Multilabel Toxic Comment Classification

Introduction

In the twenty-first century, social media has offered several job prospects while also serving as a unique platform for people to freely voice their thoughts. Social media is a platform to express individual perspectives and thoughts while also making a positive contribution to the development of a safe environment where everyone may exercise their rights. However, some people believe that by using computers as virtual walls, they can abuse and harass other people's beliefs and personalities. Meanwhile, some groups among these users are taking advantage of this structure and abusing their freedom to apply their harmful worldview (i.e. insulting, verbal sexual harassment, threads, Obscene, etc.). One of their main goals is to identify harmful comments and develop an online toxicity monitoring system on numerous social media sites. The challenge's main goal is to create a multi-label classifier that can not only identify toxic comments but also detect the type of toxicity they contain, such as threats, obscenity, insults, and identity-based hate; currently, there are no practical tools that can distinguish these types of comments with a low rate of errors and high accuracy.

The purpose of this project is to apply a text classification algorithm to detect toxicity in text, which could help users avoid sending potentially harmful words while conversing with others, as well as to assess the toxicity of other users' comments. The major goal is to detect several sorts of toxicity in online comments, such as threats, vulgarity, insults, and identity-based hate, to create a model that can predict the intensity of hatred.

Methodology

The methods I tested in this project contain three basic components based on the reported challenges in the field of differentiating poisonous comments. First, a baseline classification solution was developed using a Naïve Bayes [NB] technique. Following that, a classification solution based on the Logistic regression approach was built. The next steps were performing the complete solution using pipeline approach.

The steps to develop my solution were as follows: Data gathering, Data preprocessing, Classification, Metrics Evaluation, Result preparation. A separate solution doing all the above steps using pipeline.

1) Data gathering

One of the most difficult aspects of classification is obtaining adequately labeled data from which a representative training set for modeling may be taken. Although there are numerous big unlabeled text corpora accessible, human-labeled data is significantly less common. Fortunately, some groups provide open-sourced labeled data sets for varied ethical motives. Wikipedia's chat pages are one such source. Over 159,571 text comments were manually labeled using the comments from this page, with each being assigned to one of six categories: Toxic, Severe Toxic, Obscene, Threat, Insult, and Identity Hate are the six categories. A closed Kaggle competition provides access to the data set. The data collection contains a textual user comment with six binary fields for each of the six toxicity levels.

2) Data Preprocessing

After data gathering from Kaggle, the most important step or barrier I came across was cleaning the data. The labeled data provided has many stop words, punctuations, links, etc. For example, ‘don’t’, ‘I’m’, these type of data without cleaning can become a problem for getting better results. As a result, lemmatization was used to condense vocabulary into it root form. So the steps, I took here was, cleaning the data such that all the html characters, brackets, non-ascii characters were removed and data was brought to a format which could be used further for training and testing the

classifiers I was going to use in this project. I have also applied stratified sampling approach to balance the multiple label while splitting the data into train and test data.

3) Classification

I have used 2 classifiers to basically train the dataset provided. Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. It is based on Bayes Theorem: $\overbrace{P(c \mid d)}^{\text{posterior}} = \frac{\overbrace{P(d \mid c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}}{P(d)}$

Knowing that d has occurred, the likelihood of c occurring can be calculated. The evidence is represented by event d, while the hypothesis to be authorized is represented by event c. The theorem assumes that all predictors/features are independent of one another and that their presence has no effect on the other. This is a basic simplification of the Bayes technique. The likelihood of one event, d, occurring is unrelated to the probability of another d event occurring. To classify an Internet comment as offensive or toxic, we'd look at our training data, which is divided into toxic and non-toxic categories.

The second classifier I used was Logistic Regression Classifier. Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The equation is similar to the equation of line I.e. z = w . x + b, where there is a dot product between w (weight vector) and x(feature vector) and b is scalar. The value of y is bound between 0 and 1. The logistic function is an S-shaped curve(sigmoid) that stretches from zero to one, while never being exactly zero and never being exactly one, either. While computing z value we sum up the product values for all the features and bias values and then we take sigmoid of the z value which gives the value of y which is between the 0 and 1.

I have used OneVsRestClassifier, it is helpful for a multilabel classification problem. This basically works consists of fitting one classifier per class. For every classifier the class is fitted against all the other classes. This estimator does multilabel classification using the binary relevance technique, which includes training one binary classifier individually for each label.

4) Metrics Evaluation

After training the data using the above classifiers and testing the data if the classifier has given accurate results or not. Like how much is the precision, recall and F1 score for the predicted results. I have used various evaluation metrics – Confusion Matrix, Accuracy, Precision, Recall, F score, ROC Auc.

Confusion matrix: For classification problems, it is a very useful performance measure. A matrix member called Ci,j indicates how many objects with label i are classed as label j. We're seeking for a diagonal Confusion matrix with no misclassified items. Our binary classification is well represented by the matrix below. The letter Positive stands for toxic label, while the letter negative stands for non-toxic label.

		Predicted Value	
		positive	**negative**
Actual Value	**Positive**	True Positive	False Negative
	negative	False Positive	True Negative

Fig: Confusion Matrix

Precision: measures what fraction of toxic classified comments were truly toxic. P = tp/(tp + fp)

Recall: measures what fraction of toxic comments are labeled correctly. R = tp/(tp + fn)

F-measure: F-measure is the weighted harmonic mean of Precision and Recall. Precision and recall are not of same type, so we go for harmonic mean, fractional values. Most common is where beta = 1, F = 2PR/(P+R).

Accuracy: It means how accurate is the calculated data and whether to consider the classifier or not i.e. how may comments are correctly labeled. But if the classes are imbalanced, we cannot rely on this metric for checking the performance of the classifier. In this data set, most of the comments are not toxic, we are getting high accuracy.

ROC AUC: It is a technique that compares classifier performance. Here, we measure the area under the curve (AUC). A perfect classifier will have a ROC AUC equal to 1, whereas a purely

random classifier will have a ROC AUC equal to 0.5. So, ROC AUC is the percentage of the ROC plot that is underneath the curve.

5) Result

I have shown the results for the naïve bayes and logistic regression approach. For naïve bayes, I have tried to train and predict the model in 2 ways. The first one is by using OneVsRestClassifier and the other one is for every label separately I have tried to train and predict the results using Naive Bayes classifier. Similarly for Logistic Regression. I will showcase the detailed result analysis in coming sections.

6) Pipeline

Pipeline is a way through which I was able to process my data, fit , transform and predict the results on dataset provided without having to do it individually for each thing. To begin with, I did some feature engineering and preprocessing on the dataset. I tried to count the number or words, character length, word length, number of stop words and add them to the train dataset as columns. Splitting the data into training and testing. Now to process my variables I tried to create a selector transformer that simply returns the one column in the dataset by the key value I pass. Two types of selector were created – text and numeric. Created a pipeline to pass an array of tuples, so the pipeline consists of selecting and performing tfidf on that column. Later fit to training data and transforming to apply it to training data.(return sparse matrix). Applied the classifier, did cross fold validation and after refitting the model tried to predict the results.

Experimental Setup

Data Gathering:

The data provided by Kaggle has 1,59,571 records in the train dataset and 1,53,164 in the test dataset. The train dataset has comment_text and six class labels: toxic, severe_toxic, obscene, threat, insult, identity_hate. Each label having 0 or 1 values saying whether it is present or not in the comment given. The test set has all the comments on which I would be predicting the results.

Percentage distribution:

Percent of toxics: 9.58 % Percent of severe_toxics: 1.0 % Percent of obscene: 5.29 % Percent of threats: 0.3 % Percent of insults: 4.94 % Percent of identity_hates: 0.88 %

Percentage of unlabeled comments is 89.83%

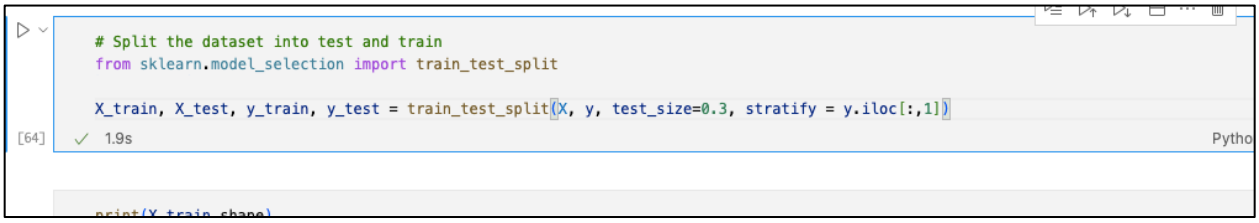
Data Preprocessing:

I used a function to clean the text and remove all unnecessary elements like, remove html chars, square brackets, parenthesis, punctuation marks, non-ascii characters, remove words containing numbers, converting the complete sentence into lower case, and lemmatizing and removing all the stop words. The text after data preprocessing was as below:

“explan edit usernam hardcor metallica fan reve...”

Data Split

The percentage of toxic comments is less than non-toxic comments so while splitting the data into train and test set, I tried to using stratified sampling approach to balance the class labels in both train and test dataset. So, stratified works very well for a single label. But for multilabel we must select a particular column and split or balance the labels in train and test dataset. Below is the screenshot how I used the method.



Vectorize the data:

I have used TfidfVectorizer to word vectorize and character vectorize, later combing both using make_union function.The let the vocabulary in the training data learn and use it to create a document-term matrix using fit_transform function. Later transforming the test data using the fitter vocabulary into a document-term matrix.

```
word_vectorizer = TfidfVectorizer(
    strip_accents='unicode',
    analyzer='word',
    token_pattern=r'\w{1,}',
    ngram_range=(1, 3),
    stop_words='english',
    sublinear_tf=True)

# character
char_vectorizer = TfidfVectorizer(ngram_range =(1,4),
                                  min_df=3, max_df=0.9,
                                  strip_accents='unicode',
                                  analyzer = 'char',
                                  stop_words = 'english',
                                  use_idf=1,
                                  smooth_idf=1,
                                  sublinear_tf=1,
                                  max_features=50000)

vectorizer = make_union(word_vectorizer, char_vectorizer)
vectorizer.fit(X_train)    # Fiting it on Train
```

Classifier used to solve the multilabel classification problem:

Transforming a multi-label classification problem into distinct single-class classifier tasks was one way to approach it. Binary Relevance was used, which analyzes each label as its own classification problem. However, the important assumption is that there is no association between the various labels. I used this approach for both Naïve Bayes and Logistic regression classifier, calculated the metrics separately for all the labels.

The Second approach I tried was OneVsRestClassifier which calculated separately for all the labels considering the rest 5 labels as one and generated the overall ROC acc for the classifier. The confusion matrix and evaluation metrics will be shared in the coming section.

I tried to calculate the mean score for all the label using cross_val score with Naive bayes classifier. The accuracy, precision, recall, F1 score have changed for all the 3 approaches.

Test result generation

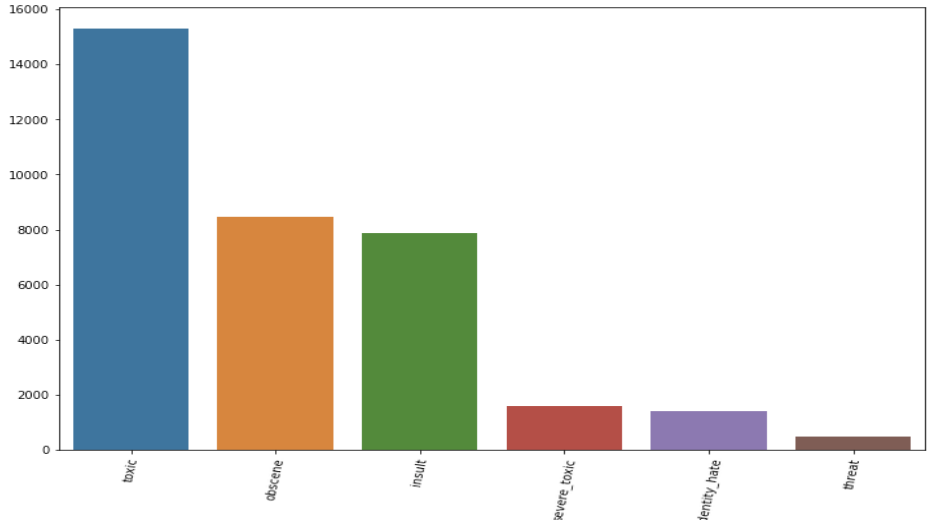
After training the model, analyzing the results of the 2 classifiers used in my research I went forward with logistic regression classifier and generated the desired result. The result file named submission file is created which has all the probabilities for the six multilabel for every comment in the test dataset.

Pipeline

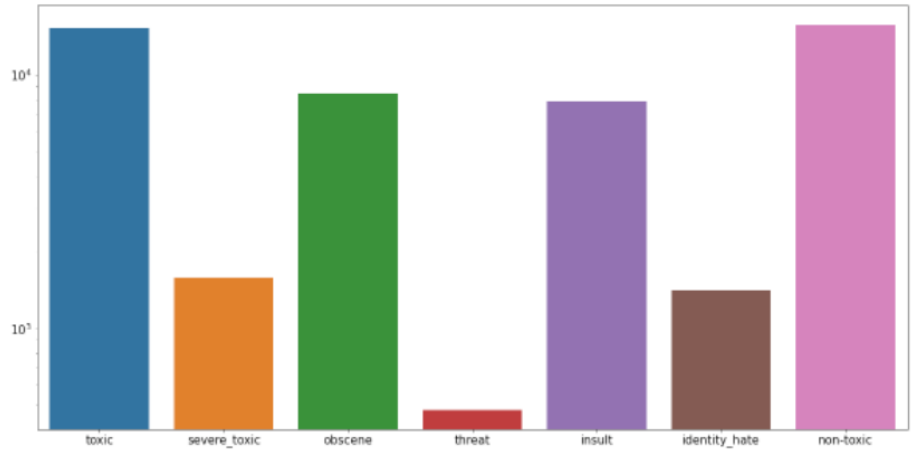
In pipeline I processed the data, trained, and transformed them. Later using OneVsRestClassifier for logistic regression, I have used the pipeline to fit the train data and predict the accuracy. I applied cross validation to find out the best hyperparameters of the dataset and predicted the results. I have done 5-fold cross validation which took much longer to execute and was able to find the best hyperparameters.

Results

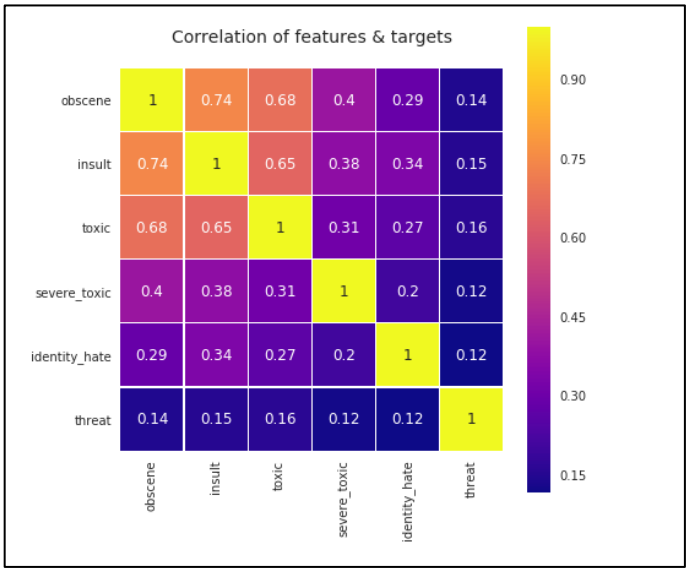
Distribution of types of toxic comments before balancing:



Distribution of types of toxic comments after balancing:



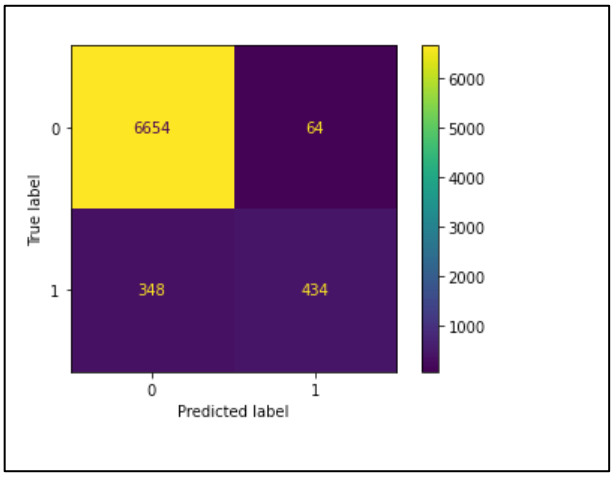
Correlation of features and targets: it looks like some of the labels are higher correlated, e.g. insult-obscene has the highest at 0.74, followed by toxic-obscene and toxic-insult.



Naive Bayes Classifier with method 1 – OneVsRestClassifier:

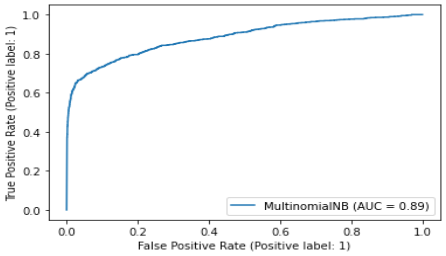
Score: 0.945066

Confusion matrix:



Naïve Bayes ROC AUC curve

```
nb_auc_score = roc_auc_score(nb_y_test, nb_model.predict_proba(nb_x_test)[:,1])
RocCurveDisplay.from_estimator(nb_model, nb_x_test, nb_y_test)
plt.show()
```



Naive Bayes Classifier with method 2 - Cross Validation

Displaying mean accuracy for all the class labels

```
DF_score = pd.DataFrame(index=class_names)
DF_score['NB'] = NB_score
DF_score
```

	NB
toxic	0.906136
severe_toxic	0.990004
obscene	0.947741
threat	0.997004
insult	0.950687
identity_hate	0.991195

Naive Bayes Classifier with method 3 Calculated separately for every label

```
CV Log_loss score for class toxic is -0.16741946553425982
CV Accuracy score for class toxic is 0.9400889855845976
CV ROC_AUC score 0.9335356718238577

[[28873    54]
 [ 1765  1223]]
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     28927
           1       0.96      0.41      0.57      2988

    accuracy                           0.94     31915
   macro avg       0.95      0.70      0.77     31915
weighted avg       0.94      0.94      0.93     31915

Training accuracy is 0.9464106661653193
Testing accuracy is 0.9430048566504778
```

```
CV Log_loss score for class severe_toxic is -0.04784264639320109
CV Accuracy score for class severe_toxic is 0.9898868831287437
CV ROC_AUC score 0.930375219921081

[[31611     1]
 [  301     2]]
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     31612
           1       0.67      0.01      0.01       303

    accuracy                           0.99     31915
   macro avg       0.83      0.50      0.50     31915
weighted avg       0.99      0.99      0.99     31915

Training accuracy is 0.9899573854734599
Testing accuracy is 0.9905373648754504
```

```
CV Log_loss score for class obscene is -0.11289916764149707
CV Accuracy score for class obscene is 0.9634094836789879
CV ROC_AUC score 0.9373264270890913

[[30266    25]
 [ 1050   574]]
              precision    recall  f1-score   support

           0       0.97      1.00      0.98     30291
           1       0.96      0.35      0.52      1624

    accuracy                           0.97     31915
   macro avg       0.96      0.68      0.75     31915
weighted avg       0.97      0.97      0.96     31915

Training accuracy is 0.9674750893025005
Testing accuracy is 0.9663167789440702
```

```
CV Log_loss score for class threat is -0.025102692445182512
CV Accuracy score for class threat is 0.996944914456855
CV ROC_AUC score 0.850694597189491

[[31824     1]
 [   90     0]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     31825
           1       0.00      0.00      0.00        90

    accuracy                           1.00     31915
   macro avg       0.50      0.50      0.50     31915
weighted avg       0.99      1.00      1.00     31915

Training accuracy is 0.9969527480102777
Testing accuracy is 0.9971486761710794
```

```
CV Log_loss score for class insult is -0.12169665685482775
CV Accuracy score for class insult is 0.9599157102270445
CV ROC_AUC score 0.9293513240956378

[[30349    63]
 [ 1117   386]]
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     30412
           1       0.86      0.26      0.40      1503

    accuracy                           0.96     31915
   macro avg       0.91      0.63      0.69     31915
weighted avg       0.96      0.96      0.95     31915

Training accuracy is 0.9634799774393683
Testing accuracy is 0.9630267899107003
```

```
CV Log_loss score for class identity_hate is -0.050548799592071744
CV Accuracy score for class identity_hate is 0.9911089177141804
CV ROC_AUC score 0.8770641049252337

[[31637     1]
 [  277     0]]
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     31638
           1       0.00      0.00      0.00       277

    accuracy                           0.99     31915
   macro avg       0.50      0.50      0.50     31915
weighted avg       0.98      0.99      0.99     31915

Training accuracy is 0.9911559190323995
Testing accuracy is 0.9912893623687921
```

Classification – 2 Logistic Regression Classifier with method 1 – OneVsRestClassifier

```
Time Taken: 329.56 seconds
ROC AUC Score Train: 0.9999177375137026
ROC AUC Score Test: 0.9799509159902606
```

Confusion matrix:

Toxic:

[[28766 161]

[1235 1753]]

Severe Toxic:

[[31558 54]

[234 69]]

Obscene:

[[30209 82]

[598 1026]]

Threat:
[[31815 10]
[76 14]]

Insult:
[[30235 177]
[755 748]]

Identity Hate:
[[31617 21]
[233 44]]

Toxic Classification report:

	precision	recall	f1-score	support
0	0.96	0.99	0.98	28927
1	0.92	0.59	0.72	2988
avg / total	0.95	0.96	0.95	31915

Severe Toxic:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	31612
1	0.56	0.23	0.32	303
avg / total	0.99	0.99	0.99	31915

Obscene:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	30291
1	0.93	0.63	0.75	1624
avg / total	0.98	0.98	0.98	31915

Threat:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	31825
1	0.58	0.16	0.25	90
avg / total	1.00	1.00	1.00	31915

Insult:

	precision	recall	f1-score	support
0	0.98	0.99	0.98	30412
1	0.81	0.50	0.62	1503

avg / total	0.97	0.97	0.97	31915

Identity Hate:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	31638
1	0.68	0.16	0.26	277
avg / total	0.99	0.99	0.99	31915

Logistic Regression Classifier with method 2 Calculated separately for every label

CV Log_loss score for class **toxic** is -0.11565056788653363

CV ROC_AUC score 0.9655633400034033

Training accuracy is 0.9989738045998622

Testing accuracy is 0.9621494595018016

CV Log_loss score for class **severe_toxic** is -0.028702235760378142

CV ROC_AUC score 0.9819670455594212

Training accuracy is 0.9994516513129035

Testing accuracy is 0.990224032586558

CV Log_loss score for class **obscene** is -0.06353767904159087

CV ROC_AUC score 0.9825806365447279

Training accuracy is 0.999224478285392

Testing accuracy is 0.9808867303775655

CV Log_loss score for class **threat** is -0.009942427338906403

CV ROC_AUC score 0.9779689272933578

Training accuracy is 0.9999059973679263

Testing accuracy is 0.9975560081466395

CV Log_loss score for class **insult** is -0.0832688553036

CV ROC_AUC score 0.9710193252847193

Training accuracy is 0.9987466315723507

Testing accuracy is 0.9720820930596898

CV Log_loss score for class **identity_hate** is -0.025804571029573165

CV ROC_AUC score 0.9725885998717897

Training accuracy is 0.9997649934198157 Testing accuracy is 0.9926680244399185

Pipeline Approach:

Score before tuning:

```
preds = pipeline.predict(X_test)
np.mean(preds == y_test)
```

✓ 7.5s

```
toxic            0.957387
severe_toxic     0.991227
obscene          0.979069
threat           0.997337
insult           0.971205
identity_hate    0.992167
dtype: float64
```

After tunning with best parameters:

{'classifier__estimator__C': 1, 'classifier__estimator__penalty': 'l2',
'features__comment_text__tfidf__max_features': 5000,
'features__comment_text__tfidf__ngram_range': (1, 1)}

```
#refitting on entire training data using best settings
grid_result.refit

preds_0 = grid_result.predict(X_test)
probs_0 = grid_result.predict_proba(X_test)

np.mean(preds_0 == y_test)
```

✓ 7.4s

```
toxic            0.958013
severe_toxic     0.990913
obscene          0.979320
threat           0.997243
insult           0.971267
identity_hate    0.991947
dtype: float64
```

Classification Report:

Toxic Classification report:

	precision	recall	f1-score	support
0	0.96	0.99	0.98	28927
1	0.90	0.61	0.73	2988
accuracy			0.96	31915
macro avg	0.93	0.80	0.85	31915
weighted avg	0.96	0.96	0.95	31915

Severe Toxic:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	31612
1	0.59	0.26	0.36	303
accuracy			0.99	31915
macro avg	0.79	0.63	0.68	31915
weighted avg	0.99	0.99	0.99	31915

Obscene:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	30291
1	0.92	0.65	0.76	1624
accuracy			0.98	31915
macro avg	0.95	0.82	0.87	31915
weighted avg	0.98	0.98	0.98	31915

Threat:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	31825
1	0.63	0.13	0.22	90
accuracy			1.00	31915
macro avg	0.81	0.57	0.61	31915
weighted avg	1.00	1.00	1.00	31915

Insult:

	precision	recall	f1-score	support
0	0.98	0.99	0.99	30412
1	0.81	0.51	0.63	1503
accuracy			0.97	31915
macro avg	0.89	0.75	0.81	31915
weighted avg	0.97	0.97	0.97	31915

Identity Hate:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	31638
1	0.68	0.19	0.29	277
accuracy			0.99	31915
macro avg	0.83	0.59	0.64	31915
weighted avg	0.99	0.99	0.99	31915

Confusion Matrix:

```
Toxic Confusion Matrixs:
[[28734   193]
 [ 1167  1821]]

Severe Toxic:
[[31557    55]
 [  225    78]]

Obscene:
[[30199    92]
 [  576  1048]]
```

```
Threat:
[[31818     7]
 [   78    12]]

Insult:
[[30226   186]
 [  733   770]]

Identity Hate:
[[31613    25]
 [  225    52]]
```

Analysis

After careful Analysis, I have come to analysis that when I trained the data with naïve bayes and logistics regression, the accuracy was higher for logistics. Even if we consider the ROC AUC curve and percentage, logistic has higher i.e. 97.9%.
The number of True positive and True Negative for all the six labels was higher than then remaining two concluding the precision and recall to be improved than Naïve Bayes classifier. The f1 score for value 1 is good after changing the approach to split the data and balancing the data using stratified sampling approach.

Conclusion

According to my research, harmful or toxic remarks on social media have numerous negative consequences for society. The capacity to identify toxic comments quickly and reliably could give numerous benefits while reducing harm. In addition, the research has demonstrated that easily available techniques can be used to handle this problem. In our research, we found that a Logistic regression classifier improves classification significantly above a Naive Bayes-based baseline approach. Also, using pipeline approach has helped me understand using pipeline how we can make a continuous flow of all the steps in classifying. For future work, I would like to improve the solution by using LSTM solution and SVC and CNN methods. I believe these will improve the performance more than the current solution provided.