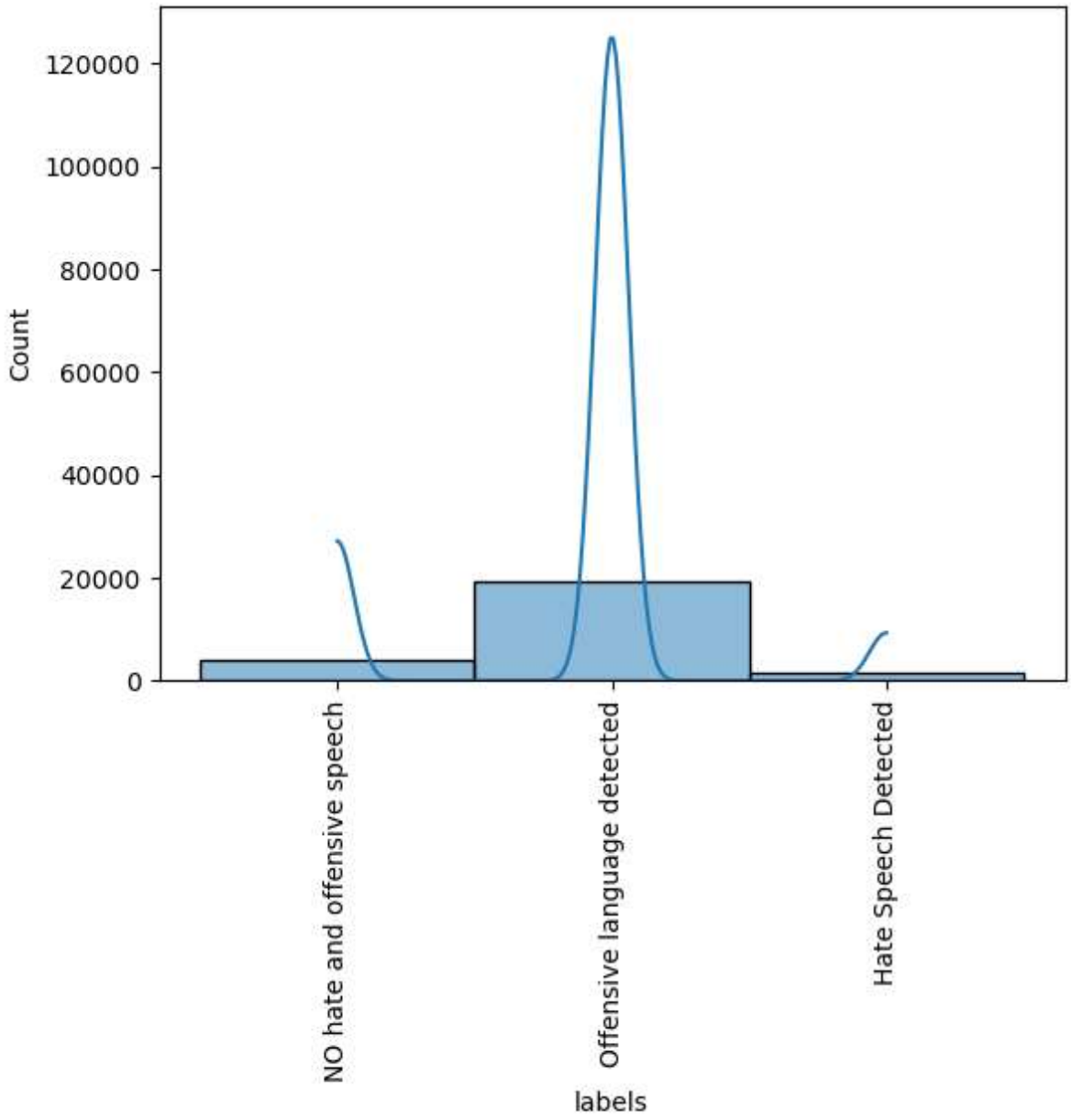
Donut Chart

In [38]:
```
i=df.groupby('offensive_language')['offensive_language'].count()
plt.pie(i, labels=i.index, autopct='%.2f%%', wedgeprops=dict(width=0.3))
plt.show()
```

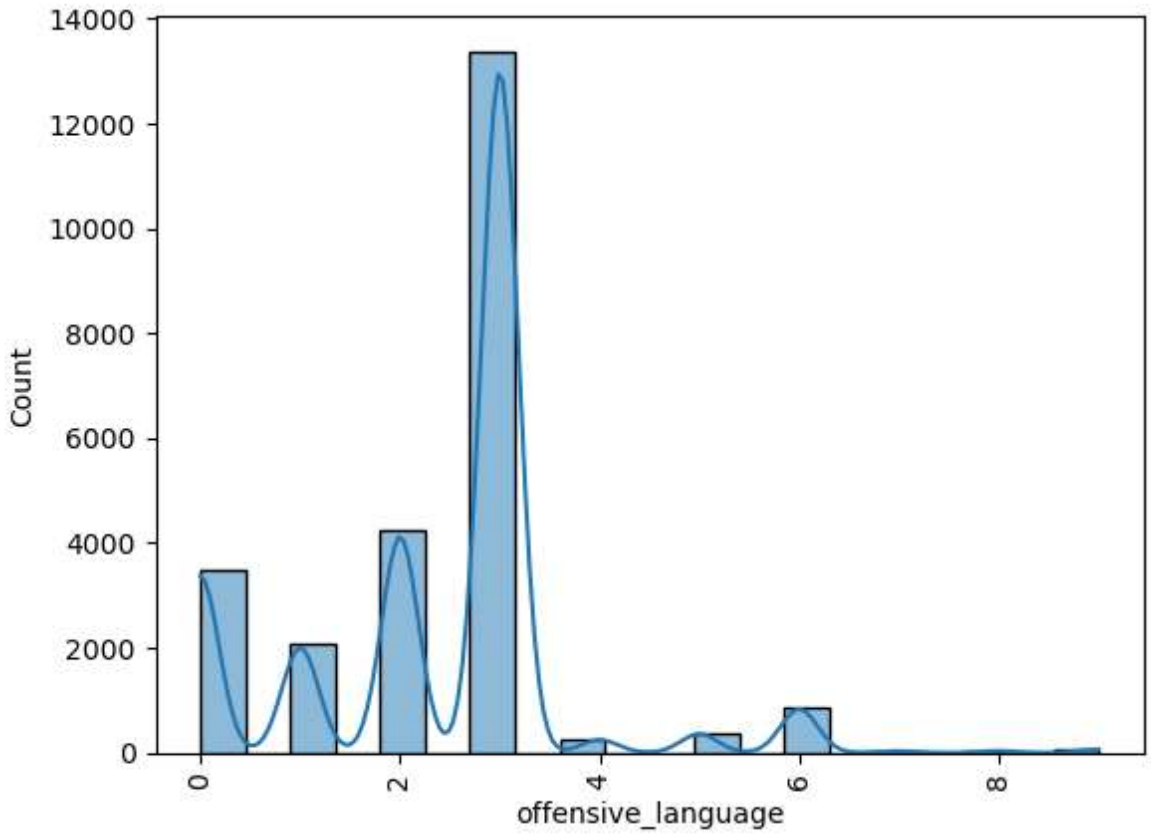


Hist plot

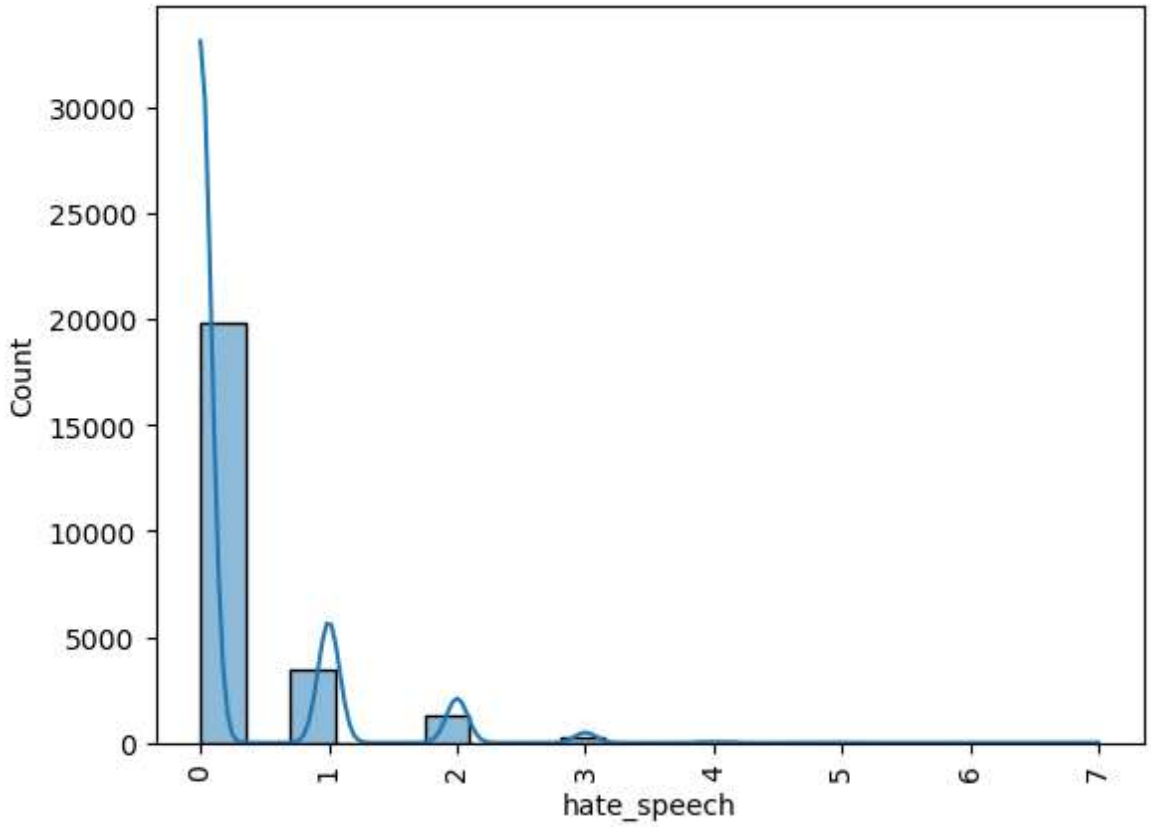
In [40]:
```
sns.histplot(df['labels'], bins=20, kde=True)
plt.xticks(rotation=90)
plt.show()
```



In [42]:
```
sns.histplot(df['offensive_language'], bins=20, kde=True)
plt.xticks(rotation=90)
plt.show()
```



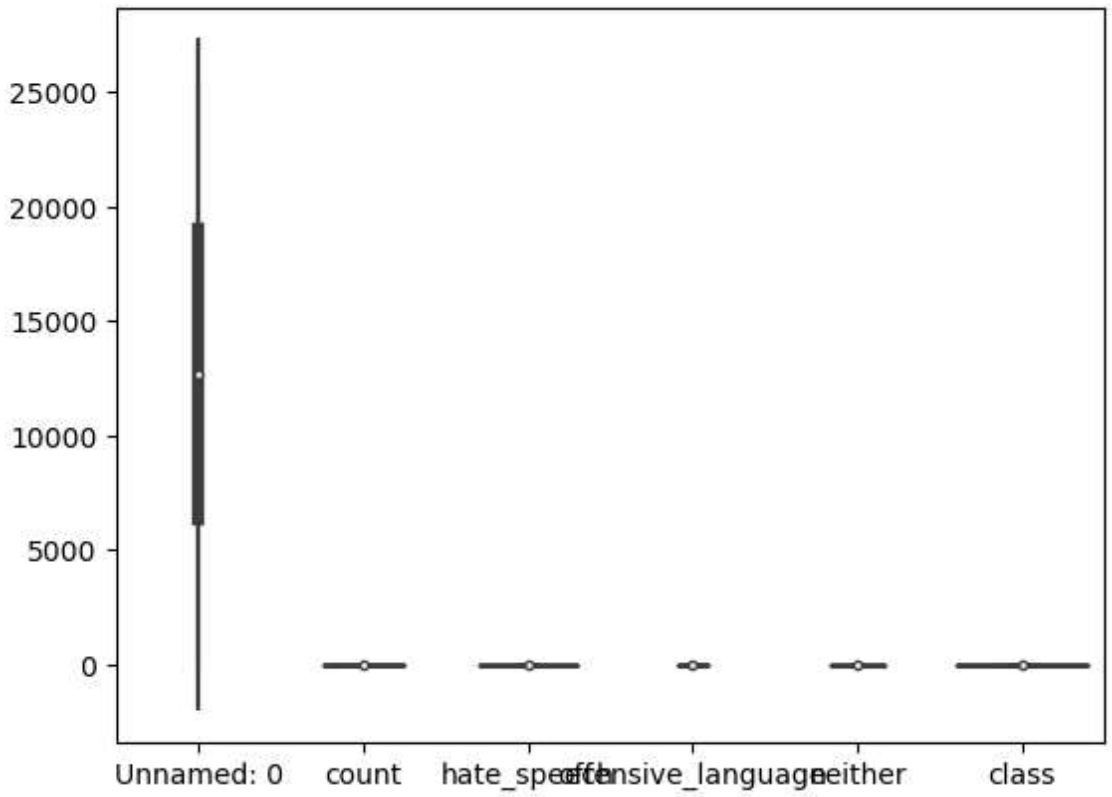
In [43]:
```
sns.histplot(df['hate_speech'], bins=20, kde=True)
plt.xticks(rotation=90)
plt.show()
```



Violinplot

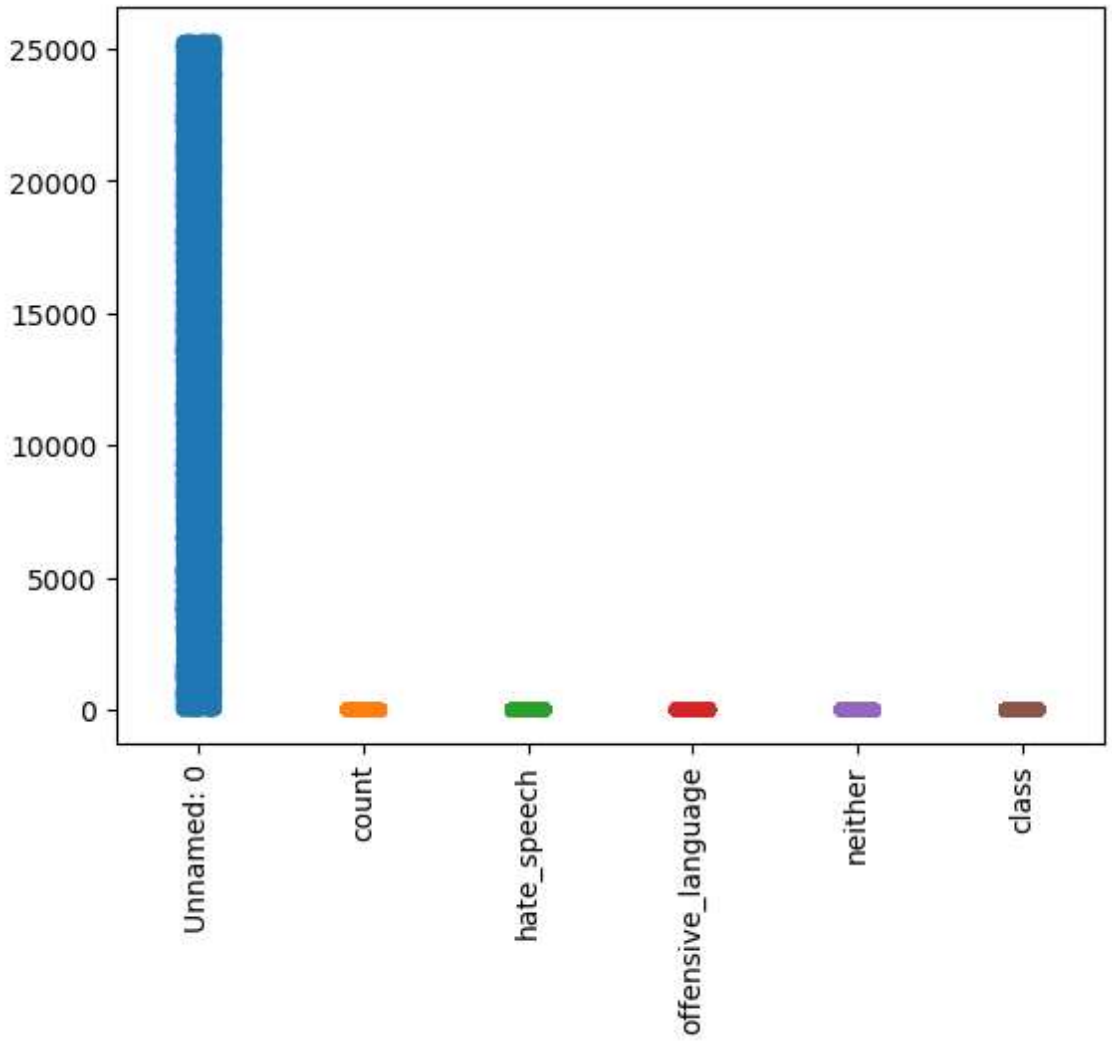
In [50]:
```
sns.violinplot(df)
```

Out[50]: <Axes: >



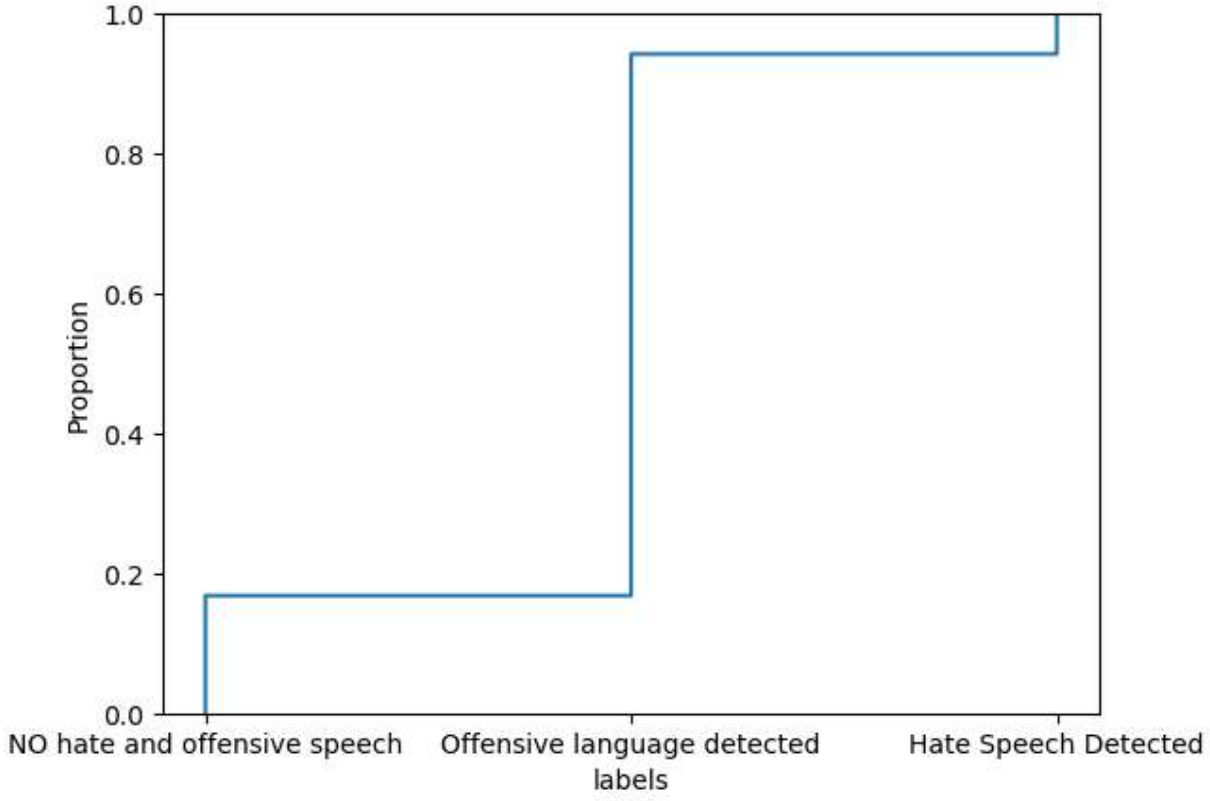
Strip plot

In [55]:
```
sns.stripplot(df)
plt.xticks(rotation=90)
plt.show()
```



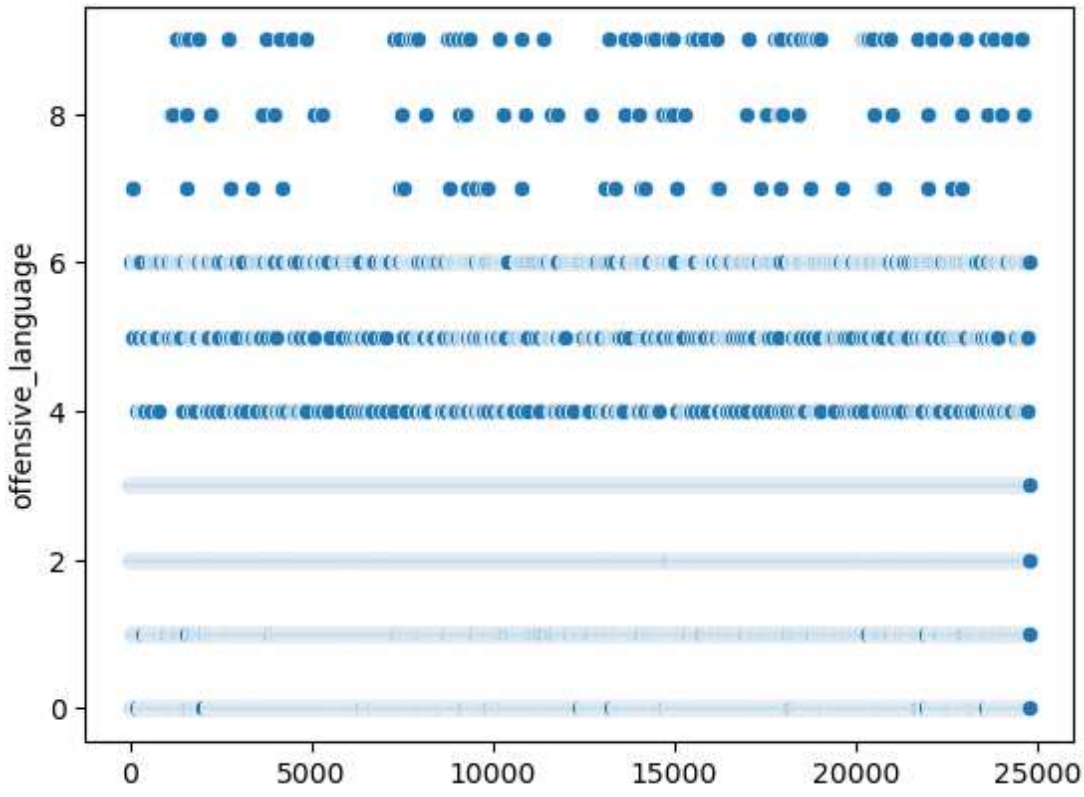
ECDF Plot

In [70]:
```
sns.ecdfplot(x=df['labels'], data=df)
plt.show()
```



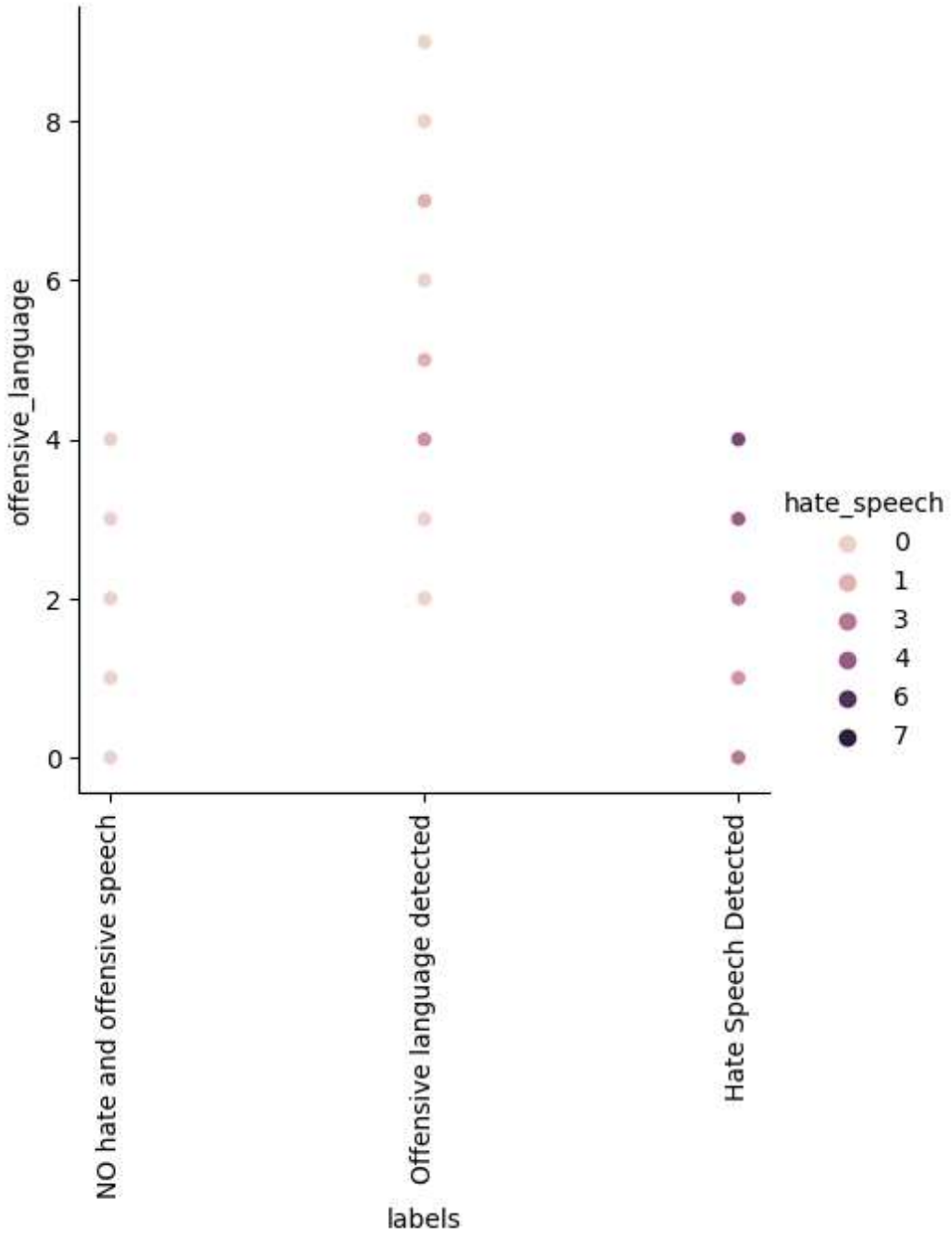
Scatterplot

In [74]:
```
sns.scatterplot(df['offensive_language'])
plt.show()
```



Relplot

In [85]:
```
sns.relplot(x='labels', y='offensive_language', hue = 'hate_speech', data=df)
plt.xticks(rotation=90)
plt.show()
```



In [86]:
```
df=df[['tweet','labels']]
```

In [87]:
```
df
```

Out[87]:

	tweet	labels
0	!!! RT @mayasolovely: As a woman you shouldn't...	NO hate and offensive speech
1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...	Offensive language detected
2	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...	Offensive language detected
3	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...	Offensive language detected
4	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...	Offensive language detected
...	...	...
24778	you's a muthaf***in lie “@LifeAsKing: @2...	Offensive language detected
24779	you've gone and broke the wrong heart baby, an...	NO hate and offensive speech
24780	young buck wanna eat!!.. dat nigguh like I ain...	Offensive language detected
24781	youu got wild bitches tellin you lies	Offensive language detected
24782	~~Ruffled | Ntac Eileen Dahlia - Beautiful col...	NO hate and offensive speech

24783 rows × 2 columns

In [88]:
```
def clean(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]','',text)
    text = re.sub('https?://\S+|www\.\S+','',text)
    text = re.sub('<.*?>+','',text)
    text = re.sub('[%s]'% re.escape(string.punctuation),'',text)
    text =  re.sub('\n','',text)
    text = re.sub('\w*\d\w*','',text)
    text = [word for word in text.split(' ') if word not in stopword]
    text = " ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text = " ".join(text)
    return text


df["tweet"] = df["tweet"].apply(clean)
print(df.head())
```

```
                                               tweet  \
0   rt mayasolov woman shouldnt complain clean ho...
1   rt  boy dat coldtyga dwn bad cuffin dat hoe  ...
2   rt urkindofbrand dawg rt  ever fuck bitch sta...
3             rt cganderson vivabas look like tranni
4   rt shenikarobert shit hear might true might f...

                        labels
0  NO hate and offensive speech
1   Offensive language detected
2   Offensive language detected
3   Offensive language detected
4   Offensive language detected
```

In [89]:
```
df.isnull().sum()
```

Out[89]:
```
tweet     0
labels    0
dtype: int64
```

In [90]:
```
df.head()
```

Out[90]:

	tweet	labels
0	rt mayasolov woman shouldnt complain clean ho...	NO hate and offensive speech
1	rt boy dat coldtyga dwn bad cuffin dat hoe ...	Offensive language detected
2	rt urkindofbrand dawg rt ever fuck bitch sta...	Offensive language detected
3	rt cganderson vivabas look like tranni	Offensive language detected
4	rt shenikarobert shit hear might true might f...	Offensive language detected

Decision Tree

In [91]:
```
x=np.array(df["tweet"])
y=np.array(df["labels"])
```

In [92]:
```
cv=CountVectorizer()
```

In [93]:
```
x=cv.fit_transform(x)
```

In [94]:
```
x
```

Out[94]:
```
<24783x25693 sparse matrix of type '<class 'numpy.int64'>'
        with 197862 stored elements in Compressed Sparse Row format>
```

In [95]:
```
y
```

Out[95]:
```
array(['NO hate and offensive speech', 'Offensive language detected',
       'Offensive language detected', ..., 'Offensive language detected',
       'Offensive language detected', 'NO hate and offensive speech'],
      dtype=object)
```

In [96]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42
```

In [97]:
```
clf=DecisionTreeClassifier()
```

In [98]:
```
clf.fit(x_train,y_train)
```

Out[98]:
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

In [99]:
```
y_pred=clf.predict(x_test)
```

In [100]:
```
from sklearn.metrics import accuracy_score, classification_report

acc = accuracy_score(y_pred,y_test)
print(acc)
```

```
0.8784692505196234
```

In [101]:
```
print('Classification Report')
print(classification_report(y_pred,y_test))
```

```
Classification Report
                              precision    recall  f1-score   support

        Hate Speech Detected       0.31      0.37      0.34       399
NO hate and offensive speech       0.82      0.82      0.82      1388
 Offensive language detected       0.93      0.92      0.93      6392

                    accuracy                           0.88      8179
                   macro avg       0.69      0.70      0.70      8179
                weighted avg       0.88      0.88      0.88      8179
```

In [102]:
```
test_data = "I will kill you"
df1=cv.transform([test_data]).toarray()
print(clf.predict(df1))
```

```
['Hate Speech Detected']
```

In [103]:
```
test_data2 = "You are bad i don't like you"
df2=cv.transform([test_data2]).toarray()
print(clf.predict(df2))
```

```
['Offensive language detected']
```

In [104]:
```
test_data3 = "I will not kill you"
df3=cv.transform([test_data3]).toarray()
print(clf.predict(df3))
```

```
['Hate Speech Detected']
```

KNN

In [360]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
```

In [361]:
```
x_knn=np.array(df["tweet"])
y_knn=np.array(df["labels"])
```

In [362]:
```
cv_knn=CountVectorizer()
```

In [363]:
```
x_knn= cv.fit_transform(x_knn)
```

In [364]:
```
X_train_knn, X_test_knn, Y_train_knn, Y_test_knn = train_test_split(x_knn, y_knn,
```

In [365]:
```
knn = KNeighborsClassifier(n_neighbors=10)

#Training the model
knn.fit(X_train_knn,Y_train_knn)
```

Out[365]:
```
KNeighborsClassifier

KNeighborsClassifier(n_neighbors=10)
```

In [366]:
```
knn.score(X_test_knn,Y_test_knn)
```

Out[366]:
```
0.8482953399233407
```

In [367]:
```
#Making prediction on test set
Y_pred_knn= knn.predict(X_test_knn)
```

In [368]:
```
#Accuracy of KNN model
acc_knn = accuracy_score(Y_test_knn, Y_pred_knn)
print("Accuracy:", acc_knn)
```

```
Accuracy: 0.8482953399233407
```

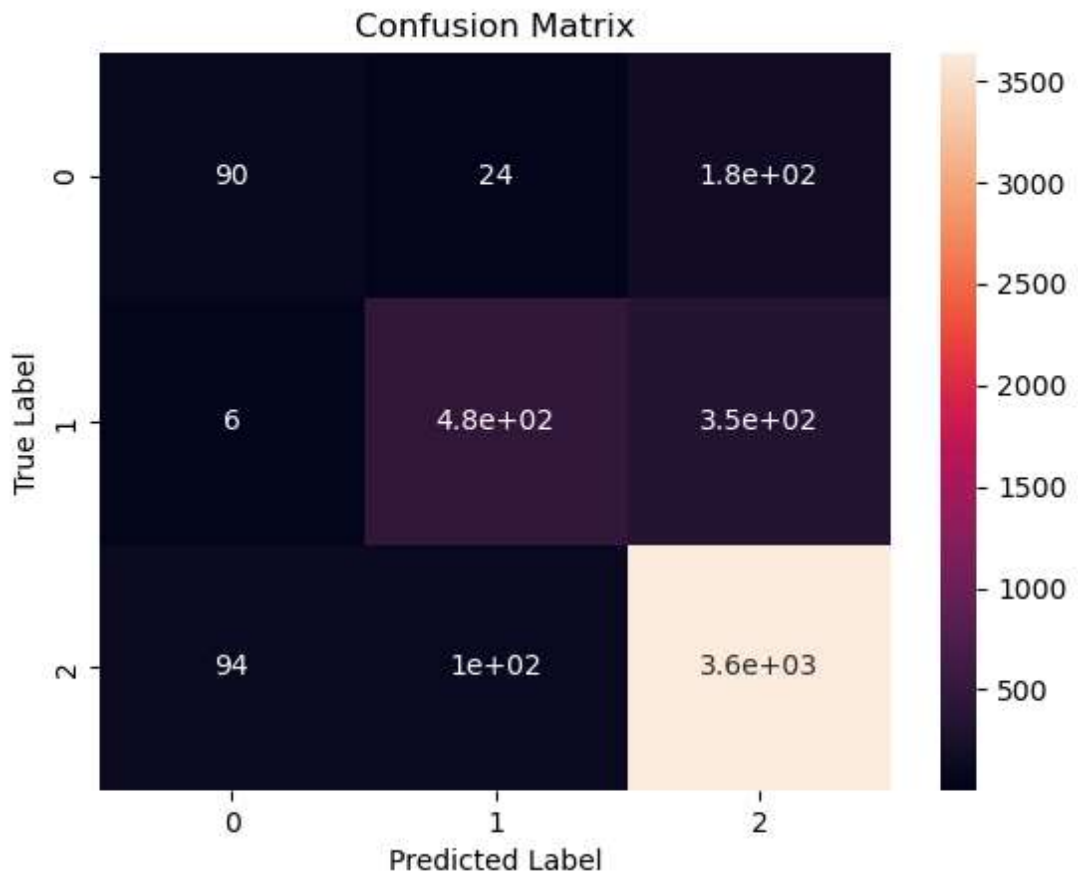
In [369]:
```
#Classification report of KNN model
print(classification_report(Y_test_knn, Y_pred_knn))
```

```
                              precision    recall  f1-score   support

        Hate Speech Detected       0.47      0.31      0.38       290
NO hate and offensive speech       0.79      0.57      0.67       835
 Offensive language detected       0.87      0.95      0.91      3832

                    accuracy                           0.85      4957
                   macro avg       0.71      0.61      0.65      4957
                weighted avg       0.84      0.85      0.84      4957
```

In [370]:
```
#Confusion Matrix of KNN model
cm=confusion_matrix(Y_test_knn, Y_pred_knn)
print('Confusion matrix : ')
print(cm)
print('True Positives(TP) = ', cm[0,0])
print('True Negatives(TN) = ', cm[1,1])
print('False Positives(FP) = ', cm[0,1])
print('False Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix :
[[  90   24  176]
 [   6  479  350]
 [  94  102 3636]]
True Positives(TP) =  90
True Negatives(TN) =  479
False Positives(FP) =  24
False Negatives(FN) =  6
```

In [371]:
```
sns.heatmap(cm, annot=True)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



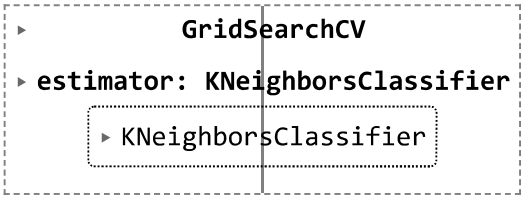
KNN Tuning

In [117]:
```
from sklearn.model_selection import GridSearchCV
```

In [118]:
```
#Grid Search

param_grid = {
    'n_neighbors': [3, 5, 7, 9],  # Test different values of k
    'weights': ['uniform', 'distance'],  # Weighting type
    'p': [1, 2],  # Distance metric (1 for Manhattan, 2 for Euclidean)
}
```

In [119]:
```
knn = KNeighborsClassifier()
grid_search = GridSearchCV(knn, param_grid, cv=5,scoring='accuracy')
grid_search.fit(X_train_knn, Y_train_knn)
```

Out[119]:



In [120]:
```
best_param = grid_search.best_params_
best_knn = KNeighborsClassifier(n_neighbors = best_param['n_neighbors'], weights =
best_knn.fit(X_train_knn, Y_train_knn)
Y_pred_knn = best_knn.predict(X_test_knn)
```

In [121]:
```
print("Best Hyperparameter : ", best_param)
```

```
Best Hyperparameter :  {'n_neighbors': 5, 'p': 1, 'weights': 'distance'}
```

In [122]:
```
acc_knn = accuracy_score(Y_test_knn, Y_pred_knn)
acc_knn
```

Out[122]:
```
0.8539439176921525
```

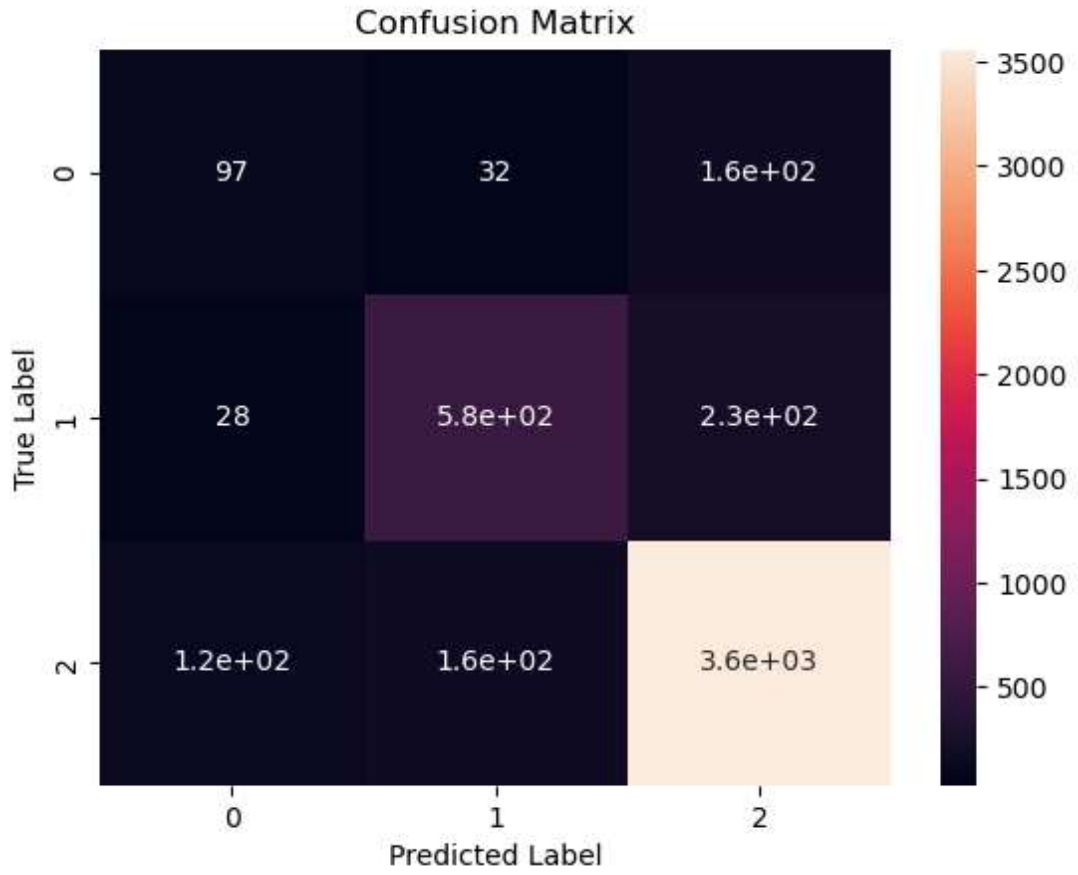
In [123]:
```
print (classification_report(Y_test_knn, Y_pred_knn))
```

```
                            precision    recall  f1-score   support

        Hate Speech Detected       0.40      0.33      0.37       290
NO hate and offensive speech       0.75      0.69      0.72       835
 Offensive language detected       0.90      0.93      0.91      3832

                    accuracy                           0.85      4957
                   macro avg       0.69      0.65      0.67      4957
                weighted avg       0.85      0.85      0.85      4957
```

In [124]:
```
cm=confusion_matrix(Y_test_knn, Y_pred_knn)
print('Confusion matrix : ')
print(cm)
print('True Positives(TP) = ', cm[0,0])
print('True Negatives(TN) = ', cm[1,1])
print('False Positives(FP) = ', cm[0,1])
print('False Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix :
[[  97   32  161]
 [  28  579  228]
 [ 116  159 3557]]
True Positives(TP) =  97
True Negatives(TN) =  579
False Positives(FP) =  32
False Negatives(FN) =  28
```

In [125]:
```
sns.heatmap(cm, annot=True)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



In [126]:
```
#Euclidean method

knn_classifier = KNeighborsClassifier(n_neighbors=10, metric='euclidean')
```

In [127]:
```
knn_classifier.fit(X_train_knn, Y_train_knn)
```

Out[127]:
```
KNeighborsClassifier

KNeighborsClassifier(metric='euclidean', n_neighbors=10)
```

In [128]:
```
Y_pred_knn_eu = knn_classifier.predict(X_test_knn)
```

In [129]:
```
accuracy = accuracy_score(Y_test_knn, Y_pred_knn_eu)  # For classification
print('accuracy:',accuracy)
```

```
accuracy: 0.8482953399233407
```

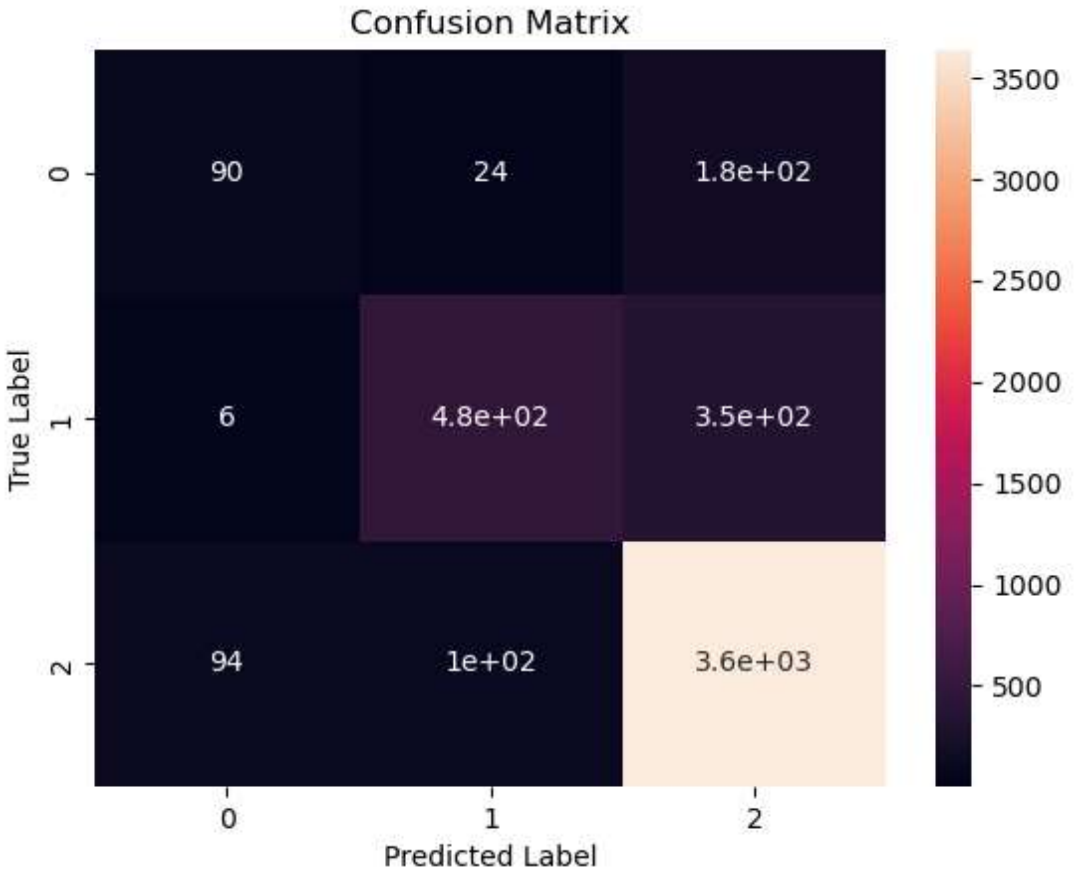
In [130]:
```
print(classification_report(Y_test_knn, Y_pred_knn_eu))
```

```
                              precision    recall  f1-score   support

        Hate Speech Detected       0.47      0.31      0.38       290
NO hate and offensive speech       0.79      0.57      0.67       835
 Offensive language detected       0.87      0.95      0.91      3832

                    accuracy                           0.85      4957
                   macro avg       0.71      0.61      0.65      4957
                weighted avg       0.84      0.85      0.84      4957
```

In [131]:
```
cm=confusion_matrix(Y_test_knn, Y_pred_knn_eu)
print('Confusion matrix : ')
print(cm)
print('True Positives(TP) = ', cm[0,0])
print('True Negatives(TN) = ', cm[1,1])
print('False Positives(FP) = ', cm[0,1])
print('False Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix :
[[  90   24  176]
 [   6  479  350]
 [  94  102 3636]]
True Positives(TP) =  90
True Negatives(TN) =  479
False Positives(FP) =  24
False Negatives(FN) =  6
```

In [132]:
```
sns.heatmap(cm, annot=True)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



In [133]:
```
#Manhatan distance method

knn_classifier = KNeighborsClassifier(n_neighbors=10, metric='manhattan')
```

In [134]:
```
knn_classifier.fit(X_train_knn, Y_train_knn)
```

Out[134]:
```
KNeighborsClassifier(metric='manhattan', n_neighbors=10)
```

In [135]:
```
Y_pred_knn_mh = knn_classifier.predict(X_test_knn)
```

In [136]:
```
accuracy_knn_mh = accuracy_score(Y_test_knn, Y_pred_knn_mh)
print('accuracy:',accuracy_knn_mh)
```

```
accuracy: 0.8452693161186201
```

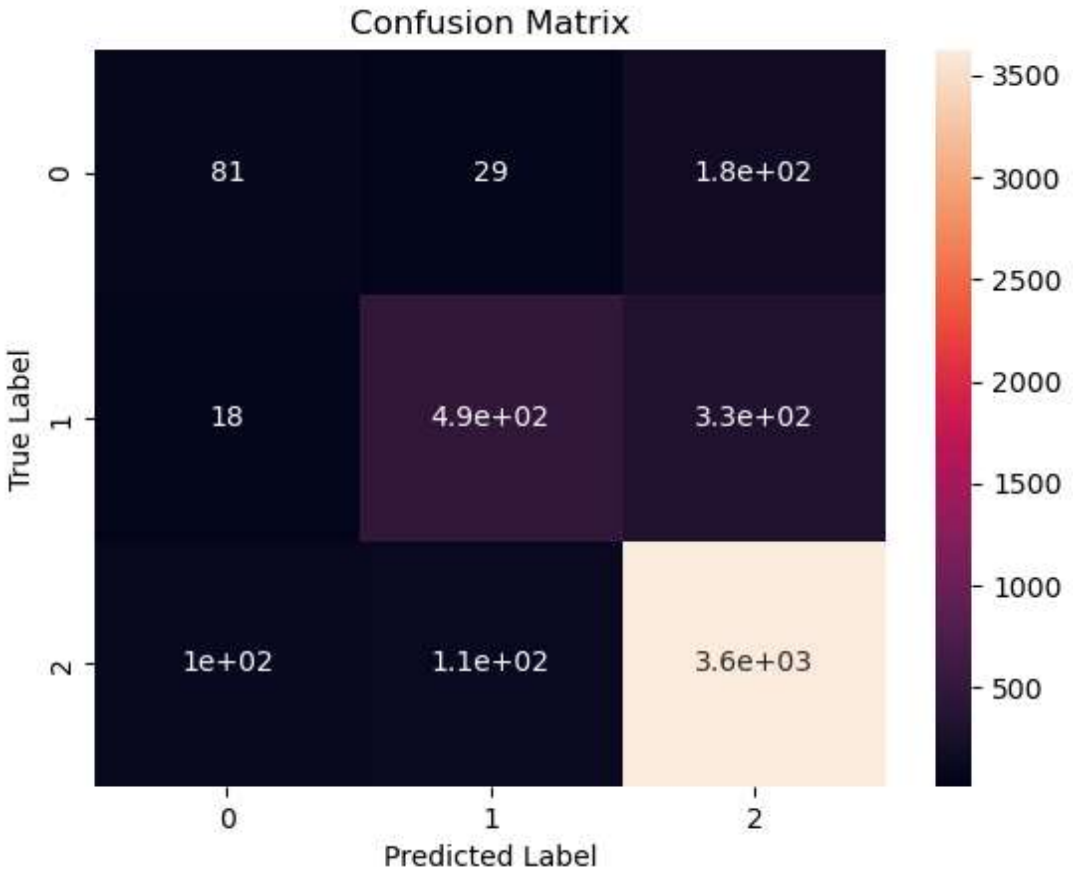
In [137]:
```
print(classification_report(Y_test_knn, Y_pred_knn))
```

```
                              precision    recall  f1-score   support

        Hate Speech Detected       0.40      0.33      0.37       290
NO hate and offensive speech       0.75      0.69      0.72       835
 Offensive language detected       0.90      0.93      0.91      3832

                    accuracy                           0.85      4957
                   macro avg       0.69      0.65      0.67      4957
                weighted avg       0.85      0.85      0.85      4957
```

In [138]:
```
cm=confusion_matrix(Y_test_knn, Y_pred_knn_mh)
print('Confusion matrix : ')
print(cm)
print('True Positives(TP) = ', cm[0,0])
print('True Negatives(TN) = ', cm[1,1])
print('False Positives(FP) = ', cm[0,1])
print('False Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix :
[[  81   29  180]
 [  18  486  331]
 [ 102  107 3623]]
True Positives(TP) =  81
True Negatives(TN) =  486
False Positives(FP) =  29
False Negatives(FN) =  18
```

In [139]:
```
sns.heatmap(cm, annot=True)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



In [140]:
```
#Minkowsky distance method

knn_classifier = KNeighborsClassifier(n_neighbors=10, metric='minkowski', p=2)
```

In [141]:
```
knn_classifier.fit(X_train_knn, Y_train_knn)
```

Out[141]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=10)
```

In [142]:
```
Y_pred_knn_mk = knn_classifier.predict(X_test_knn)
```

In [143]:
```
accuracy_knn_mk = accuracy_score(Y_test_knn, Y_pred_knn_mk)  # For classification
print('accuracy:',accuracy_knn_mk)
```

```
accuracy: 0.8482953399233407
```

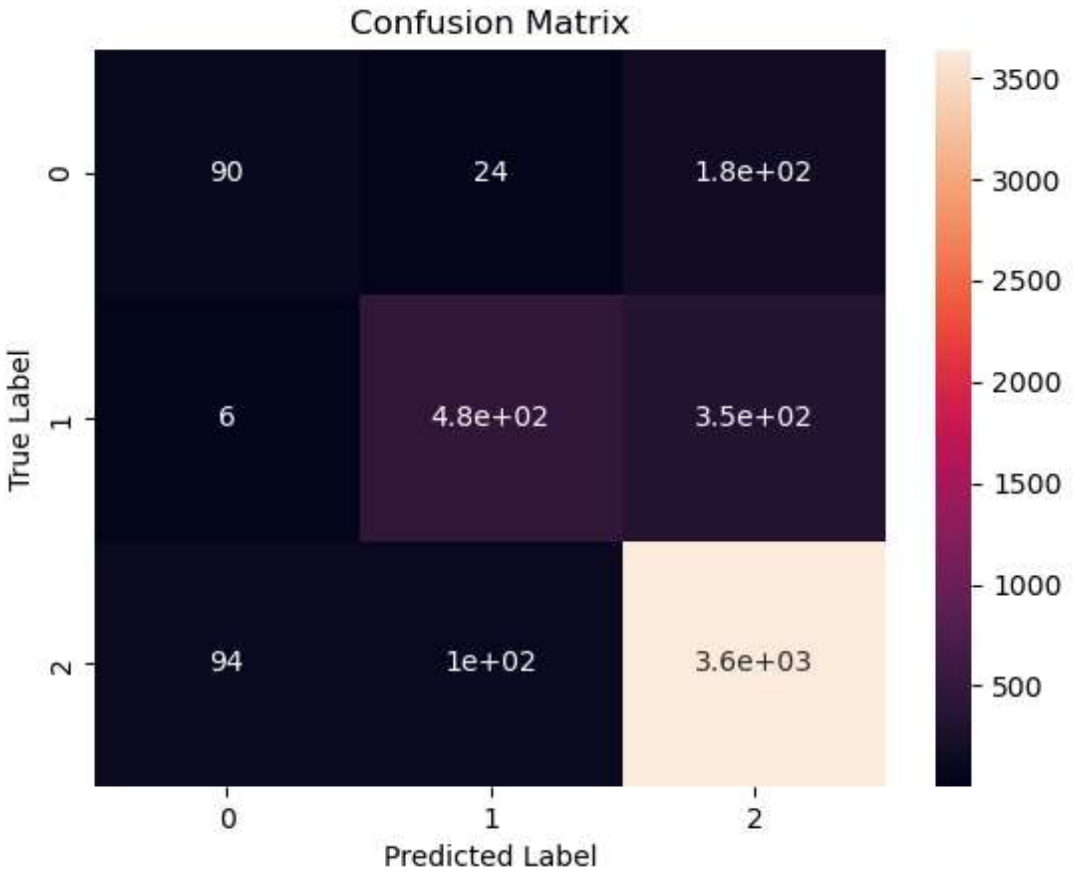
In [144]:
```
print(classification_report(Y_test_knn, Y_pred_knn_mk))
```

```
                              precision    recall  f1-score   support

        Hate Speech Detected       0.47      0.31      0.38       290
NO hate and offensive speech       0.79      0.57      0.67       835
 Offensive language detected       0.87      0.95      0.91      3832

                    accuracy                           0.85      4957
                   macro avg       0.71      0.61      0.65      4957
                weighted avg       0.84      0.85      0.84      4957
```

In [145]:
```
cm=confusion_matrix(Y_test_knn, Y_pred_knn_mk)
print('Confusion matrix : ')
print(cm)
print('True Positives(TP) = ', cm[0,0])
print('True Negatives(TN) = ', cm[1,1])
print('False Positives(FP) = ', cm[0,1])
print('False Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix :
[[  90   24  176]
 [   6  479  350]
 [  94  102 3636]]
True Positives(TP) =  90
True Negatives(TN) =  479
False Positives(FP) =  24
False Negatives(FN) =  6
```

In [146]:
```
sns.heatmap(cm, annot=True)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



SVM

In [147]:
```
from sklearn.svm import SVC
from sklearn.multiclass import OneVsRestClassifier, OneVsOneClassifier
```

In [148]:
```
x_svm=np.array(df["tweet"])
y_svm=np.array(df["labels"])
```

In [149]:
```
cv_svm=CountVectorizer()
x_svm= cv.fit_transform(x_svm)
```

In [150]:
```
X_train_svm, X_test_svm, Y_train_svm, Y_test_svm = train_test_split(x_svm, y_svm,
```

In [151]:
```
#Running with default hyperparameter
svc = SVC()
```

In [152]:
```
svc.fit(X_train_svm, Y_train_svm)
```

Out[152]:
```
SVC
SVC()
```

In [153]:
```
Y_pred_svm = svc.predict(X_test_svm)
```

In [154]:
```
#Accuracy of SVM model
acc_svm = accuracy_score(Y_test_svm, Y_pred_svm)
print("Accuracy Score with default parameter : {:.2f}%". format(acc_svm*100))
```

```
Accuracy Score with default parameter : 89.29%
```

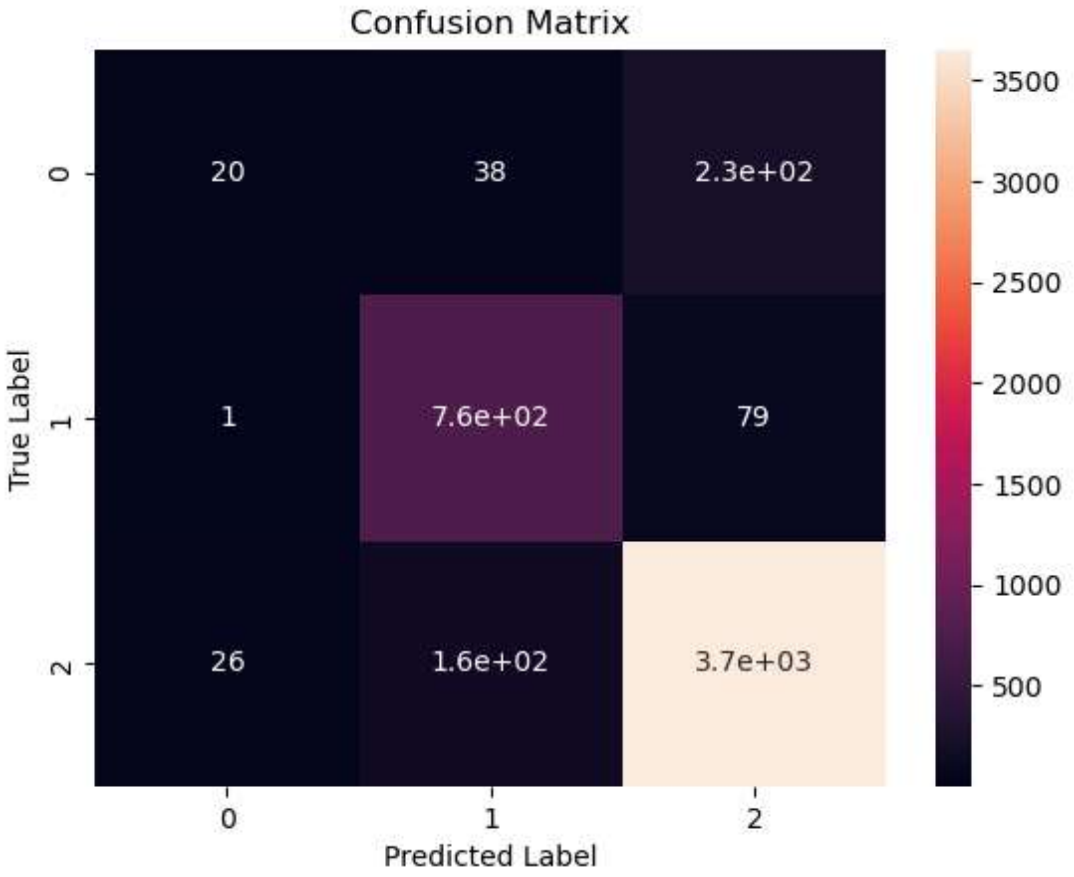
In [155]:
```
#Classification report of SVM model
print(classification_report(Y_test_svm, Y_pred_svm))
```

```
                              precision    recall  f1-score   support

        Hate Speech Detected       0.43      0.07      0.12       290
NO hate and offensive speech       0.80      0.90      0.85       835
 Offensive language detected       0.92      0.95      0.94      3832

                    accuracy                           0.89      4957
                   macro avg       0.71      0.64      0.63      4957
                weighted avg       0.87      0.89      0.87      4957
```

In [156]:
```
#Confusion Matrix of SVM model
cm = confusion_matrix(Y_test_svm, Y_pred_svm)
print('Confusion matrix : ')
print(cm)
print('True Positives(TP) = ', cm[0,0])
print('True Negatives(TN) = ', cm[1,1])
print('False Positives(FP) = ', cm[0,1])
print('False Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix :
[[  20   38  232]
 [   1  755   79]
 [  26  155 3651]]
True Positives(TP) =  20
True Negatives(TN) =  755
False Positives(FP) =  38
False Negatives(FN) =  1
```

In [157]:
```
sns.heatmap(cm, annot=True)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



In [158]:
```
#SVM with linear kernel

linear_svc = SVC(kernel='linear')
```

In [159]:
```
linear_svc.fit(X_train_svm, Y_train_svm)
```

Out[159]:
```
SVC
SVC(kernel='linear')
```

In [160]:
```
Y_predLinear_svm = linear_svc.predict(X_test_svm)
```

In [161]:
```
acc_svm = accuracy_score(Y_test_svm, Y_predLinear_svm)
print("Accuracy Score with Linear Kernel : {:.2f}%". format(acc_svm*100))
```

```
Accuracy Score with Linear Kernel : 88.42%
```

In [162]:
```
print(classification_report(Y_test_svm, Y_predLinear_svm))
```

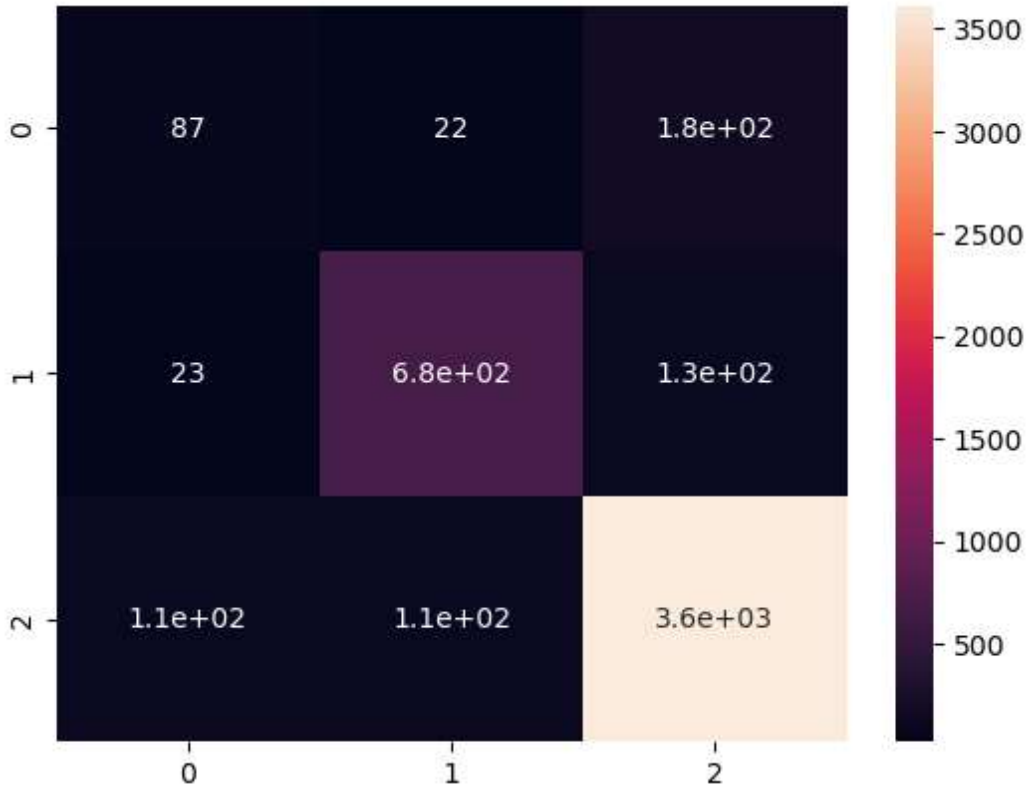
```
                              precision    recall  f1-score   support

        Hate Speech Detected       0.40      0.30      0.34       290
NO hate and offensive speech       0.84      0.82      0.83       835
 Offensive language detected       0.92      0.94      0.93      3832

                    accuracy                           0.88      4957
                   macro avg       0.72      0.69      0.70      4957
                weighted avg       0.88      0.88      0.88      4957
```

In [163]:
```
cm = confusion_matrix(Y_test_svm, Y_predLinear_svm)
print('Confusion matrix : ')
print(cm)
print('True Positives(TP) = ', cm[0,0])
print('True Negatives(TN) = ', cm[1,1])
print('False Positives(FP) = ', cm[0,1])
print('False Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix :
[[  87   22  181]
 [  23  684  128]
 [ 109  111 3612]]
True Positives(TP) =  87
True Negatives(TN) =  684
False Positives(FP) =  22
False Negatives(FN) =  23
```

In [164]:
```
sns.heatmap(cm, annot=True)
```

Out[164]: <Axes: >



In [165]:
```
#SVM with RBF Kernel

rbf_svc = SVC(kernel='rbf')
```

In [166]:
```
rbf_svc.fit(X_train_svm, Y_train_svm)
```

Out[166]:
```
SVC
SVC()
```

In [167]:
```
Y_predRBF_svm = rbf_svc.predict(X_test_svm)
```

In [168]:
```
acc_rbf_svm = accuracy_score(Y_test_svm, Y_predRBF_svm)
print("Accuracy Score with RBF Kernel : {:.2f}%". format(acc_rbf_svm*100))
```

```
Accuracy Score with RBF Kernel : 89.29%
```

In [169]:
```
print(classification_report(Y_test_svm, Y_predRBF_svm))
```

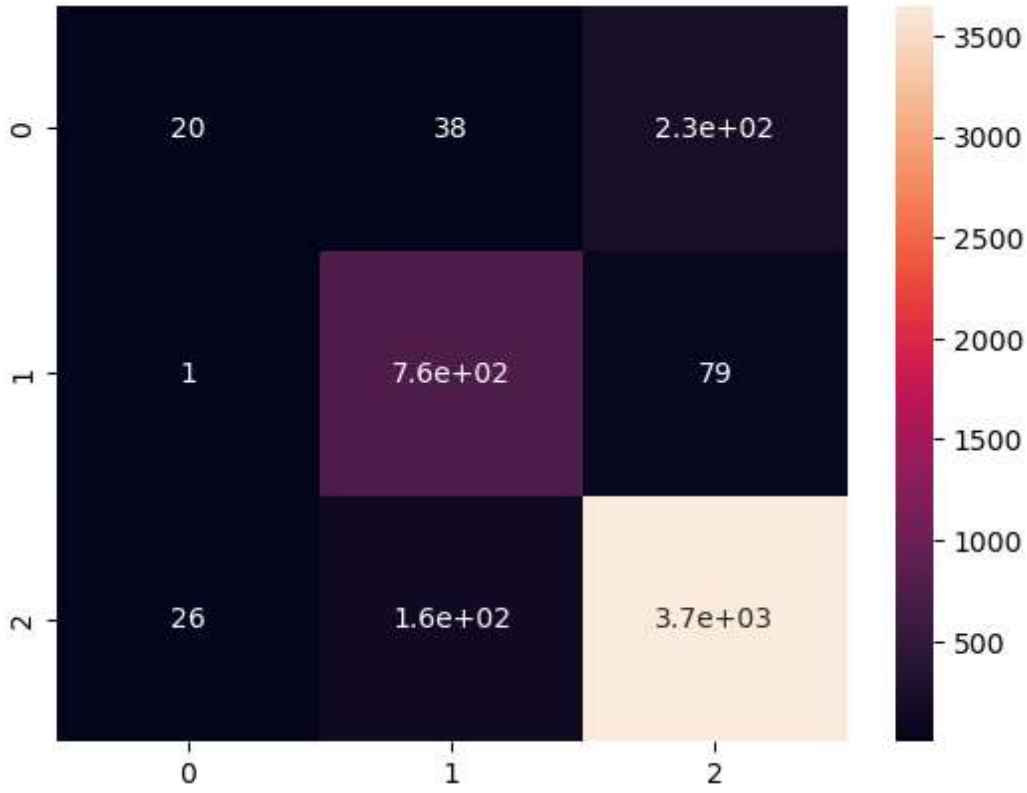
```
                              precision    recall  f1-score   support

        Hate Speech Detected       0.43      0.07      0.12       290
NO hate and offensive speech       0.80      0.90      0.85       835
 Offensive language detected       0.92      0.95      0.94      3832

                    accuracy                           0.89      4957
                   macro avg       0.71      0.64      0.63      4957
                weighted avg       0.87      0.89      0.87      4957
```

In [170]:
```
cm = confusion_matrix(Y_test_svm, Y_predRBF_svm)
print('Confusion matrix : ')
print(cm)
print('True Positives(TP) = ', cm[0,0])
print('True Negatives(TN) = ', cm[1,1])
print('False Positives(FP) = ', cm[0,1])
print('False Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix :
[[  20   38  232]
 [   1  755   79]
 [  26  155 3651]]
True Positives(TP) =  20
True Negatives(TN) =  755
False Positives(FP) =  38
False Negatives(FN) =  1
```

In [171]:
```
sns.heatmap(cm, annot=True)
```

Out[171]: <Axes: >



In [172]:
```
#SVM with Sigmoid Kernel

sigmoid_svc = SVC(kernel='sigmoid')
```

In [173]:
```
sigmoid_svc.fit(X_train_svm, Y_train_svm)
```

Out[173]:
```
SVC
SVC(kernel='sigmoid')
```

In [174]:
```
Y_predsigmoid_svm = sigmoid_svc.predict(X_test_svm)
```

In [175]:
```
acc_sigmoid_svm = accuracy_score(Y_test_svm, Y_predsigmoid_svm)
print("Accuracy Score with Sigmoid Kernel : {:.2f}%". format(acc_sigmoid_svm*100)
```

```
Accuracy Score with Sigmoid Kernel : 88.78%
```

In [176]:
```
print(classification_report(Y_test_svm, Y_predsigmoid_svm))
```

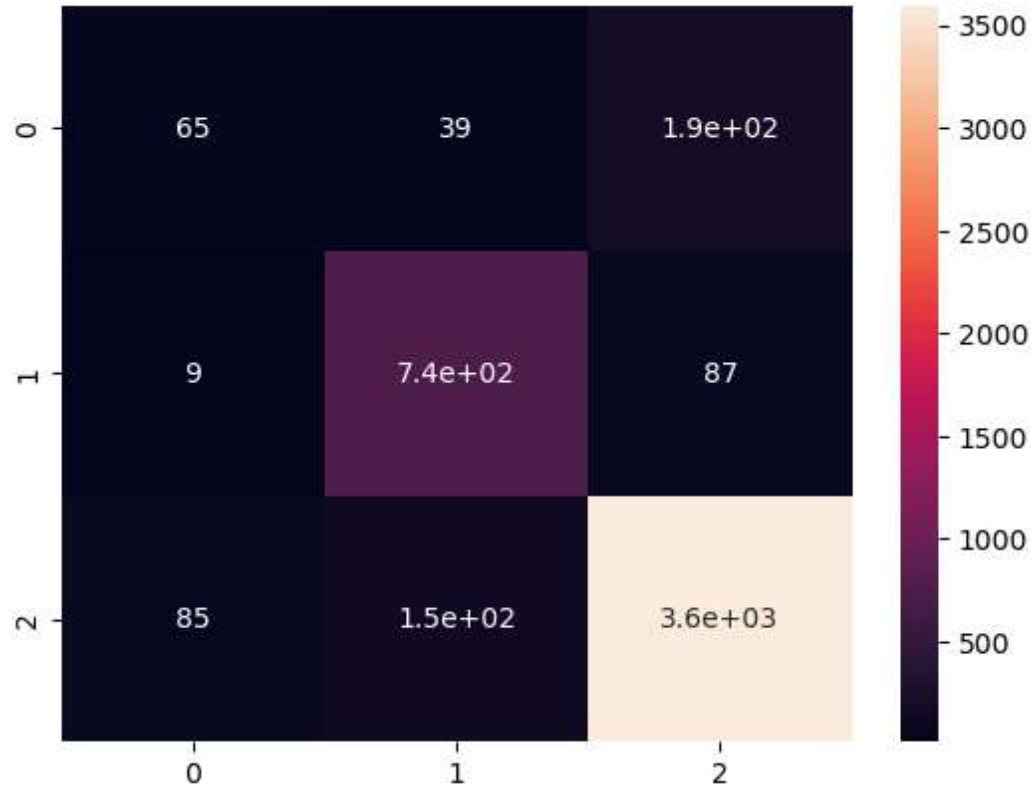
```
                              precision    recall  f1-score   support

        Hate Speech Detected       0.41      0.22      0.29       290
NO hate and offensive speech       0.80      0.89      0.84       835
 Offensive language detected       0.93      0.94      0.93      3832

                    accuracy                           0.89      4957
                   macro avg       0.71      0.68      0.69      4957
                weighted avg       0.88      0.89      0.88      4957
```

In [177]:
```
cm = confusion_matrix(Y_test_svm, Y_predsigmoid_svm)
print('Confusion matrix : ')
print(cm)
print('True Positives(TP) = ', cm[0,0])
print('True Negatives(TN) = ', cm[1,1])
print('False Positives(FP) = ', cm[0,1])
print('False Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix :
[[  65   39  186]
 [   9  739   87]
 [  85  150 3597]]
True Positives(TP) =  65
True Negatives(TN) =  739
False Positives(FP) =  39
False Negatives(FN) =  9
```

In [178]:
```
sns.heatmap(cm, annot=True)
```

Out[178]: <Axes: >



In [179]:
```
#SVM with Polynomial Kernel

poly_svc = SVC(kernel='poly')
```

In [180]:
```
poly_svc.fit(X_train_svm, Y_train_svm)
```

Out[180]:
```
SVC
SVC(kernel='poly')
```

In [181]:
```
Y_predPoly_svm = poly_svc.predict(X_test_svm)
```

In [182]:
```
acc_poly_svm = accuracy_score(Y_test_svm, Y_predPoly_svm)
print("Accuracy Score with Polynomial Kernel : {:.2f}%". format(acc_poly_svm*100)
```

```
Accuracy Score with Polynomial Kernel : 80.11%
```

In [183]:
```
print(classification_report(Y_test_svm, Y_predPoly_svm))
```

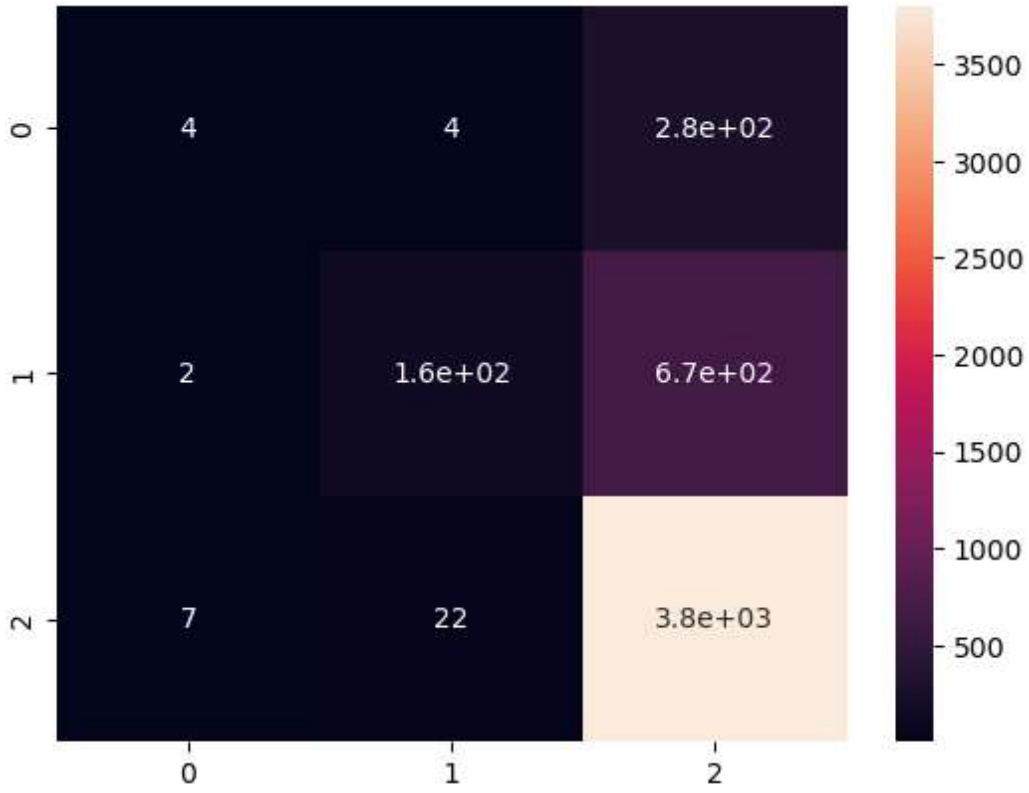
```
                              precision    recall  f1-score   support

        Hate Speech Detected       0.31      0.01      0.03       290
NO hate and offensive speech       0.86      0.20      0.32       835
 Offensive language detected       0.80      0.99      0.89      3832

                    accuracy                           0.80      4957
                   macro avg       0.66      0.40      0.41      4957
                weighted avg       0.78      0.80      0.74      4957
```

In [184]:
```
cm = confusion_matrix(Y_test_svm, Y_predPoly_svm)
print('Confusion matrix : ')
print(cm)
print('True Positives(TP) = ', cm[0,0])
print('True Negatives(TN) = ', cm[1,1])
print('False Positives(FP) = ', cm[0,1])
print('False Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix :
[[   4    4  282]
 [   2  164  669]
 [   7   22 3803]]
True Positives(TP) =  4
True Negatives(TN) =  164
False Positives(FP) =  4
False Negatives(FN) =  2
```

In [185]:
```
sns.heatmap(cm, annot=True)
```

Out[185]: <Axes: >



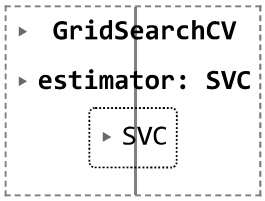
Tuning SVM

In [186]:
```
#Grid Search

param_grid = {
    'C' : [0.1, 1, 10, 100],
    'kernel' : ['linear', 'rbf', 'poly', 'sigmoid']
}
```

In [187]:
```
svcm = SVC()
```

In [188]:
```
grid_search = GridSearchCV(svcm, param_grid, cv=5)
```

In [189]:
```
grid_search.fit(X_train_svm, Y_train_svm)
```

Out[189]:


In [190]:
```
best_param_svm = grid_search.best_params_
print("Best hyperparameter : ", best_param_svm)
```

```
Best hyperparameter :  {'C': 0.1, 'kernel': 'linear'}
```

In [191]:
```
best_svm = SVC(C=best_param_svm['C'], kernel=best_param_svm['kernel'])
```

In [192]:
```
best_svm.fit(X_train_svm, Y_train_svm)
```

Out[192]:
```
SVC(C=0.1, kernel='linear')
```

In [193]:
```
Y_pred_tun_svm = best_svm.predict(X_test_svm)
```

In [194]:
```
acc_tun_svm = accuracy_score(Y_test_svm, Y_pred_tun_svm)
print("Accuracy after Grid Search : {:.2f}%". format(acc_tun_svm * 100))
```

```
Accuracy after Grid Search : 89.71%
```

In [195]:
```
print(classification_report(Y_test_svm, Y_pred_svm))
```

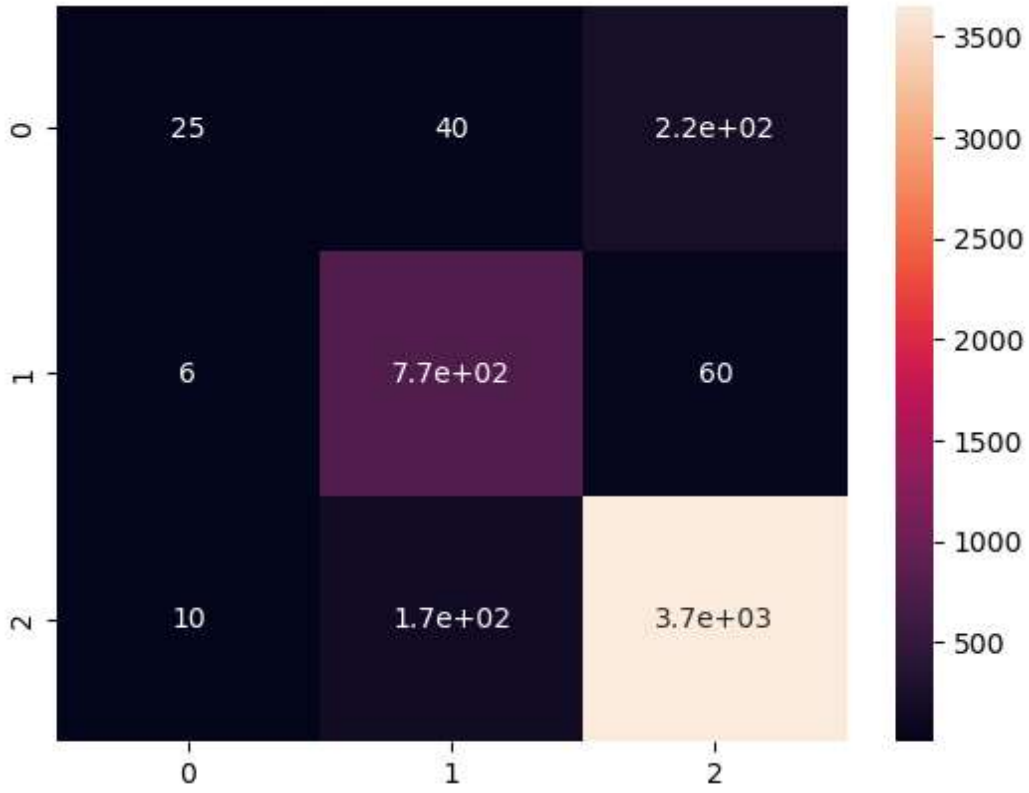
```
                              precision    recall  f1-score   support

        Hate Speech Detected       0.43      0.07      0.12       290
NO hate and offensive speech       0.80      0.90      0.85       835
 Offensive language detected       0.92      0.95      0.94      3832

                    accuracy                           0.89      4957
                   macro avg       0.71      0.64      0.63      4957
                weighted avg       0.87      0.89      0.87      4957
```

In [196]:
```
cm = confusion_matrix(Y_test_svm, Y_pred_tun_svm)
print('Confusion matrix : ')
print(cm)
print('True Positives(TP) = ', cm[0,0])
print('True Negatives(TN) = ', cm[1,1])
print('False Positives(FP) = ', cm[0,1])
print('False Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix :
[[  25   40  225]
 [   6  769   60]
 [  10  169 3653]]
True Positives(TP) =  25
True Negatives(TN) =  769
False Positives(FP) =  40
False Negatives(FN) =  6
```

In [197]:
```
sns.heatmap(cm, annot=True)
```

Out[197]: <Axes: >



In [198]:
```
#Tuning the model with Random Search

param_grid = {
    'C' : [0.1, 1, 10, 100],
    'kernel' : ['linear', 'rbf', 'poly', 'sigmoid']
}
```

In [199]:
```
svcmm = SVC()
```

In [200]:
```
from sklearn.model_selection import RandomizedSearchCV
random_search = RandomizedSearchCV(svcmm, param_grid, cv=5)
```

In [201]:
```
random_search.fit(X_train_svm, Y_train_svm)
```

Out[201]:

▸ **RandomizedSearchCV**

▸ **estimator: SVC**

▸ SVC

In [202]:
```
best_param_rs_svm = random_search.best_params_
print("Best Parameter : ", best_param_rs_svm)
```

```
Best Parameter :  {'kernel': 'rbf', 'C': 10}
```

In [203]:
```
best_svm_rs = SVC(C=best_param_rs_svm['C'], kernel=best_param_rs_svm['kernel'])
```

In [204]:
```
best_svm_rs.fit(X_train_svm, Y_train_svm)
```

Out[204]:
```
SVC
SVC(C=10)
```

In [205]:
```
Y_pred_rs_svm = best_svm_rs.predict(X_test_svm)
```

In [206]:
```
acc_rs_svm = accuracy_score(Y_test_svm, Y_pred_rs_svm)
print("Accuracy after Randomized Search: {:.2f}%" . format(acc_rs_svm * 100))
```

```
Accuracy after Randomized Search: 89.51%
```

In [207]:
```
print(classification_report(Y_test_svm, Y_pred_rs_svm))
```

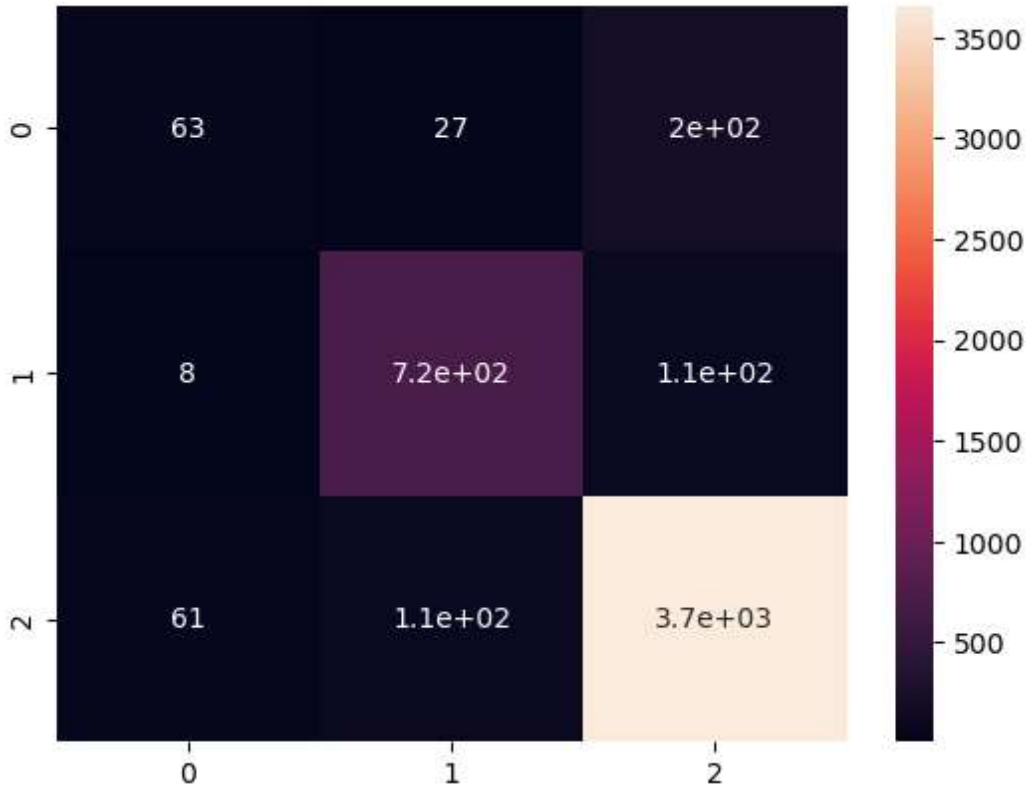
```
                              precision    recall  f1-score   support

        Hate Speech Detected       0.48      0.22      0.30       290
NO hate and offensive speech       0.84      0.86      0.85       835
 Offensive language detected       0.92      0.95      0.94      3832

                    accuracy                           0.90      4957
                   macro avg       0.75      0.68      0.69      4957
                weighted avg       0.88      0.90      0.89      4957
```

In [208]:
```
cm = confusion_matrix(Y_test_svm, Y_pred_rs_svm)
print('Confusion matrix : ')
print(cm)
print('True Positives(TP) = ', cm[0,0])
print('True Negatives(TN) = ', cm[1,1])
print('False Positives(FP) = ', cm[0,1])
print('False Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix :
[[  63   27  200]
 [   8  715  112]
 [  61  112 3659]]
True Positives(TP) =  63
True Negatives(TN) =  715
False Positives(FP) =  27
False Negatives(FN) =  8
```

In [209]:
```
sns.heatmap(cm, annot=True)
```

Out[209]: <Axes: >



Logistic Regression

In [322]:
```
from sklearn.linear_model import LogisticRegression
```

In [323]:
```
x_lor=np.array(df["tweet"])
y_lor=np.array(df["labels"])
```

In [324]:
```
cv_lor=CountVectorizer()
x_lor= cv_lor.fit_transform(x_lor)
```

In [325]:
```
X_train_lor, X_test_lor, Y_train_lor, Y_test_lor = train_test_split(x_lor, y_lor,
```

In [326]:
```
lor = LogisticRegression()

lor.fit(X_train_lor,Y_train_lor)
y_predicted_lor = lor.predict(X_test_lor)
```

In [327]:
```
lor.predict_proba(X_test_lor)
```

Out[327]:
```
array([[4.67877529e-03, 2.55407708e-04, 9.95065817e-01],
       [1.52501681e-03, 8.00522536e-06, 9.98466978e-01],
       [2.22506802e-03, 1.20128779e-01, 8.77646153e-01],
       ...,
       [9.90904192e-03, 2.74413957e-04, 9.89816544e-01],
       [1.68439435e-01, 1.05232980e-02, 8.21037268e-01],
       [1.12164226e-02, 1.47285981e-03, 9.87310718e-01]])
```

In [328]:
```
lor.score(X_test_lor,Y_test_lor)
```

Out[328]:
```
0.8926770223925762
```

In [329]:
```
lor.coef_
```

Out[329]:
```
array([[ 0.28616955,  0.        , -0.08501712, ..., -0.01417363,
        -0.00780366,  0.42725769],
       [-0.06017044,  0.        , -0.00061859, ..., -0.00194629,
         0.13422793, -0.02672619],
       [-0.22599911,  0.        ,  0.08563571, ...,  0.01611992,
        -0.12642428, -0.4005315 ]])
```

In [330]:
```
lor.intercept_   #y_intercept
```

Out[330]:
```
array([-1.75531072,  0.78914823,  0.96616249])
```

In [331]:
```
import math

def sigmoid(x):
    return 1/(1 + math.exp(-x))
```

In [332]:
```
def prediction_function(age):
    z= 0.14915498 * age - 5.80219736 # y = B0 + B1 * x
    y = sigmoid(z)
    return y
```

In [333]:
```
#Calculating Test Score
print('Test Score : ', lor.score(X_test_lor, Y_test_lor))
```

```
Test Score :  0.8926770223925762
```

In [334]:
```
#Predicting the train value
pred_train = lor.predict(X_train_lor)
```

In [335]:
```
#Predicting the test value
pred_test = lor.predict(X_test_lor)
```

In [336]:
```
#Classification report for the Training set
print(classification_report(Y_train_lor, pred_train))
```

```
                              precision    recall  f1-score   support

        Hate Speech Detected       0.97      0.66      0.79      1140
NO hate and offensive speech       0.97      0.98      0.98      3328
 Offensive language detected       0.97      1.00      0.98     15358

                    accuracy                           0.97     19826
                   macro avg       0.97      0.88      0.92     19826
                weighted avg       0.97      0.97      0.97     19826
```

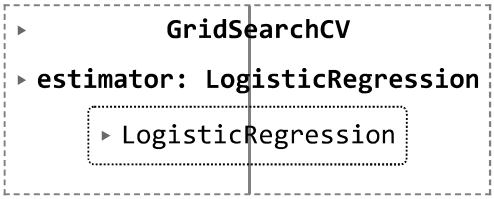
Tuning - LOR

In [225]:
```
param_grid = {
    'penalty' : ['l1', 'l2'],                          #l1 lasso l2 Ridge
    'C' : [0.1, 0.5, 1, 5, 10]
}
```

In [226]:
```
grid = GridSearchCV(estimator=lor, param_grid=param_grid, cv=5)
```

In [227]:
```
grid.fit(X_train_lor,Y_train_lor)
```

Out[227]:



In [228]:
```
 #Getting the best hyperparameters
best_params_lor = grid.best_params_

# Getting the best model
best_lor = grid.best_estimator_
```

In [229]:
```
# Making predictions on the test set
Y_pred_tun_lor = best_lor.predict(X_test_lor)
```

In [230]:
```
# Evaluating the best model
acc_tun_lor = accuracy_score(Y_test_lor, Y_pred_tun_lor)
print("Accuracy after Grid Search : {:.2f}%". format(acc_tun_lor * 100))
```

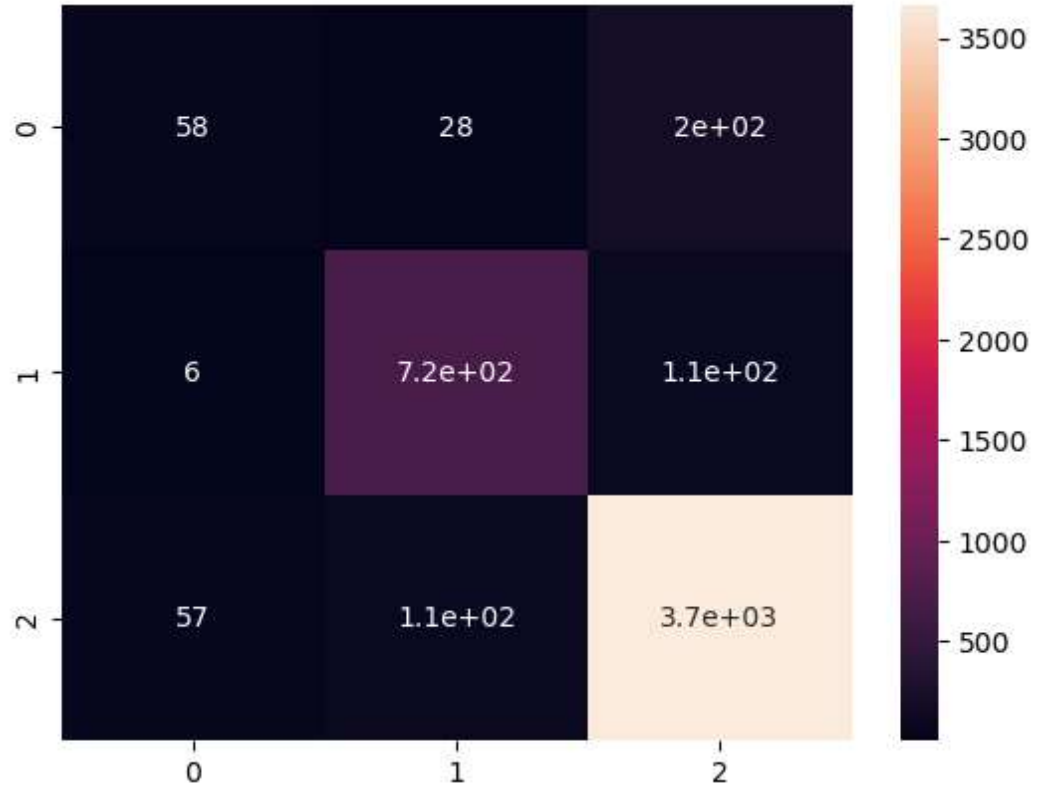
```
Accuracy after Grid Search : 89.57%
```

In [231]:
```
#Confusion Matrix after Grid Search
confusion = confusion_matrix(Y_test_lor, Y_pred_tun_lor)
print('Confusion Matrix:')
print(confusion)
```

```
Confusion Matrix:
[[  58   28  204]
 [   6  715  114]
 [  57  108 3667]]
```

In [232]:
```
sns.heatmap(confusion, annot=True)
```

Out[232]: <Axes: >



In [233]:
```
#Classification report after Grid Search
classification_rep = classification_report(Y_test_lor, Y_pred_tun_lor)
print('Classification Report:')
print(classification_rep)
```

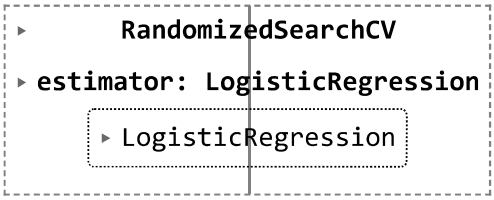
```
Classification Report:
                            precision    recall  f1-score   support

        Hate Speech Detected       0.48      0.20      0.28       290
NO hate and offensive speech       0.84      0.86      0.85       835
 Offensive language detected       0.92      0.96      0.94      3832

                    accuracy                           0.90      4957
                   macro avg       0.75      0.67      0.69      4957
                weighted avg       0.88      0.90      0.88      4957
```

In [234]:
```
from scipy.stats import uniform


# Defining hyperparameters and their possible values for tuning
param_dist = {
    'penalty': ['l1', 'l2'],
    'C': uniform(loc=0, scale=4)  # Random values between 0 and 4 for regularizat
}
```

In [235]:
```
# Creating a RandomizedSearchCV object
random_search = RandomizedSearchCV(lor, param_distributions=param_dist, n_iter=10
```

In [236]:
```
# Fitting the RandomizedSearchCV object to the training data
random_search.fit(X_train_lor, Y_train_lor)
```

Out[236]:



In [237]:
```
# Getting the best hyperparameters
best_params_rs_tun_lor = random_search.best_params_
```

In [238]:
```
# Getting the best model
best_lor_rs = random_search.best_estimator_
```

In [239]:
```
# Making predictions on the test set
Y_pred_rs_tun_lor = best_lor_rs.predict(X_test_lor)
```

In [240]:
```
# Evaluating the best model
acc_rs_tun_lor = accuracy_score(Y_test_lor, Y_pred_rs_tun_lor)
acc_rs_tun_lor
```

Out[240]:
```
0.8961065160379261
```

In [241]:
```
#Classification report after Randomized Search
classification_rep = classification_report(Y_test_lor, Y_pred_rs_tun_lor)
print('Classification Report:')
print(classification_rep)
```

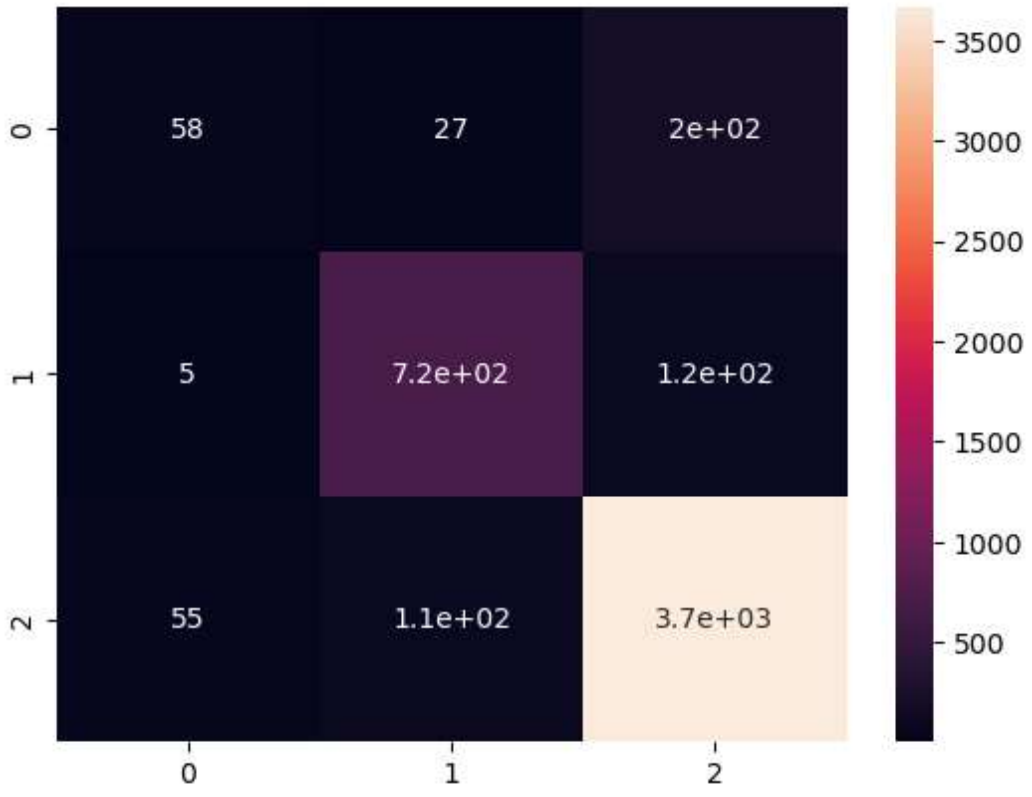
```
Classification Report:
                              precision    recall  f1-score   support

        Hate Speech Detected       0.49      0.20      0.28       290
NO hate and offensive speech       0.84      0.86      0.85       835
 Offensive language detected       0.92      0.96      0.94      3832

                    accuracy                           0.90      4957
                   macro avg       0.75      0.67      0.69      4957
                weighted avg       0.88      0.90      0.88      4957
```

In [242]:
```
#Confusion Matrix
confusion = confusion_matrix(Y_test_lor, Y_pred_rs_tun_lor)
print('Confusion Matrix:')
print(confusion)
```

```
Confusion Matrix:
[[  58   27  205]
 [   5  715  115]
 [  55  108 3669]]
```

In [243]:
```
sns.heatmap(confusion, annot=True)
```

Out[243]: <Axes: >



Random Forest

In [244]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [245]:
```
x_rf=np.array(df["tweet"])
y_rf=np.array(df["labels"])
```

In [246]:
```
cv_rf=CountVectorizer()
x_rf= cv_rf.fit_transform(x_rf)
```

In [247]:
```
X_train_rf, X_test_rf, Y_train_rf, Y_test_rf = train_test_split(x_rf, y_rf, test_
```

In [248]:
```
# Create a Random Forest classifier with 100 trees
rf = RandomForestClassifier(n_estimators=100, random_state=42)
```

In [249]:
```
# Fit the model on the training data
rf.fit(X_train_rf, Y_train_rf)
```

Out[249]:
```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [250]:
```
# Predict on the test data
y_pred_rf = rf.predict(X_test_rf)
```

In [251]:
```
# Calculate accuracy
acc_rf = rf.score(X_test_rf, Y_test_rf)
print("Accuracy:", acc_rf)
```

```
Accuracy: 0.8916683477910026
```

In [252]:
```
#Classification report
classification_rep_rf = classification_report(Y_test_rf, y_pred_rf)
print('Classification Report:')
print(classification_rep_rf)
```

```
Classification Report:
                              precision    recall  f1-score   support

        Hate Speech Detected       0.54      0.27      0.36       290
NO hate and offensive speech       0.82      0.82      0.82       835
 Offensive language detected       0.92      0.95      0.94      3832

                    accuracy                           0.89      4957
                   macro avg       0.76      0.68      0.71      4957
                weighted avg       0.88      0.89      0.88      4957
```

Tuning - RF

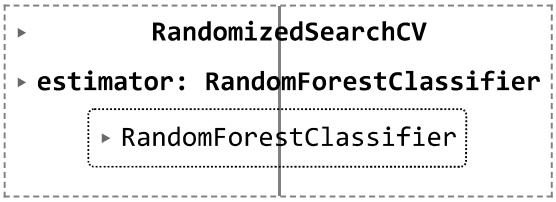
In [253]:
```
param_grid = {
    'n_estimators': [100, 200, 300, 400, 500],
    'max_depth': [5, 10, 15, 20, 25],
    'max_features': ['auto', 'sqrt', 'log2'],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
}
```

In [254]:
```
rf = RandomForestClassifier(random_state=42)
```

In [255]:
```
# Create a RandomizedSearchCV object
random_search = RandomizedSearchCV(rf, param_grid, n_iter=20, cv=5, random_state=
```

In [256]:
```
# Fit the model to the training data
random_search.fit(X_train_rf, Y_train_rf)
```

Out[256]:



In [257]:
```
# Print the best hyperparameters found
print("Best hyperparameters:", random_search.best_params_)
```

```
Best hyperparameters: {'n_estimators': 200, 'min_samples_split': 2, 'min_samples
_leaf': 1, 'max_features': 'auto', 'max_depth': 25}
```

In [258]:
```
# Get the best model
best_model = random_search.best_estimator_
```

In [259]:
```
# Make predictions on the test set
y_pred_tun_rf = best_model.predict(X_test_rf)
```

In [260]:
```
# Evaluate performance
acc_tun_rf = best_model.score(X_test_rf, Y_test_rf)
print("Accuracy:", acc_tun_rf)
```

```
Accuracy: 0.7740568892475288
```

In [261]:
```
#Classification report after Randomized Search
classification_rep_tun_rf = classification_report(Y_test_rf, y_pred_tun_rf)
print('Classification Report:')
print(classification_rep_tun_rf)
```

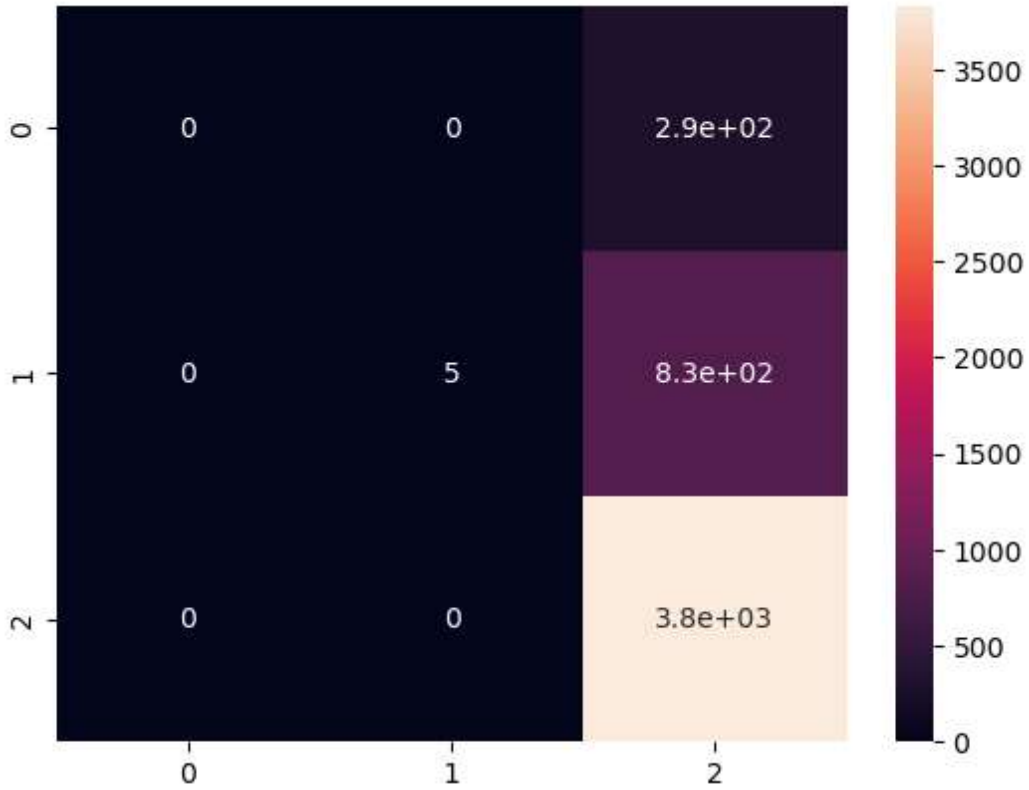
```
Classification Report:
                              precision    recall  f1-score   support

        Hate Speech Detected       0.00      0.00      0.00       290
NO hate and offensive speech       1.00      0.01      0.01       835
 Offensive language detected       0.77      1.00      0.87      3832

                    accuracy                           0.77      4957
                   macro avg       0.59      0.34      0.29      4957
                weighted avg       0.77      0.77      0.68      4957
```

In [262]:
```
confusion = confusion_matrix(Y_test_rf, y_pred_tun_rf)
print('Confusion Matrix:')
print(confusion)
```

```
Confusion Matrix:
[[   0    0  290]
 [   0    5  830]
 [   0    0 3832]]
```

In [263]:
```
sns.heatmap(confusion, annot=True)
```

Out[263]: <Axes: >



Naive Bayes

In [275]:
```
from sklearn import model_selection, naive_bayes, svm, metrics,feature_extractio
```

In [265]:
```
x_nb=np.array(df["tweet"])
y_nb=np.array(df["labels"])
```

In [266]:
```
cv_nb=CountVectorizer()
x_nb= cv_nb.fit_transform(x_nb)
```

In [267]:
```
X_train_nb, X_test_nb, Y_train_nb, Y_test_nb = train_test_split(x_nb, y_nb, test_
```

In [280]:
```
from sklearn.preprocessing import MaxAbsScaler

scaler = MaxAbsScaler()
scaled_data = scaler.fit_transform(X_train_nb)
X_train_nb = scaler.fit_transform(X_train_nb)
X_test_nb = scaler.transform(X_test_nb)
```

In [282]:
```
#choosing the naive bayes
bayes = naive_bayes.BernoulliNB()
bayes.fit(X_train_nb,Y_train_nb)
```

Out[282]:
```
BernoulliNB()
```

In [283]:
```
# predicting the y value based on bayes model
y_pred_nb=bayes.predict(X_test_nb)
```

In [310]:
```
#calculating the accuracy of the model on the testing data
acc_nb=metrics.accuracy_score(Y_test_nb,y_pred_nb)
acc_nb
```

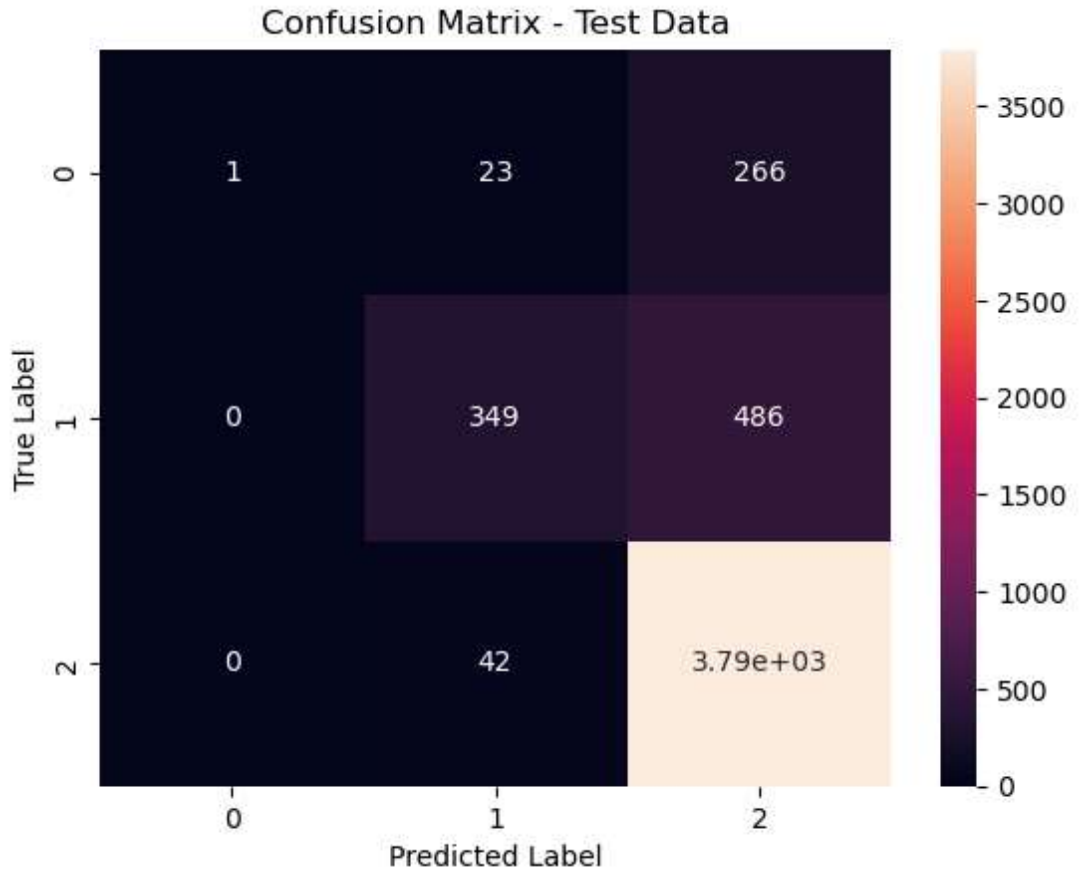
Out[310]:
```
0.8351825701028848
```

In [285]:
```
# generating the classification report for the performance measures.
print(metrics.classification_report(Y_test_nb, y_pred_nb))
```

```
                                precision    recall  f1-score   support

        Hate Speech Detected       1.00      0.00      0.01       290
NO hate and offensive speech       0.84      0.42      0.56       835
 Offensive language detected       0.83      0.99      0.91      3832

                    accuracy                           0.84      4957
                   macro avg       0.89      0.47      0.49      4957
                weighted avg       0.85      0.84      0.79      4957
```

In [286]:
```
cm=confusion_matrix(Y_test_nb,y_pred_nb)
print(cm)
sns.heatmap(cm, annot=True, fmt='.3g')
plt.title('Confusion Matrix - Test Data')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```
[[   1   23  266]
 [   0  349  486]
 [   0   42 3790]]
```

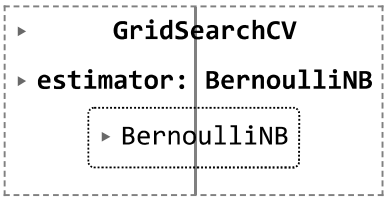


Tuning Naive bayes

In [293]:
```
 param_grid = {
'alpha': [0.1, 1, 10, 100],
'fit_prior': [True, False]
}
```

In [294]:
```
bayes = naive_bayes.BernoulliNB()
grid_search = GridSearchCV(bayes, param_grid, cv=5)
grid_search.fit(X_train_nb, Y_train_nb)
```

Out[294]:



In [295]:
```
best_param = grid_search.best_params_
best_nb = naive_bayes.BernoulliNB(alpha = best_param['alpha'], fit_prior = best_p
best_nb.fit(X_train_nb, Y_train_nb)
y_pred_1 = best_nb.predict(X_test_nb)
```

In [296]:
```
print("Best Hyperparameter : ", best_param)
```

```
Best Hyperparameter :  {'alpha': 0.1, 'fit_prior': True}
```

In [308]:
```
acc_nb_tun = accuracy_score(Y_test_nb, y_pred_1)
acc_nb_tun
```

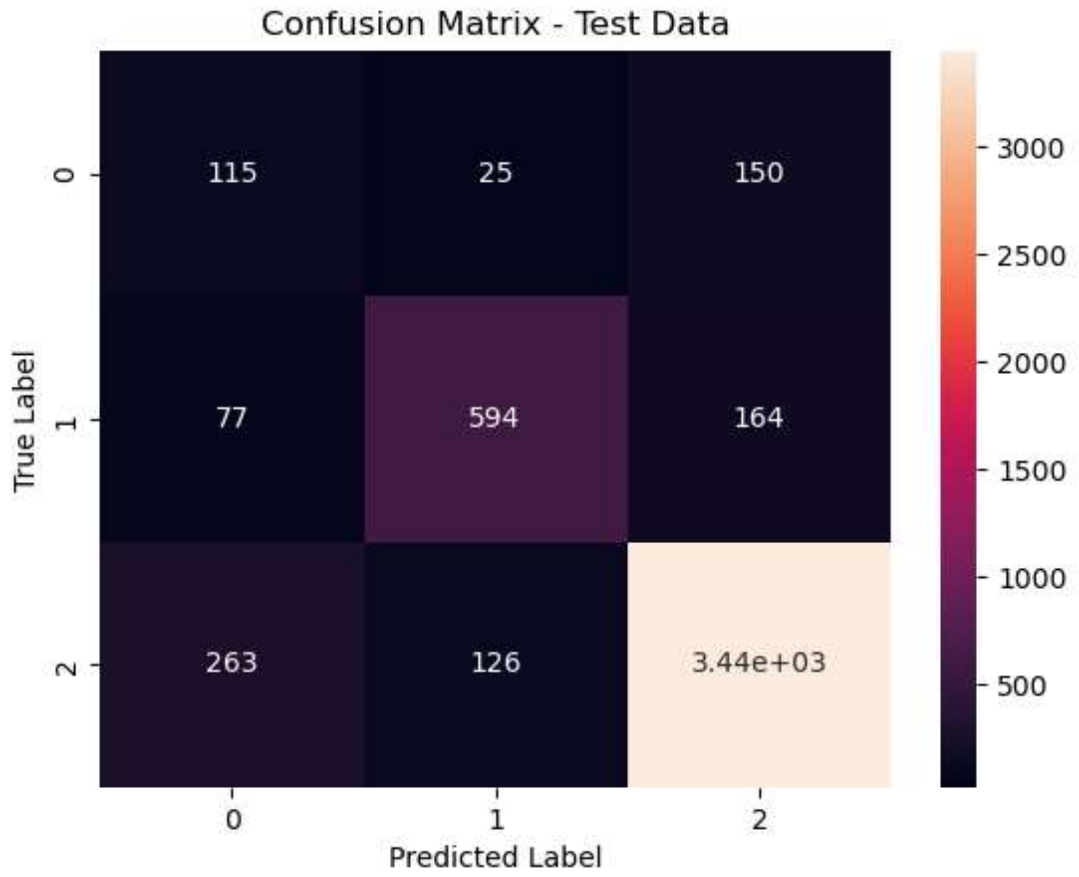
Out[308]:
```
0.8376033891466613
```

In [309]:
```
print (classification_report(Y_test_nb,y_pred_1))
```

```
                              precision    recall  f1-score   support

        Hate Speech Detected       0.25      0.40      0.31       290
NO hate and offensive speech       0.80      0.71      0.75       835
 Offensive language detected       0.92      0.90      0.91      3832

                    accuracy                           0.84      4957
                   macro avg       0.66      0.67      0.66      4957
                weighted avg       0.86      0.84      0.85      4957
```

In [299]:
```
cm=confusion_matrix(Y_test_nb,y_pred_1)
print(cm)
sns.heatmap(cm, annot=True, fmt='.3g')
plt.title('Confusion Matrix - Test Data')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```
[[ 115   25  150]
 [  77  594  164]
 [ 263  126 3443]]
```



In [300]:
```
#Randomized Search
param_dist = {
    'alpha': uniform(0.1, 2.0),  # Example: Uniform distribution for alpha
    'fit_prior':[True,False]
}
```

In [304]:
```
bayes = naive_bayes.BernoulliNB()
x_new_nb=scaler.fit_transform(x_nb)
```

In [305]:
```
randomized_search = RandomizedSearchCV(bayes, param_distributions=param_dist, n_i
randomized_search.fit(x_new_nb, y_nb)
```

Out[305]:

▸ RandomizedSearchCV
▸ estimator: BernoulliNB
▸ BernoulliNB

In [306]:
```
best_param = randomized_search.best_params_
print("Best Hyperparameter : ", best_param)
```

```
Best Hyperparameter :  {'alpha': 0.5152534796161073, 'fit_prior': False}
```

In [307]:
```
best_nb = naive_bayes.BernoulliNB(alpha = best_param['alpha'], fit_prior = best_p
best_nb.fit(X_train_nb, Y_train_nb)
y_pred_2 = best_nb.predict(X_test_nb)
```

In [311]:
```
acc_nb_tun_rs = accuracy_score(Y_test_nb, y_pred_2)
acc_nb_tun_rs
```

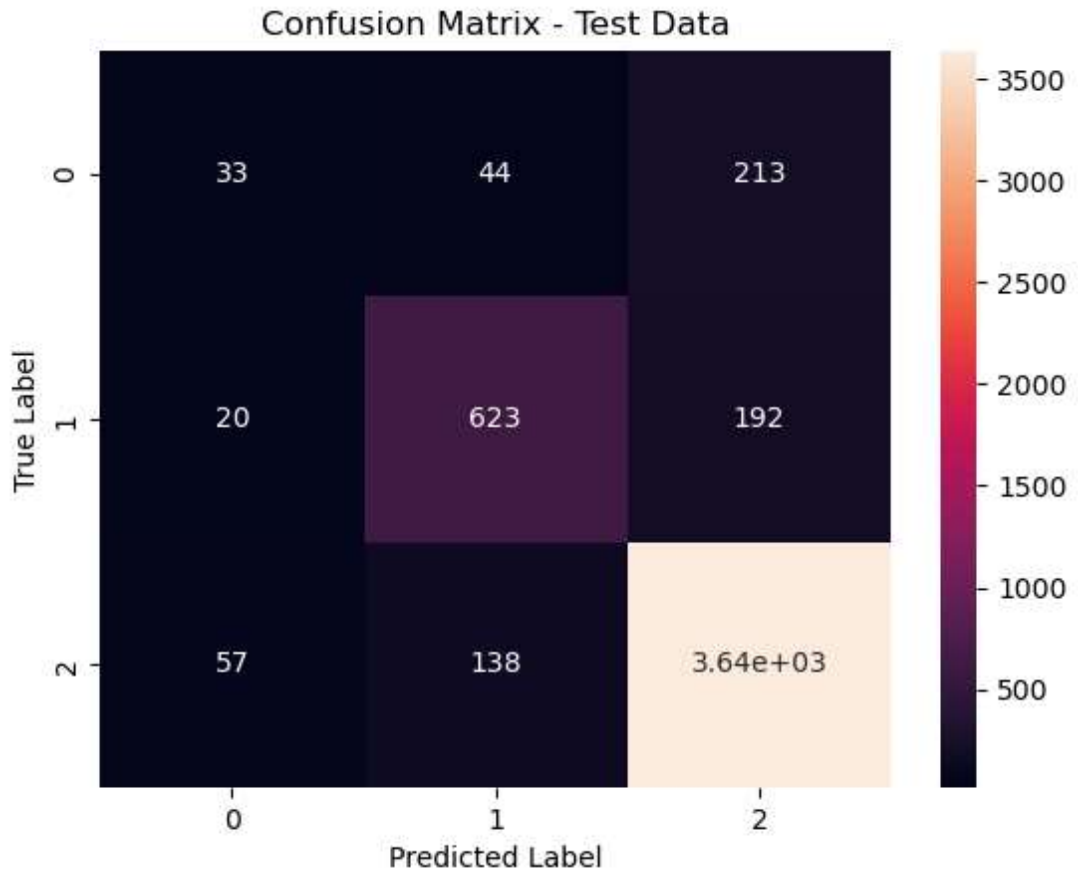
Out[311]:
```
0.866048012911035
```

In [312]:
```
print (classification_report(Y_test_nb,y_pred_2))
```

```
                              precision    recall  f1-score   support

        Hate Speech Detected       0.30      0.11      0.17       290
NO hate and offensive speech       0.77      0.75      0.76       835
 Offensive language detected       0.90      0.95      0.92      3832

                    accuracy                           0.87      4957
                   macro avg       0.66      0.60      0.62      4957
                weighted avg       0.84      0.87      0.85      4957
```

In [313]:
```
cm=confusion_matrix(Y_test_nb,y_pred_2)
print(cm)
sns.heatmap(cm, annot=True, fmt='.3g')
plt.title('Confusion Matrix - Test Data')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```
[[  33   44  213]
 [  20  623  192]
 [  57  138 3637]]
```



Pickle

In [316]:
```
import pickle as pkl
```

In [317]:
```
pkl.dump(svc,open('model.pkl','wb'))
```

In [ ]:
```

```