Data Mining Project (DSE 32347611)

Offensive Comment Detection

Project submitted in fulfillment of the requirements

for the Data Mining Course of

Third Year B. Sc.
in
Computer Science (Honors)

Submitted by

Roll No	Name of Students
214047	Om Gupta
214020	Ashish Durgapal

Under the guidance of

Dr. Megha Ummat

Department of Computer Science
SRI GURU GOBIND SINGH COLLEGE OF COMMERCE
University of Delhi, New Delhi, India – 110015

Semester VI - May, 2024

Acknowledgements

It is a great pleasure for us to express respect and deep sense of gratitude to our supervisor **Dr. Megha Ummat**, Associate Professor, Department of Computer Science, Sri Guru Gobind Singh College of Commerce, as a convenor of this course for her wisdom, vision, expertise, guidance, enthusiastic involvement and persistent encouragement during the planning and development of this work.

We also gratefully acknowledge her painstaking efforts in thoroughly going through and improving the manuscripts without which this work could not have been completed.

Place: Delhi

Date: May 1, 2024

Ashish Durgapal

Om Gupta

Department of Computer Science
Sri Guru Gobind Singh College of Commerce
University of Delhi, New Delhi, India – 110015

Certificate

This is to certify that the data mining project on **"Offensive Comment Detection"** *has been compiled by* **Ashish Durgapal (214020)** *and* **Om Gupta (214047)**, *the students of the 'Sri Guru Gobind Singh College of Commerce' under the course 'BSc. (H) Computer Science' of 'Semester VI'. It embodies the work done by them during semester VI of their course under the due supervision of* **Dr. Megha Ummat**.

Dr. Megha Ummat
Department of Computer Science
Sri Guru Gobind Singh College of Commerce

Place: Delhi
Date: May 1, 2024

Index

1. Problem Statement

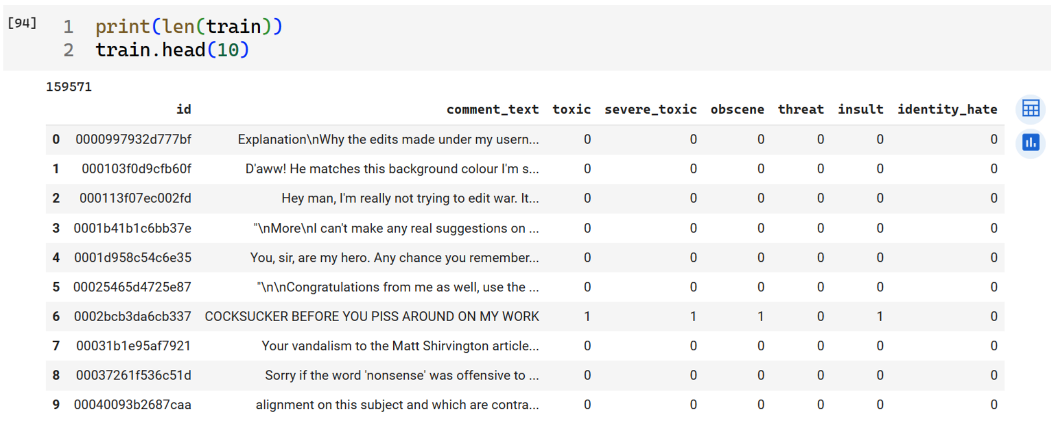
With the push towards declining data rates and the growth of telecommunication networks in the last few years, there has been a huge spike in the number of internet users. The penetration of high-bandwidth connections into the remotest parts of the world has enabled many users to come online and express their opinions and thoughts.

However, the other side of the coin is that it has led to an increase in instances of offensive speech and bullying, which not only inhibits certain users from freely expressing ingenious ideas but also spreads negativity, depression, communal/racial hatred, and, at times, can trigger riots. Therefore, it becomes imperative for social media ventures to ensure that their platform is free from toxicity to encourage the free flow of information and opinions.

The overall objective of the project is to ensure that hate speech/offensive comments are detected and classified, which can then be taken down.

2. Description of Dataset

The dataset for the problem has been publicly provided by Conversational AI [1]. In the dataset snapshot, approximately 160k training examples are visible, each associated with a set of labels that can be either 0 or 1, indicating whether a particular comment should belong to the class label or not. The dataset preparation involved the use of crowdsourced annotation platforms, leading to different users labeling different examples across the internet, resulting in subjective labeling due to individual perspectives on what constitutes offensive content. This subjectivity makes the dataset susceptible to judgment bias.



```
[94]
print(len(train))
train.head(10)
```

```
159571
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0
7	00031b1e95af7921	Your vandalism to the Matt Shirvington article...	0	0	0	0	0	0
8	00037261f536c51d	Sorry if the word 'nonsense' was offensive to ...	0	0	0	0	0	0
9	00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0

Figure. 1 Dataset

Further exploration of the dataset reveals that toxicity can be influenced by various factors, such as the rendering of comments. Depending on how a comment is displayed, it may appear as offensive symbols on certain screen widths, making it obviously toxic. However, if an annotator views the comment on a different screen size, they may not detect its toxicity, potentially misclassifying it as clean or spam. Additionally, some comments, despite being long and detailed expressions of opinions on various topics, have been classified as toxic without clear reasons.

Therefore, achieving 100% accuracy in classifying examples is impossible due to the inherent inaccuracies in some of the labels themselves.

The dataset contains a vast array of Wikipedia comments that have been manually labeled by human reviewers to identify instances of toxic behavior. These behaviors encompass a range of categories, including:

- **toxic:** Comments or posts that display general toxicity, including rude, disrespectful, or offensive language or content.
- **severe_toxic:** Comments or posts exhibiting extreme levels of toxicity, often containing highly offensive or malicious content intended to harm or provoke others.
- **obscene:** Comments or posts containing explicit or vulgar language, sexual content, or inappropriate material.
- **threat:** Comments or posts that include direct or implied threats of harm, violence, or intimidation towards individuals or groups.
- **insult:** Comments or posts intended to demean, belittle, or humiliate others through personal attacks or derogatory remarks.
- **identity_hate:** Comments or posts that target individuals or groups based on their race, ethnicity, religion, gender, sexual orientation, or other personal characteristics, often expressing hatred or discrimination.

3. Data Visualisation

Data Visualization is an important part of Data Mining. It helps us get visual insights about the dataset, such as some striking features or patterns in the dataset, which can help us choose the appropriate machine learning algorithms to apply.

We first plotted a treemap (Figure 2.a), showing the relative amount of hate tags present in the dataset, and found a massive class imbalance. "Toxic" tags were the highest, whereas tags labeled with "threat" had the lowest count. Also, plotting a bar graph (Figure 2.b) after adding a column for "clean" tags, it was found that a significantly large amount of comments were clean.

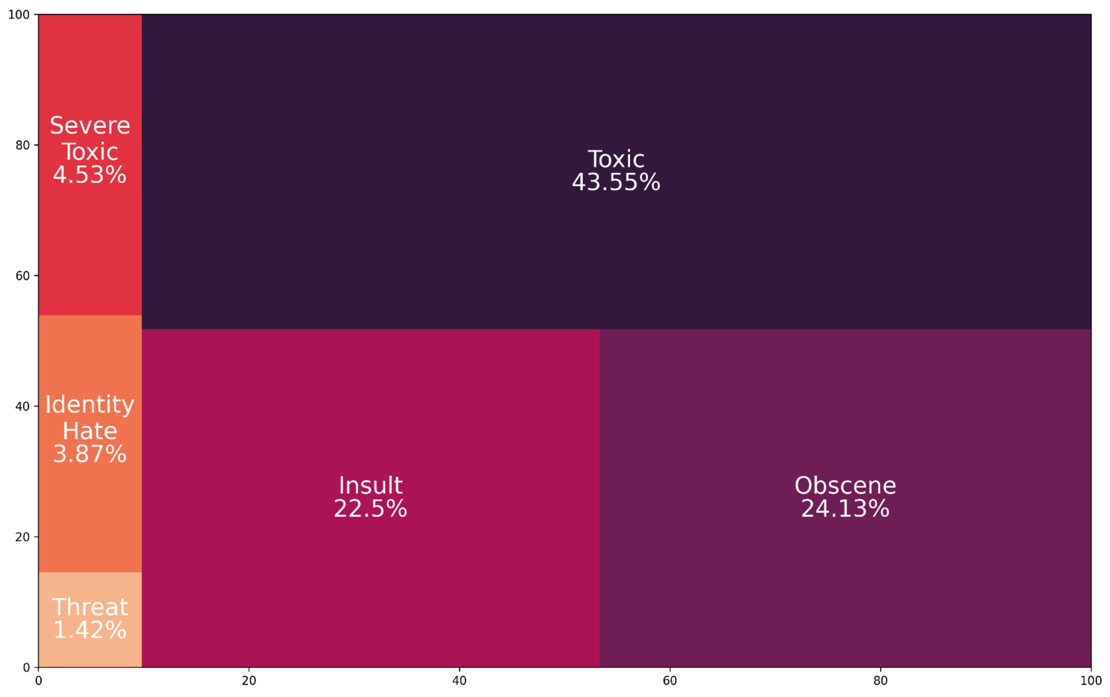


Figure 2 a) Tree map showing the composition of the dataset.

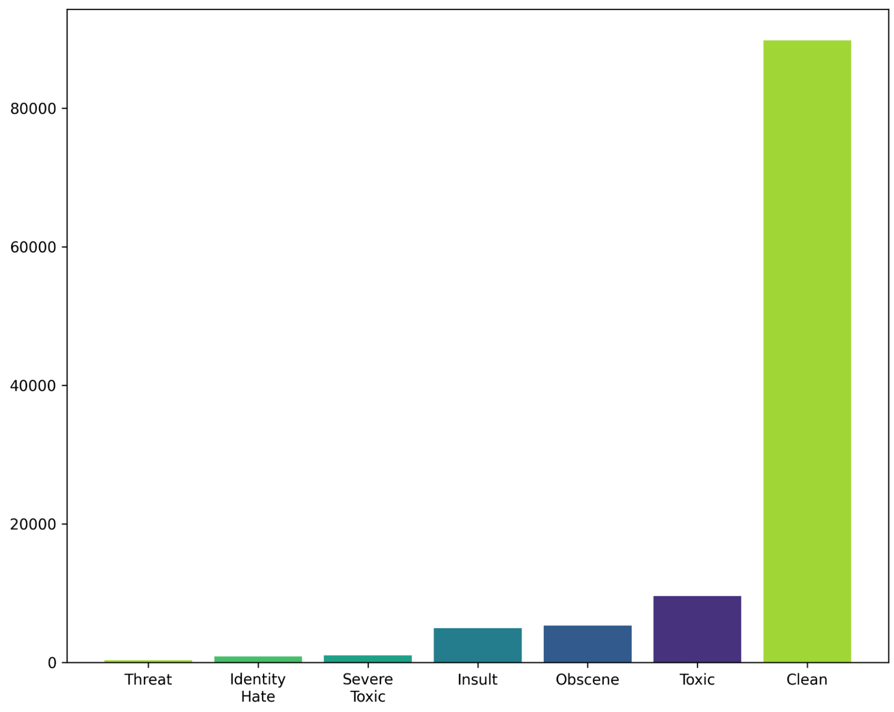
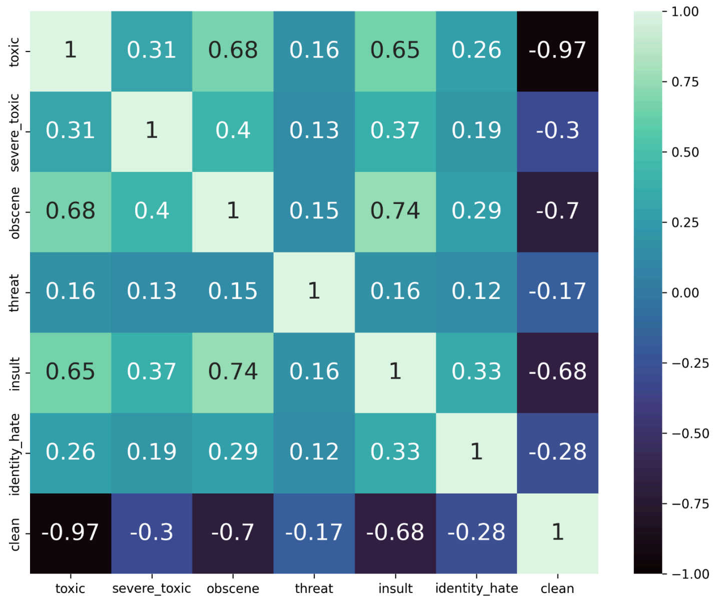


Figure 2 b) Bar Plot showing the class imbalance present in the dataset.

The heatmap (Figure 3) of the correlation between the tags show a negative correlation between the clean comments and the toxic comments. Also, a strong correlation can be observed between certain tags, such as toxic-obscene, and toxic-insult.



	toxic	severe_toxic	obscene	threat	insult	identity_hate	clean
toxic	1	0.31	0.68	0.16	0.65	0.26	-0.97
severe_toxic	0.31	1	0.4	0.13	0.37	0.19	-0.3
obscene	0.68	0.4	1	0.15	0.74	0.29	-0.7
threat	0.16	0.13	0.15	1	0.16	0.12	-0.17
insult	0.65	0.37	0.74	0.16	1	0.33	-0.68
identity_hate	0.26	0.19	0.29	0.12	0.33	1	-0.28
clean	-0.97	-0.3	-0.7	-0.17	-0.68	-0.28	1

Figure 3. Heatmap of the correlation between tags

4. Data Cleaning and Preprocessing

In this phase of the data mining process, the data undergoes cleaning to eliminate any anomalies or disturbances present in the dataset. If any data is absent, it either needs to be filled in manually or automatically, or the corresponding entry needs to be disregarded entirely. Fortunately, upon inspecting the dataset, it was found that none of the values were missing. Given that the dataset contains comments extracted from the internet, it may include elements such as URLs, IP addresses, user handles, and trailing newline marks which are shown in Figure 4, which are irrelevant for text classification purposes. Therefore, these elements are also removed from the dataset. Additionally, to ensure consistency, all letters are converted to lowercase since the case of the words is not significant.

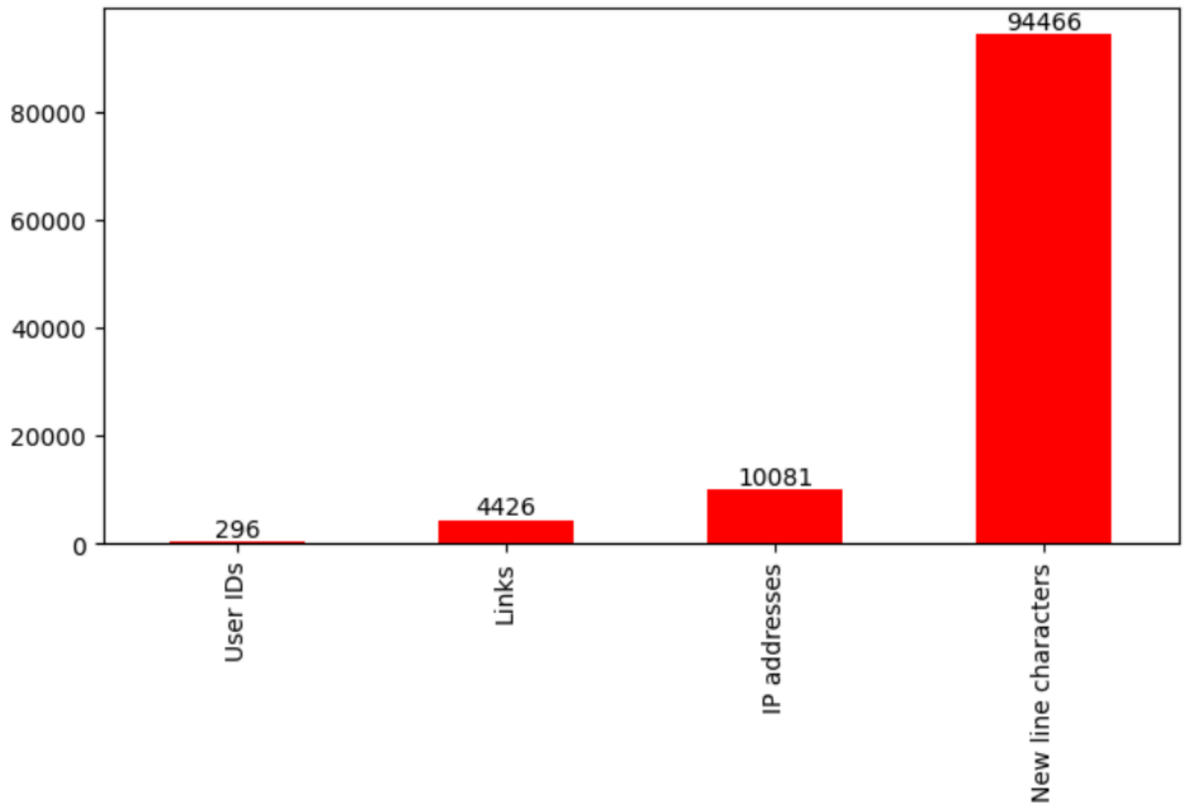


Figure 4. Bar Plot showing count of irrelevant strings in the data

Next, the cleaned data undergoes preprocessing to make it understandable for machines, enabling the application of machine learning algorithms for analysis. This preprocessing involves two key steps: getting rid of common stop words and applying either stemming or lemmatization to the comment text.

[83]
```python
from google.colab import drive
drive.mount('/content/drive')
```

[59]
```python
path='/content/drive/MyDrive/Colab Notebooks/Data_Mining/Offensive Comment Analysis/dataset'
```

[60]
```python
import pandas as pd
```

[61]
```python
train = pd.read_csv(path + "/train.csv")
test = pd.read_csv(path + "/test.csv")
```

[64]
```python
train.head()
len(train)
```

```
159571
```

[84]
```python
train = train[:100000]
train.describe()
```

	toxic	severe_toxic	obscene	threat	insult	identity_hate
count	100000.000000	100000.00000	100000.000000	100000.000000	100000.000000	100000.000000
mean	0.096120	0.00999	0.053270	0.003140	0.049670	0.008540
std	0.294757	0.09945	0.224573	0.055948	0.217263	0.092017
min	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.00000	1.000000	1.000000	1.000000	1.000000

[66]
```python
classes = list(train.columns)[2:]
classes
```

```
['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']
```

[67]
```python
train[classes].sum()
```

```
toxic            9612
severe_toxic      999
obscene          5327
threat            314
insult           4967
identity_hate     854
dtype: int64
```

[68]
```python
links = '(http://.*?\s)|(http://.*)'
ip_addr = '\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}'
users = '\[\[User.*'
newline = '\\n'
print(train['comment_text'].str.contains(links).sum(), 'Links')
print(train['comment_text'].str.contains(ip_addr).sum(), 'IP addresses')
print(train['comment_text'].str.contains(users).sum(), 'Users')
print(train['comment_text'].str.contains(newline).sum(), 'New line characters')
```

```
4426 Links
10081 IP addresses
296 Users
94466 New line characters
```

```
[69]
def clean(comment):
  import re
  comment=comment.lower()
  comment=re.sub(links,'',comment)
  comment=re.sub(ip_addr,'',comment)
  comment=re.sub(users,'',comment)
  comment=re.sub(newline,'',comment)
  return comment
```

```
[70]
train['comment_text']=train['comment_text'].map(lambda i:clean(i))
test['comment_text']=test['comment_text'].map(lambda i:clean(i))
```

```
[71]
print(train['comment_text'].str.contains(links).sum(), 'Links')
print(train['comment_text'].str.contains(ip_addr).sum(), 'IP addresses')
print(train['comment_text'].str.contains(users).sum(), 'Users')
print(train['comment_text'].str.contains(newline).sum(), 'New line characters')
```

```
<ipython-input-71-da3370f66566>:1: UserWarning: This pattern is interpreted as a regular expression, and has match groups. To actually get the groups,
  print(train['comment_text'].str.contains(links).sum(), 'Links')
0 Links
0 IP addresses
0 Users
0 New line characters
```

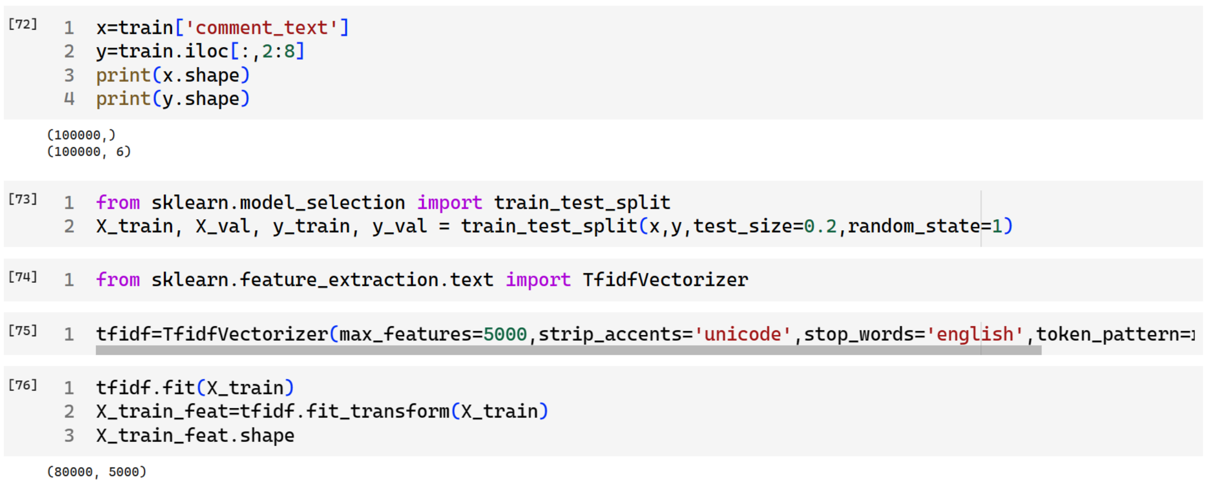
5. Data Classification

Data classification is the process of organizing and categorizing data into distinct groups or classes based on specific criteria or attributes. This enables easier management, retrieval, and analysis of data, as well as ensuring appropriate handling and protection of sensitive information.

5.1 Logistic Regression (LR)

In this approach, the multi-label classification problem is divided into separate single-class classification tasks, each focusing on predicting one label. For instance, if a comment can belong to six possible toxic categories, six individual classification tasks are created, each aimed at predicting one label. Subsequently, machine learning algorithms are applied separately to each task to obtain the predictions. Finally, the outcomes from all tasks are combined to determine all the labels for a given comment.

While this method offers a straightforward solution to multi-label classification, it has limitations. By treating each label prediction as an independent problem, it fails to consider any correlations between the labels. Consequently, if there are correlations present, this approach may yield suboptimal results.



```
[72]
x=train['comment_text']
y=train.iloc[:,2:8]
print(x.shape)
print(y.shape)
```

```
(100000,)
(100000, 6)
```

```
[73]
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(x,y,test_size=0.2,random_state=1)
```

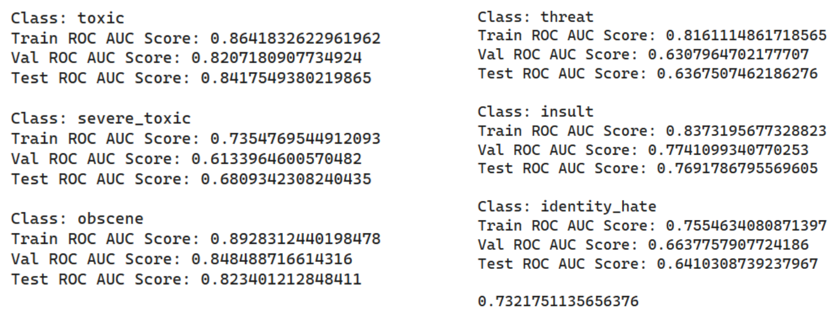
```
[74]
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
[75]
tfidf=TfidfVectorizer(max_features=5000,strip_accents='unicode',stop_words='english',token_pattern=
```

```
[76]
tfidf.fit(X_train)
X_train_feat=tfidf.fit_transform(X_train)
X_train_feat.shape
```

```
(80000, 5000)
```

[77]
```python
X_val_feat=tfidf.transform(X_val)
X_val_feat.shape
```
```
(20000, 5000)
```
[77]
```python

```
[78]
```python
import numpy as np

labels = pd.read_csv(path + "/test_labels.csv")
labels=labels.iloc[:,1:]
sum_labels=np.sum(labels,axis=1)
idx=sum_labels≥0
```
[79]
```python
labels_consider=labels[idx]
test=test.iloc[:,1:]
tests_consider=test[idx].values[:,0]

print(labels_consider.shape, tests_consider.shape)
```
```
(63978, 6) (63978,)
```
[80]
```python
X_test_feat=tfidf.transform(tests_consider)
X_test_feat.shape
```
```
(63978, 5000)
```
[80]
```python

```
[80]
```python

```
[81]
```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

model=LogisticRegression(C=20.0, max_iter=1000000)
```
[82]
```python
scores_roc_aoc = []

for label_name in classes:
    print('Class:', label_name)

    model.fit(X_train_feat, y_train[label_name])
    preds_train = model.predict(X_train_feat)
    train_roc_auc = roc_auc_score(y_train[label_name], preds_train)
    print('Train ROC AUC Score:', train_roc_auc)

    preds_val = model.predict(X_val_feat)
    val_roc_auc = roc_auc_score(y_val[label_name], preds_val)
    print('Val ROC AUC Score:', val_roc_auc)

    preds_test = model.predict(X_test_feat)
    test_roc_auc = roc_auc_score(labels_consider[label_name], preds_test)
    print('Test ROC AUC Score:', test_roc_auc)
    scores_roc_aoc.append(test_roc_auc)

    print()


print(np.mean(scores_roc_aoc))
```



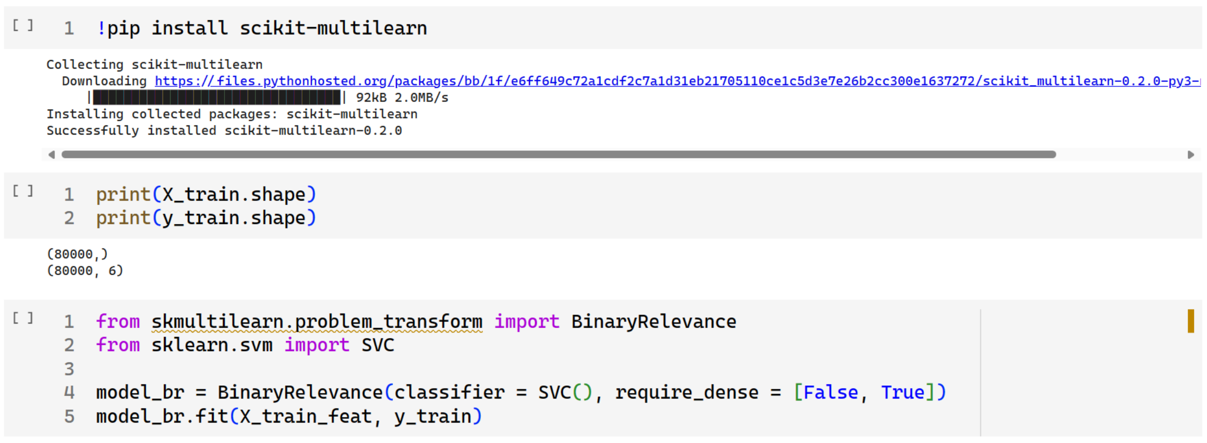
```
Class: toxic
Train ROC AUC Score: 0.8641832622961962
Val ROC AUC Score: 0.8207180907734924
Test ROC AUC Score: 0.8417549380219865

Class: severe_toxic
Train ROC AUC Score: 0.7354769544912093
Val ROC AUC Score: 0.6133964600570482
Test ROC AUC Score: 0.6809342308240435

Class: obscene
Train ROC AUC Score: 0.8928312440198478
Val ROC AUC Score: 0.848488716614316
Test ROC AUC Score: 0.823401212848411

Class: threat
Train ROC AUC Score: 0.8161114861718565
Val ROC AUC Score: 0.6307964702177707
Test ROC AUC Score: 0.6367507462186276

Class: insult
Train ROC AUC Score: 0.8373195677328823
Val ROC AUC Score: 0.7741099340770253
Test ROC AUC Score: 0.7691786795569605

Class: identity_hate
Train ROC AUC Score: 0.7554634080871397
Val ROC AUC Score: 0.6637757907724186
Test ROC AUC Score: 0.6410308739237967

0.7321751135656376
```

5.2 Support Vector Machines (SVM)

Support Vector Machines (SVMs) are supervised machine learning algorithms utilized for classification and regression tasks. They operate by constructing optimal hyperplanes in multi-dimensional space to separate data points into different classes. The goal is to maximize the margin, which represents the distance between the hyperplane and the closest data points, known as support vectors.

SVMs are versatile, capable of handling both linear and non-linear data. They excel in classifying high-dimensional data, particularly when the dimensionality exceeds the number of samples. This efficiency stems from their reliance on support vectors rather than the entire dataset for training, making them memory-efficient as well.

However, SVMs may struggle with noisy data or when classes overlap, as this can compromise their ability to find an optimal hyperplane. Despite this limitation, SVMs remain a powerful tool for various classification tasks.



```
[ ]  !pip install scikit-multilearn
```

```
Collecting scikit-multilearn
  Downloading https://files.pythonhosted.org/packages/bb/1f/e6ff649c72a1cdf2c7a1d31eb21705110ce1c5d3e7e26b2cc300e1637272/scikit_multilearn-0.2.0-py3-
     |████████████████████████████████| 92kB 2.0MB/s
Installing collected packages: scikit-multilearn
Successfully installed scikit-multilearn-0.2.0
```

```
[ ]  print(X_train.shape)
     print(y_train.shape)
```

```
(80000,)
(80000, 6)
```

```
[ ]  from skmultilearn.problem_transform import BinaryRelevance
     from sklearn.svm import SVC

     model_br = BinaryRelevance(classifier = SVC(), require_dense = [False, True])
     model_br.fit(X_train_feat, y_train)
```

```
BinaryRelevance(classifier=SVC(C=1.0, break_ties=False, cache_size=200,
                               class_weight=None, coef0=0.0,
                               decision_function_shape='ovr', degree=3,
                               gamma='scale', kernel='rbf', max_iter=-1,
                               probability=False, random_state=None,
                               shrinking=True, tol=0.001, verbose=False),
                require_dense=[False, True])
```

```
preds_train = model_br.predict(X_train_feat)
```

```
from sklearn.metrics import roc_auc_score, accuracy_score

print(roc_auc_score(y_train, preds_train.toarray()))
```

```
0.7689564894630726
```

```
preds_val = model_br.predict(X_val_feat)

```

```
print(roc_auc_score(y_val, preds_val.toarray()))
```

```
0.6665239237377513
```

```
preds_test = model_br.predict(X_test_feat)
```

```
print(roc_auc_score(labels_consider, preds_test.toarray()))
```

```
0.6636501540392014
```

```
from skmultilearn.problem_transform import ClassifierChain
from sklearn.svm import SVC

model_cc = ClassifierChain(classifier = SVC(), require_dense = [False, True])
model_cc.fit(X_train_feat, y_train)
```

```
ClassifierChain(classifier=SVC(C=1.0, break_ties=False, cache_size=200,
                               class_weight=None, coef0=0.0,
                               decision_function_shape='ovr', degree=3,
                               gamma='scale', kernel='rbf', max_iter=-1,
                               probability=False, random_state=None,
                               shrinking=True, tol=0.001, verbose=False),
                order=None, require_dense=[False, True])
```

```
preds_train = model_cc.predict(X_train_feat)

```

```
from sklearn.metrics import roc_auc_score, accuracy_score

print(roc_auc_score(y_train, preds_train.toarray()))
```

```
0.7645674561502386
```

```
preds_val = model_cc.predict(X_val_feat)

```

```
print(roc_auc_score(y_val, preds_val.toarray()))
```

```
0.6751504109269661
```

```
preds_test = model_cc.predict(X_test_feat)
```

```
print(roc_auc_score(labels_consider, preds_test.toarray()))
```

```
0.6718590534519624
```

6. Result and Analysis

6.1 Performance Measures

Precision-Recall or F1 score seem like the next obvious choice however they have their own share of limitations including selection of threshold value and relative importance to be given to precision vs recall. Hence we finally settled on the ROC curve and AUC score which give a very accurate picture of the performance of a discriminative model.

6.1.1 Receiver Operating Characteristic

A Receiver Operating Characteristic is a curve which plots True Positive Rate vs True Negative Rate. These two factors are an indicator of the model performance.

1. True Positive Rate (TPR):

$$TPR = \frac{TP}{TP+FN}$$

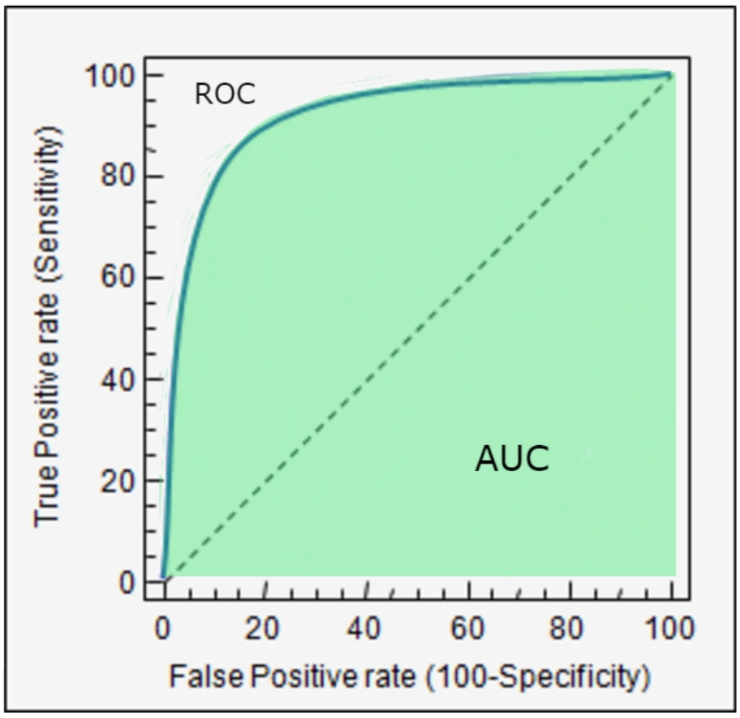
2. False Positive Rate (FPR):

$$FPR = \frac{FP}{FP+TN}$$

The aim of any discriminative model is to maximize the TPR while at the same time keeping the FPR as low as possible. The adjoining figure shows a perfect classifier and a random classifier.

6.1.2 Area Under Curve

AUC denotes the complete Area Under the ROC curve for the given domain. AUC values can range from 0 to 1. The idea of AUC stems from the observation that a better model will have more area under the ROC curve with a perfect model having AUC=1 and a model which always predicts incorrectly having AUC score=0. In this sense AUC can be understood as the average of performance measures of the classifier across all thresholds. Another interesting interpretation is that it expresses the probability that the model gives a higher positive score to a hate comment in our case as compared to a clean comment for any threshold value chosen.



ROC
AUC
True Positive rate (Sensitivity)
False Positive rate (100-Specificity)
0
20
40
60
80
100

Figure 5. AUC-ROC curve

6.2 Comparison Table

The mean AUC-ROC scores obtained from the different models used are mentioned in the table:

Model	Mean AUC_ROC score
Logistic Regression	0.73
Support Vector Machine	0.67

7. Conclusion

This study on detecting offensive comments has revealed that data mining can be a powerful tool in distinguishing between harmful and harmless speech. The evaluation of our models through the AUC-ROC scores indicates that LR, with a AUC-ROC score of **0.73**, is more effective at recognizing offensive comments than the SVM, which scored **0.67**. This finding highlights the importance of choosing the appropriate model for effectively moderating online content.

Although the results are encouraging, they also point to the possibility of further improvements. Both models show a good start, but there's room to grow. Future efforts could focus on exploring more sophisticated models or refining the current ones to improve their understanding of the subtleties of human language, which is often complex and nuanced.

In summary, this project has taken an important step towards making online interactions safer by effectively pinpointing and categorizing offensive remarks. As we continue to advance technologically, our methods for dealing with online negativity will also progress, aiming for digital communication that is reflective of human decency and collaborative spirit.

8. References

1. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data