CIS 600: Principles of social media & Data Mining

TERM PROJECT REPORT

Unmasking cyberbullying: Classifying comments and revealing communities

TEAM:

Sl. No.	Name	SUID	Email
1	Shivakumar Suresh	376148007	shsuresh@syr.edu
2	Sanjay Kumar Thovinakere Srinivas	761896187	sthovina@syr.edu
3	Sujay Vishwanath Malghan	314885101	svishwan@syr.edu
4	Rakshitha Kandavara Jayarama	862972326	rkandava@syr.edu
5	Chaitra Sampathram	970565905	csampath@syr.edu
6	Bindushree Huruhihakkalu, Ambikanath	375144255	bhuruhih@syr.edu
7	Deepika Nandan	927509909	dnandan@syr.edu
8	Riya Jomy Kannampuzha	303601108	rjk100@syr.edu

TABLE OF CONTENTS

1. PROBLEM DEFINITION

1.1 Project Overview

This project is titled **"Unmasking cyberbullying: Classifying comments and revealing communities" and aims** to build a machine learning model focused on classifying toxic comments and preventing online bullying. The dataset for this project will be extracted from Twitter using Twitter API (Application Programming Interface) and parsed using python libraries such as Tweepy and stored in a CSV file. This dataset will then be used to train a classification model that can detect different types of toxic comments, such as those related to identity-based hate, threats, insults, and obscenity. This model can also be used to analyze the type of toxic comments appearing online and help identify potential cases of online bullying or harassment. We use these data points to create a social network of size and then identify the common users among them. This will help us to conclude if the comments were made by a specific community which can then be associated with their agenda and eventually be reported. Libraries and tools such as scikit-learn, Keras, and TensorFlow will be used for building the machine learning model. Overall, this project aims to tackle the growing problem of online bullying and toxicity by providing a tool that can help identify and prevent toxic comments in real-time, thus creating a safer online environment for everyone.

1.2 Problem Statement

Cyberbullying is a growing issue on social media platforms, negatively impacting the mental health and well-being of victims. Despite the advancements in cyberbullying detection, there is still a need to develop a comprehensive framework that not only classifies comments but also reveals communities on social media platforms based on their behavior. The lack of a comprehensive framework makes it challenging to detect and prevent cyberbullying effectively. Therefore, the problem statement is to develop a comprehensive framework that can accurately classify instances of cyberbullying and reveal communities with varying degrees of cyberbullying prevalence to mitigate the impact of cyberbullying on social media platforms.

1.3 Significance of the project

There are several compelling reasons why we believe that this project is worthwhile and has the potential to make a valuable contribution to the field of cyberbullying research:

- **Addressing a critical social issue:** Cyberbullying is a serious social issue that can have significant impacts on individuals and communities. By focusing on this issue, our project has the potential to make a meaningful contribution to the field of social media and to society as a whole.

- **Leveraging technology to understand and combat cyberbullying:** Our project involves using social media and machine learning techniques to analyze instances of cyberbullying. This approach has the potential to provide insights that are difficult to obtain through traditional research methods and can help develop more effective strategies for preventing and mitigating cyberbullying.

- **Multidisciplinary nature:** This project is a multidisciplinary effort that brings together researchers from a variety of fields, including computer science, psychology, and sociology. This project can benefit from this interdisciplinary approach, as it can draw on insights and expertise from multiple fields to develop a more comprehensive understanding of social media and its impacts.

- **Potential for real-world impact:** Finally, our project has the potential to make a real-world impact by informing the development of interventions and strategies for addressing cyberbullying on social media. By identifying patterns and communities of cyberbullying, and by predicting its impact, this can help develop more effective interventions that can reduce the harm caused by cyberbullying.

2. DATA COLLECTION & ANALYSIS

2.1 Data Extraction

At the outset of our data collection process, we evaluated different options to access data from Twitter. After weighing the pros and cons of various methods, we decided to use the Twitter API in Python. This choice was motivated by the fact that the API provides a simple, yet powerful way to extract data from the platform.

One of the advantages of using the Twitter API is that it allows us to access data in real-time. This means that we can gather up-to-date information that is relevant to our research questions. Additionally, the API offers a variety of helpful features that make the data collection process more efficient. For example, we can filter data based on keywords, hashtags, or geographic location, which allows us to focus on specific types of content.

Once we had collected the data we needed, we stored it in a CSV file. This format is well-suited for storing and manipulating data because it is simple, flexible, and widely supported. One of the benefits of using a CSV file is that it can be opened and edited using a variety of software tools, including spreadsheet programs like Microsoft Excel or Google Sheets.

Furthermore, using a CSV file makes it easy to import the data into other programs or platforms. For instance, we can import the data into data visualization tools like Tableau or PowerBI to create informative and engaging visualizations. Additionally, we can use machine learning frameworks like scikit-learn or TensorFlow to train predictive models that can help us better understand patterns in the data.

Overall, using the Twitter API in Python and storing data in a CSV file proved to be an effective way to gather and manipulate data for our project.

To achieve this result, we performed the following steps: -

1. The authorization was established using the following commands:

```
# Authenticate with the Twitter API
auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(OAUTH_TOKEN, OAUTH_TOKEN_SECRET)
```

2. Next, we create an API object in the following manner:

```
# Create the API object
api = tweepy.API(auth)
0.1s
```

3. Later, we specify the username of the user from whom you wish to retrieve tweets and comments.

```python
username = 'thedemocrats'
tweets = api.user_timeline(screen_name=username, count=200)

# Get the next 200 tweets
next_tweets = api.user_timeline(screen_name=username, count=200, max_id=tweets[-1].id)
i=0
# Get the remaining tweets
while len(next_tweets) > 0:
    if(i==5):
      break
    i=i+1
    tweets += next_tweets
    try:
        next_tweets = api.user_timeline(screen_name=username, count=200, max_id=tweets[-1].id)
    except tweepy.TweepyException as e:
        if "Too Many Requests" in str(e):
            print("Rate limit exceeded. Waiting for 15 minutes.")
            time.sleep(15 * 60)  # Wait for 15 minutes
            next_tweets = api.user_timeline(screen_name=username, count=200, max_id=tweets[-1].id)
        else:
            print(e)

# Create a list to store the data
data = []
```

4. We are extracting the following valuable information from the Twitter stream.
   a) tweet_id
   b) tweet
   c) username

	tweet_id	tweet	user of the tweet
0	1649972417651982338	@TheDemocrats @JoeBiden YEP, WE REALLY NEED BE...	WHOAPATCH53
1	1649972186042732544	@TheDemocrats @JesusNarrowWay Get rid of DeSan...	ZeldaDan1
2	1649971468543959041	@TheDemocrats @JoeBiden You posted a picture o...	OldGardener137
3	1649971385316548611	@TheDemocrats @JoeBiden Fantastic. Why won't h...	geoffkryten
4	1649971035880722432	@TheDemocrats @JoeBiden Pedophile	stormindentist
...	...	...	...
16595	1649963156578476035	RT @CreasonJana: @TheDemocrats, with help from...	MAdamson2022
16596	1649962494021238789	@TheDemocrats @JoeBiden https://t.co/uCYc2Bou4l	BetaTomorrow
16597	1649962376870100998	@TheDemocrats @JoeBiden https://t.co/Y0s9kpJ8xl	DavidHa17134419
16598	1649962345139957761	@TheDemocrats @JoeBiden Oh yikes, look at the ...	DiforTruth
16599	1649962169893617666	@TheDemocrats @JoeBiden Meanwhile 👇\nhttps://t...	Bringit236

16600 rows × 3 columns

For the present project scope, we are utilizing the tweets from the CSV file to forecast the type of toxicity present.

2.2 Data Exploration

The dataset for training and testing for our project will be the Toxic Comment Classification Challenge [https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge] obtained

from the Kaggle platform. The file train.csv, compressed in the zip folder, is the training set, which contains comments with their binary labels. The file train.csv has a total of 151203 samples of comments and labeled data.

The dataset comprises of the following fields:

- **id**: An 8-digit integer value that identifies the author of the comment.
- **comment_text**: A multi-line text field that contains the original, unfiltered comment.
- **toxic:** A binary label that indicates whether the comment contains toxicity or not. It takes the value 0 for no toxicity and 1 for toxicity.
- **severe_toxic:** A binary label that indicates whether the comment contains severe toxicity or not. It takes the value 0 for no severe toxicity and 1 for severe toxicity.
- **obscene:** A binary label that indicates whether the comment contains obscenity or not. It takes the value 0 for no obscenity and 1 for obscenity.
- **threat:** A binary label that indicates whether the comment contains a threat or not. It takes the value 0 for no threat and 1 for a threat.
- **insult:** A binary label that indicates whether the comment contains an insult or not. It takes the value 0 for no insult and 1 for an insult.
- **identity_hate:** A binary label that indicates whether the comment contains identity-based hate or not. It takes the value 0 for no identity-based hate and 1 for identity-based hate.

These fields provide relevant information about each comment in the dataset, allowing for detailed analysis and modeling of toxicity and related phenomena.

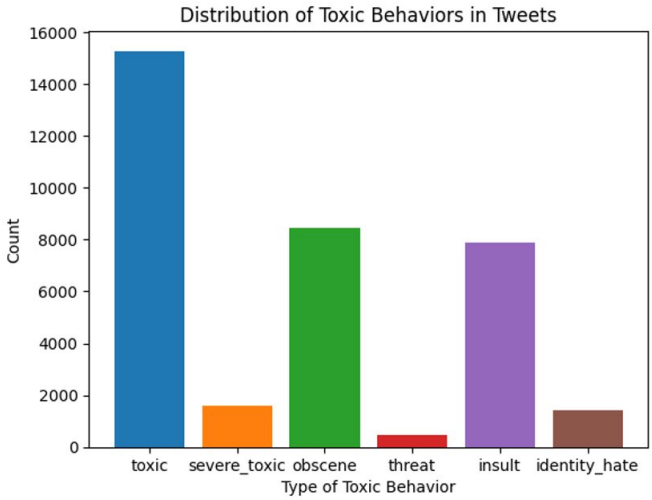
	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	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

The **"comment_text"** field is the primary focus of our analysis, as it will be preprocessed and used as input for various classification models to predict the presence of one or more of the outcome variables or labels, namely toxic, severe_toxic, obscene, threat, insult, and identity_hate. Our dataset contains 151,203 samples of comments along with their corresponding labels, which can be accessed by loading the "train.csv" file. The first five samples of the dataset are shown below:

Another important aspect of the dataset is the frequency of occurrence of multilabel data. It can be observed that 1 out of every 10 samples is toxic (9,000 samples), while 1 out of every 20 samples is both obscene and insulting (5,000 samples). However, the occurrence of severe toxic, threat, and identity hate samples is extremely rare, with only 800-900 out of 90,000 samples having these labels. In total, 9,790 samples have at least one label, and 5,957 samples have two or more labels.

2.3 Data Visualization

The distribution of tweets among the different categories of toxic behavior is an essential aspect to consider for our analysis. The data shows that toxic tweets are significantly more prevalent than tweets falling under the other categories. This observation highlights the importance of accurately identifying and addressing toxic behavior on social media platforms like Twitter.



The visualization reveals the number of tweets categorized under the six labels for toxic behavior, namely toxic, severe_toxic, obscene, threat, insult, and identity_hate, along with their corresponding tweet count. It can be observed that most of the tweets fall under the toxic category, with close to 15,000 tweets. This is followed by 8,500 obscene tweets and 7,500 insulting tweets. etc.

3. Methodology

3.1 Data Preprocessing

Data preprocessing includes the steps we need to follow to transform or encode data so that it may be easily parsed by the machine. The main requirement for a model to be accurate and precise in predictions is that the algorithm should be able to easily interpret the data's features.

We have performed the following preprocessing on the data:

1. **Prepare a string with all punctuation, URLs, and hashtags marks removed:**
   The string contains punctuation, hashtags and spaces for processing of the string we need to filter these strings to figure out if the sentences overall are toxic or not toxic.

2. **Using TfidfVectorizer(stop_words='english'):**
   The stop_words='english' parameter tells the TfidfVectorizer to automatically remove common English stop words (e.g., 'the', 'and' 'is', etc.) from the text data during preprocessing.

```
print (stop_words)
```

```
['a', 'about', 'above', 'after', 'again', 'against', 'all', 'am', 'an', 'and', 'any', 'are', "aren't", 'as', 'at', 'be',
'because', 'been', 'before', 'being', 'below', 'between', 'both', 'but', 'by', "can't", 'cannot', 'could', "couldn't", 'di
d', "didn't", 'do', 'does', "doesn't", 'doing', "don't", 'down', 'during', 'each', 'few', 'for', 'from', 'further', 'had',
"hadn't", 'has', "hasn't", 'have', "haven't", 'having', 'he', "he'd", "he'll", "he's", 'her', 'here', "here's", 'hers', 'h
erself', 'him', 'himself', 'his', 'how', "how's", 'i', "i'd", "i'll", "i'm", "i've", 'if', 'in', 'into', 'is', "isn't", 'i
t', "it's", 'its', 'itself', "let's", 'me', 'more', 'most', "mustn't", 'my', 'myself', 'no', 'nor', 'not', 'of', 'off', 'o
n', 'once', 'only', 'or', 'other', 'ought', 'our', 'ours', 'ourselves', 'out', 'over', 'own', 'same', "shan't", 'she', "sh
e'd", "she'll", "she's", 'should', "shouldn't", 'so', 'some', 'such', 'than', 'that', "that's", 'the', 'their', 'theirs',
'them', 'themselves', 'then', 'there', "there's", 'these', 'they', "they'd", "they'll", "they're", "they've", 'this', 'tho
se', 'through', 'to', 'too', 'under', 'until', 'up', 'very', 'was', "wasn't", 'we', "we'd", "we'll", "we're", "we've", 'we
re', "weren't", 'what', "what's", 'when', "when's", 'where', "where's", 'which', 'while', 'who', "who's", 'whom', 'why',
"why's", 'with', "won't", 'would', "wouldn't", 'you', "you'd", "you'll", "you're", "you've", 'your', 'yours', 'yourself',
'yourselves', '', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
'w', 'x', 'y', 'z']
```

3. **Splitting dataset into Training and Testing:**
   The dataframe["comment_text"] represents the feature data (in this case, the text comments) that is used as input for our model. The dataframe["target"] represents the target or label data (e.g., toxic value 1 or 0) that is used to predict the toxicity.

```
from sklearn.model_selection import train_test_split
X_train, X_test,Y_train, Y_test = train_test_split(dataframe["comment_text"],dataframe["target"] ,random_state=71, test_size=0.33, shuffle=True)
```

3.2 Implementation

We have implemented three ML (Machine Learning) models **Naïve Bayes**, **Support Vector Machine classifier** and **Random Forest classifier** using these models,

3.2.1 Naives Bayes Classifier

The Naive Bayes Classifier is a machine learning algorithm used for classification tasks, such as text classification. It is based on Bayes' theorem and the assumption of independence among predictors.

In the code snippet below, the Naive Bayes Classifier is implemented using the `**MultinomialNB**` class from the `**sklearn.naive_bayes**` module. This class is specifically designed for working with count-based data, such as text data.

The code first initializes an instance of `**MultinomialNB**`. Then it creates a pipeline using `**make_pipeline**`, which combines the vectorizer (presumably a text vectorizer such as `**TfidfVectorizer**`) and the classifier together. This allows for easy use of the entire pipeline for the training and testing phases. The pipeline is then fit to the training data using `**fit(X_train, Y_train)**`. This trains the classifier on the data. Once the model is trained, it is used to predict the classes of the testing data using `**predict(X_test)**`. The predicted labels are stored in `**y_pred**`.

Finally, the accuracy of the classifier is calculated using `**accuracy_score(Y_test, y_pred)**` and printed to the console. This gives an indication of how well the classifier is performing on the testing data.

```
import numpy as np
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.pipeline import make_pipeline
# Initialize classifier
nb_clf = MultinomialNB()
# Build the pipeline
nb_pipeline = make_pipeline(vectorizer, nb_clf)
# Fit the model to training data
nb_pipeline.fit(X_train, Y_train)
# Predict on testing data
y_pred = nb_pipeline.predict(X_test)
# Calculate accuracy score
accuracy = accuracy_score(Y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9170132361039898
```

3.2.2 Support Vector Machine Classifier

The key idea behind SVM is to transform the input data into a high-dimensional feature space using a kernel function, and then find the hyperplane that best separates the data points into different classes. The hyperplane is chosen to maximize the margin between the closest data points of different classes, thereby providing the best generalization performance for new data.

In the code below, we implemented a Support Vector Machine (SVM) classifier using the `**SVC**` class from the `**sklearn.svm**` module. The classifier is trained on text data using the `**TfidfVectorizer**` class from the `**sklearn.feature_extraction.text**` module.

It first imports necessary modules, including `**TruncatedSVD**` for dimensionality reduction and `Pipeline` to define the pipeline. The pipeline is defined by creating an instance of `Pipeline` and specifying a list of (name, transform) pairs. In this case, the transform steps are `**TfidfVectorizer**` for feature extraction, `**TruncatedSVD**` for dimensionality reduction, and `SVC` for classification. The `**TfidfVectorizer**` step converts text data into a matrix of TF-IDF features. The `**TruncatedSVD**` step then reduces the dimensionality of the feature matrix to `**n_components=300**`. Finally, the `SVC` step uses a linear kernel to classify the data.

Once the pipeline is defined, it is fitted to the training data using `**fit(X_train, Y_train)**`. This trains the classifier on the data. The pipeline is then evaluated on the test data by using the `**predict(X_test)**` method to predict the labels of the test data. The predicted labels are stored in `**y_pred**`. The accuracy of the classifier is then calculated using `**accuracy_score(Y_test, y_pred)**` and printed to the console.

SVM Classifier

```
from sklearn.decomposition import TruncatedSVD
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
svm_pipeline = Pipeline([
    ('tfidf', vectorizer),
    ('svd', TruncatedSVD(n_components=300)),
    ('svm', SVC(kernel='linear'))
])
svm_pipeline.fit(X_train, Y_train)
y_pred = svm_pipeline.predict(X_test)
accuracy = accuracy_score(Y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9413395620881521
```

3.2.3 Random Forest Classifier

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the class that is the mode of the classes of the individual trees. Each decision tree is constructed by randomly selecting a subset of features and training the tree on a subset of the training data. The main advantage of Random Forest is that it is less prone to overfitting compared to a single decision tree.

The code uses the `**TfidfVectorizer**` to transform the input text data into numerical features, and the `**RandomForestClassifier**` for classification. It first defines the pipeline with the vectorizer and classifier, then fits the pipeline to the training data, and finally makes predictions on the testing data. The accuracy score is calculated using the score () method of the pipeline.

Here's a brief overview of the below code:

1. Import the necessary modules: `**TfidfVectorizer**` to convert text data to a numerical form and `**RandomForestClassifier**` for building models.
2. Create an instance of `**RandomForestClassifier**` named `**rf_clf**`.
3. Define a pipeline for the Random Forest Classifier named `**rf_pipeline**` which includes a `**TfidfVectorizer**` to convert text data to numerical form, followed by `**RandomForestClassifier**`.
4. Train and fit the model on the training data using `**rf_pipeline.fit(X_train, Y_train)**`.
5. Make predictions on the test data using `**rf_pipeline.predict(X_test)**`.
6. Evaluate the accuracy of the model using `**rf_pipeline.score(X_test, Y_test)**`.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
rf_clf = RandomForestClassifier()
# Define pipeline for Random Forest
rf_pipeline = Pipeline([
    ('vect', vectorizer),
    ('clf', rf_clf)
])
rf_pipeline.fit(X_train, Y_train)
rf_prediction = rf_pipeline.predict(X_test)
rf_accuracy = rf_pipeline.score(X_test, Y_test)
print("Random Forest accuracy:", rf_accuracy)
```

```
Random Forest accuracy: 0.9530754476917526
```

4 RESULTS AND ANALYSIS

4.1 Hybrid Classifier

A hybrid classifier is a machine learning model which combines multiple classification algorithms aiming to improve accuracy and performance, instead of a single classifier alone.

To classify the comment, we make use of a hybrid classification technique. We use three classification models, Random Forest, Naive Bayes Classifier, and Support Vector Machine. If any of these models classifies a comment as toxic, we consider it toxic instead of taking the majority votes. Because the accuracy of each model is high, and the number of toxic comments in our data set is less.

The first model, Random Forest, is an ensemble learning method that builds multiple decision trees during training. It aggregates the results from these trees to determine the final classification. The second model, the Naive Bayes Classifier, is a probabilistic model that applies Bayes' theorem to predict the class of each tweet. It assumes that the features are independent, which, although not always true, can yield accurate results in practice. Lastly, the Support Vector Machine is a powerful model that aims to find the optimal hyperplane separating the data into two classes. It is effective in high-dimensional spaces and can handle non-linear relationships.

We ran through all three models to classify each tweet in our dataset and obtained predictions from each. This process will ensure that we consider each model's accuracy and minimize the recall. Next, we combine the predictions of each model. We are using Boolean 'OR' logic to combine the results.

Once the comments are classified, we create a new column in our data frame called "Predictions" and add the determined predictions to this column. This step allows us to store and analyze the classification results for each tweet in an organized manner.

We filter out only the rows corresponding to toxic tweets in the data frame to further analyze our results. From these rows, we extract the usernames associated with each toxic tweet. To obtain a list of unique usernames, we add these usernames to a set. This set helps us identify the users who have posted toxic content and eliminates any duplicate entries, providing us with a clear understanding of the distinct users involved in such activities.

Below is the code snippet for Hybrid classification.

```
inp_data = df["tweet"]
pred_1 = [1 if svm_pipeline.predict([i]) == [1] else 0 for i in inp_data]
pred_2 = [1 if rf_pipeline.predict([i]) == [1] else 0 for i in inp_data]
pred_3 = [1 if nb_pipeline.predict([i]) == [1] else 0 for i in inp_data]


n = len(pred_1)
combine_pred = []
for i in range(n):
    if pred_1[i] == 1 or pred_2[i] == 1 or pred_3[i] == 1:
        combine_pred.append(1)
    else:
        combine_pred.append(0)

```

Output:

	tweet_id	tweet	user of the tweet	Prediction
0	1651960825752502278	RT @Terry10661013: @TheDemocrats https://t.co/...	LynnRollings1	0
1	1651960815749136385	@TheDemocrats Did she borrow her pantsuits fro...	KBaker1668	0
2	1651960794827849730	@TheDemocrats I didn't think anyone could be m...	Chris05704668	0
3	1651960780097572867	@TheDemocrats This is the kind of person you v...	heavywater721	0
4	1651960752226402306	@TheDemocrats Horrible Concept	YogiGreg1	0
...	...	...	...	...

The Prediction column consists of binary values, 0 means non-toxic else 1 means toxic comment.

The Accuracy of **Naïve Bayes Classifier** is as follows:
Accuracy: 0.9182665831101996 ≈ **0.918 = 91.8% accuracy**
Recall: 0.22036969916636462 ≈ **0.22 = 22%**

```
1  nb_accuracy = accuracy_score(Y_test,nb_y_pred)
2  nb_recall = recall_score(Y_test,nb_y_pred)
3  nb_precision = precision_score(Y_test,nb_y_pred)
4  nb_f1 = f1_score(Y_test,nb_y_pred)
5  print("Naive Bayes Accuracy: ",nb_accuracy)
6  print("Naive Bayes Recall: ",nb_recall)
7  print("Naive Bayes Precision: ",nb_precision)
8  print("Naive Bayes F1 Score: ",nb_f1)
✓ 0.0s
```

```
Naive Bayes Accuracy:  0.9182665831101996
Naive Bayes Recall:  0.22036969916636462
Naive Bayes Precision:  0.9983579638752053
Naive Bayes F1 Score:  0.3610451306413301
```

The Accuracy of **Random Forest Classifier** is as follows:
Accuracy: 0.9548035473518297 ≈ **0.954 = 95.4% accuracy**
Recall: 0.6406306632837985 ≈ **0.64 = 64%**

```
1  rf_accuracy = accuracy_score(Y_test,rf_prediction)
2  rf_recall = recall_score(Y_test,rf_prediction)
3  rf_precision = precision_score(Y_test,rf_prediction)
4  rf_f1 = f1_score(Y_test,rf_prediction)
5  print("Random Forest Accuracy: ",rf_accuracy)
6  print("Random Forest Recall: ",rf_recall)
7  print("Random Forest Precision: ",rf_precision)
8  print("Random Forest F1 Score: ",rf_f1)
✓ 0.0s
```

```
Random Forest Accuracy:  0.9548035473518297
Random Forest Recall:  0.6406306632837985
Random Forest Precision:  0.8990335707019329
Random Forest F1 Score:  0.7481481481481482
```

The Accuracy of **SVM** is as follows:
Accuracy: 0.9417193642112459 ≈ **0.9417 = 94.17% accuracy**
Recall: 0.4797027908662559 ≈ **0.48 = 48%**

```
svm_accuracy = accuracy_score(Y_test,svm_y_pred)
svm_recall = recall_score(Y_test,svm_y_pred)
svm_precision = precision_score(Y_test,svm_y_pred)
svm_f1 = f1_score(Y_test,svm_y_pred)
print("SVM Accuracy: ",svm_accuracy)
print("SVM Recall: ",svm_recall)
print("SVM Precision: ",svm_precision)
print("SVM F1 Score: ",svm_f1)
```

✓ 0.0s

```
SVM Accuracy:  0.9417193642112459
SVM Recall:  0.4797027908662559
SVM Precision:  0.9304042179261863
SVM F1 Score:  0.6330264259237116
```

The **Accuracy of the Hybrid model** = (Accuracy of **Naïve Bayes Classifier**) ∪ (Accuracy of **Random Forest Classifier**) ∪ (Accuracy of **SVM**).

Our hybrid classifier is built in such a way that misclassification will happen only when all the models misclassify the data, The above Hybrid classifier can be treated as systems in parallel.

Then, theoretical accuracy is:

$$accuracy_{hybrid} = 1 - \prod_{ri \in accuracy\ set} (1 - r_i)$$

Then, theoretical recall is:

$$recall_{hybrid} = 1 - \prod_{k_i \in recall\ set} (1 - k_i)$$

Therefore,
Theoretical Accuracy of the Hybrid model = 1 - (1-0.954) (1-0.9417) (1-0.918) = 1- 0.0002199
Theoretical Accuracy of the Hybrid model = **0.99978 ≈ 99.98%**

Theoretical Recall of the Hybrid model = 1- (1-0.48) (1-0.64) (1-0.22) = 1- 0.146
Theoretical Recall of the Hybrid model = **0.854 = 85.4%**

Since we are focusing on the negative comments, we need to focus more on the recall instead of accuracy, considering theoretical, and the actual there is significant difference. However, considering the highest recall of Random Forest 64% but, hybrid model recall is 67.2%, there is a **3.2% increase in recall** which is good, the goal of hybrid model is achieved.

Actual score: of the hybrid model

```
Combined Accuracy:  0.9564936667995975
Combined Recall:  0.6719826023921711
Combined Precision:  0.8851754595368823
Combined F1 Score:  0.7639847532708356
```

4.2 Building Social Network (Community Detection)

4.2.1 K-Clique:

A maximum subgraph in which the greatest geodesic distance between any nodes <= k. To detect a community, first we select the users who have tweeted toxically, and then we build a social network using the reciprocal_friend relationship. which then can be used to detect the community using the K-Clique method (based on reachability).

We iterate through all the users who commented toxically, we collect their followers and find reciprocal relation if exists an edge is created, this process happens for at least a depth of 10. After all the iteration a graph is plotted,

Below is the code snippet for building social network.

```python
import networkx as nx
import matplotlib.pyplot as plt
import time

depth = 10

# Initialize the graph
G = nx.Graph()

# Add the nodes to the graph
for username in toxic_users_unique:
    try:
        user = api.get_user(screen_name=username)
    except tweepy.TweepyException as e:
        if "Too Many Requests" in str(e):
            print("Rate limit exceeded. Waiting for 15 minutes.")
            time.sleep(15 * 60)  # Wait for 15 minutes
            user = api.get_user(screen_name=username)
        else:
            print(e)
    G.add_node(username, name=user.name, color="blue")

# Add the edges to the graph
for d in range(1, depth+1):
    for username in toxic_users_unique:
        if G.nodes[username].get('depth') == d-1:
            try:
                user = api.get_user(screen_name=username)
                friends = api.get_friend_ids(screen_name=username)
                lookup_users = lookup_users_chunks(api=api,user_ids=friends)
                for target in lookup_users:
                    if G.has_node(target.screen_name):
                        G.add_edge(username, target.screen_name)
            except tweepy.TweepyException as e:
                if "Too Many Requests" in str(e):
                    print("Rate limit exceeded. Waiting for 15 minutes.")
                    time.sleep(15 * 60)  # Wait for 15 minutes
                    user = api.get_user(screen_name=username)
                    friends = api.get_friend_ids(screen_name=username)
                    lookup_users = lookup_users_chunks(api=api,user_ids=friends)
                else:
                    print(f"Error processing {username}: {str(e)}")
        G.nodes[username]['depth'] = d

# Draw the graph
node_colors = [node[1]['color'] for node in G.nodes(data=True)]
nx.draw(G, with_labels=True, node_color=node_colors)
plt.show()
```

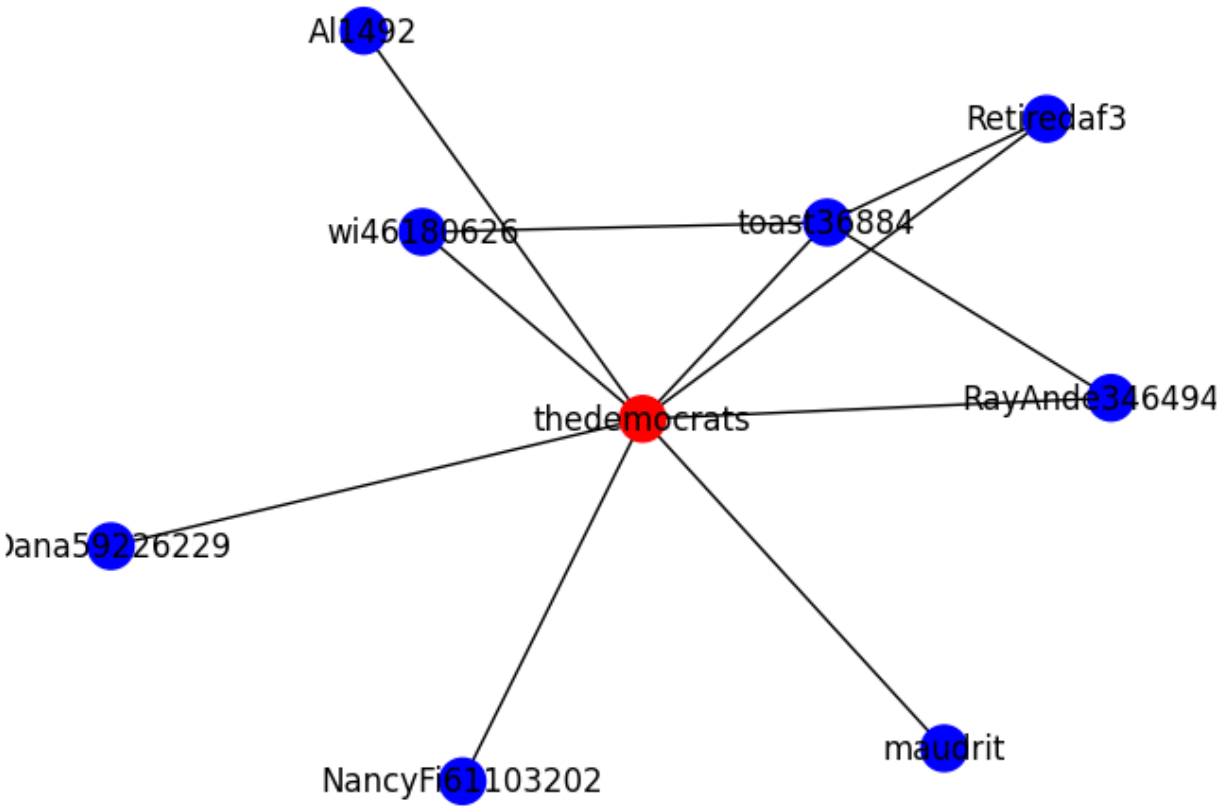
Output:

K-Clique graph:

Once the comments are classified as toxic, we construct a social network to better understand the relationships between these users. We employed the K-clique method to build this network to identify tightly-knit groups within larger social structures. By utilizing this method, we can effectively visualize the connections between users engaged in toxic behavior, offering valuable insights into the dynamics of their interactions.

The process begins by plotting users as nodes in the network. We then iterate through all the users and determine whether they are friends with others on the list based on their following and follower relationships. If users are found to be friends, we draw an edge between their respective nodes, thus establishing a connection. This technique enables us to identify groups of users who are interconnected through their toxic behavior on the platform, providing a deeper understanding of the patterns and relationships among these individuals. The resulting social network can be a powerful tool for further analysis and potential intervention strategies to mitigate toxic behavior on social media platforms.

Below is the social network for the user ‘the democrats’ and users who commented toxic in their tweets are identified, and a social network is formed to represent the community (2-clique graph).



4.3 Clustering (Community Detection)

4.3.1 K-Means:

K-Means clustering is an unsupervised machine learning technique aimed at creating ‘K’ clusters from the given data points.

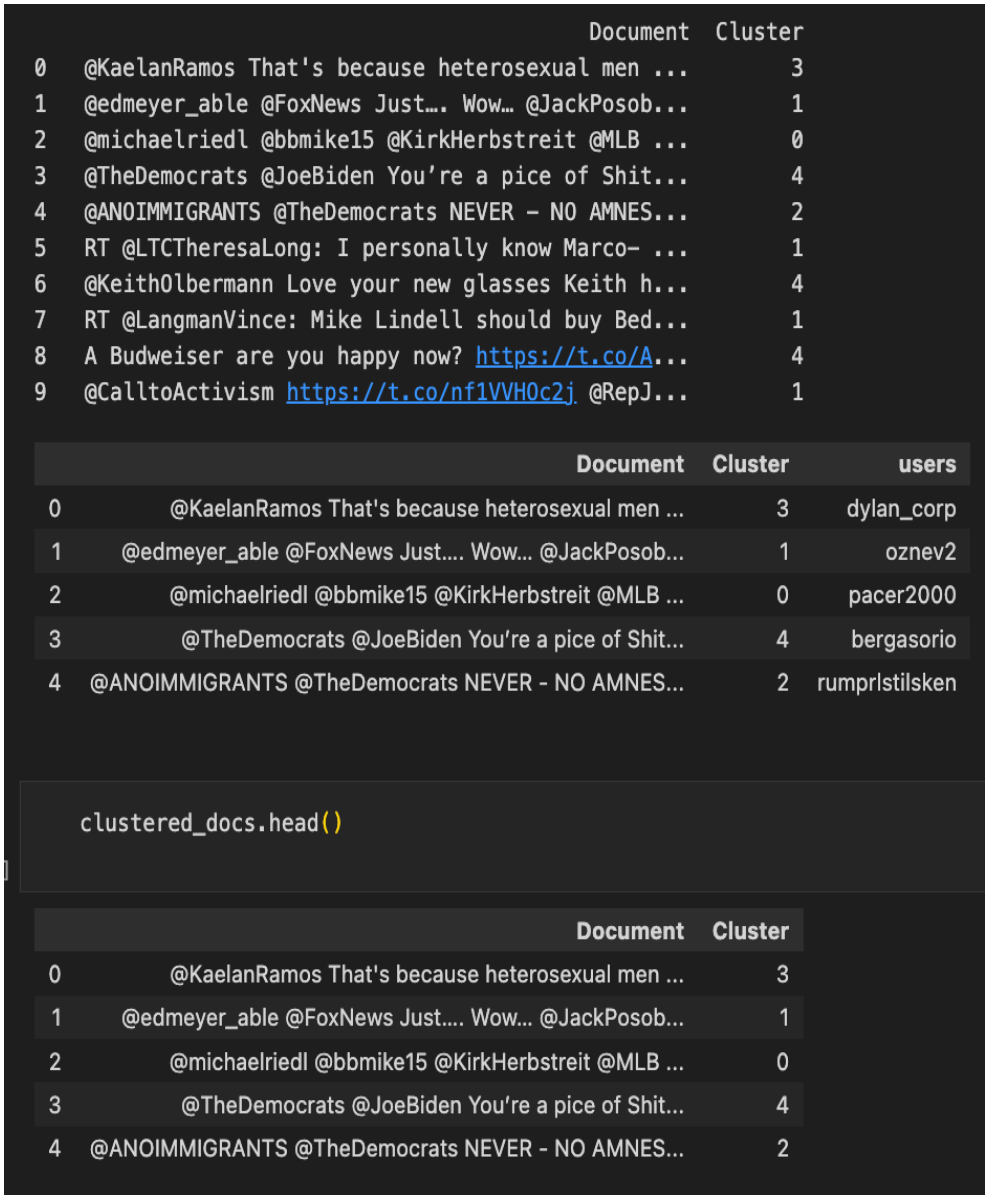
Clustering techniques play a significant role in social media analysis. This is because consider twitter, two users may not be having reciprocal friend relationship, but can share similar ideas and thoughts which can influence other users, similarly like community. In those cases, it is good to use clustering technique, which will help us to identify the clusters/community in the social media.

Here we make use of documents clustering technique using K-Means techniques. The most recent 100 tweets of a user are considered one document and then, we preprocess the documents, create a vector of words for each document using TF-IDF metric, then apply the document clustering technique.

We created clusters for K = 2-6 clusters out of which we found K=5 will produce optimal solutions. Out of five clusters, two clusters have been found to consist of more than two nodes (users/documents).

```python
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from nltk.corpus import stopwords

# all tweets of user is one document
documents = user_tweets.values()


# Preprocess the text
def preprocess(text):
    return text.lower()

preprocessed_documents = [preprocess(doc) for doc in documents]

import nltk
nltk.download('stopwords')

# Extract features using TF-IDF
vectorizer = TfidfVectorizer(stop_words=stopwords.words('english'))
X = vectorizer.fit_transform(preprocessed_documents)

# Determine the optimal number of clusters using silhouette score
# vary the range if needed
n_clusters_range = range(2, 6)
silhouette_scores = []

for n_clusters in n_clusters_range:
    kmeans = KMeans(n_clusters=n_clusters, init='k-means++', random_state=42)
    kmeans.fit(X)
    cluster_labels = kmeans.predict(X)
    silhouette_scores.append(silhouette_score(X, cluster_labels))

optimal_n_clusters = n_clusters_range[np.argmax(silhouette_scores)]

# Perform K-means clustering
kmeans = KMeans(n_clusters=optimal_n_clusters, init='k-means++', random_state=42)
kmeans.fit(X)

# Assign each document to a cluster
document_clusters = kmeans.predict(X)

# Display the results
clustered_docs = pd.DataFrame({"Document": documents, "Cluster": document_clusters})
print(clustered_docs)
```

Output:
Clusters are named {0, 1, 2, 3, 4}



	Document	Cluster
0	@KaelanRamos That's because heterosexual men ...	3
1	@edmeyer_able @FoxNews Just…. Wow… @JackPosob...	1
2	@michaelriedl @bbmike15 @KirkHerbstreit @MLB ...	0
3	@TheDemocrats @JoeBiden You're a pice of Shit...	4
4	@ANOIMMIGRANTS @TheDemocrats NEVER - NO AMNES...	2
5	RT @LTCTheresaLong: I personally know Marco- ...	1
6	@KeithOlbermann Love your new glasses Keith h...	4
7	RT @LangmanVince: Mike Lindell should buy Bed...	1
8	A Budweiser are you happy now? https://t.co/A...	4
9	@CalltoActivism https://t.co/nf1VVHOc2j @RepJ...	1

	Document	Cluster	users
0	@KaelanRamos That's because heterosexual men ...	3	dylan_corp
1	@edmeyer_able @FoxNews Just…. Wow… @JackPosob...	1	oznev2
2	@michaelriedl @bbmike15 @KirkHerbstreit @MLB ...	0	pacer2000
3	@TheDemocrats @JoeBiden You're a pice of Shit...	4	bergasorio
4	@ANOIMMIGRANTS @TheDemocrats NEVER - NO AMNES...	2	rumprlstilsken

```
clustered_docs.head()
```

	Document	Cluster
0	@KaelanRamos That's because heterosexual men ...	3
1	@edmeyer_able @FoxNews Just…. Wow… @JackPosob...	1
2	@michaelriedl @bbmike15 @KirkHerbstreit @MLB ...	0
3	@TheDemocrats @JoeBiden You're a pice of Shit...	4
4	@ANOIMMIGRANTS @TheDemocrats NEVER - NO AMNES...	2

Code Snippet to plot clusters in 3 dimensions using the dimensionality reduction technique.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.manifold import TSNE
from nltk.corpus import stopwords

# (Same dataset, preprocess function, feature extraction, and clustering code as before)

# Perform dimensionality reduction using t-SNE
tsne = TSNE(n_components=3, perplexity=5, random_state=42)
X_3d = tsne.fit_transform(X.toarray())

# Plot the clusters in 3D
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

colors = ['b', 'g', 'r', 'c', 'm', 'y', 'k']
markers = ['o', 'v', '^', '<', '>', 's', 'p']

for i in range(len(X_3d)):
    ax.scatter(X_3d[i, 0], X_3d[i, 1], X_3d[i, 2], s=100, c=colors[document_clusters[i]], marker=markers[document_clusters[i]])
    ax.text(X_3d[i, 0], X_3d[i, 1], X_3d[i, 2], unique_users[i], fontsize=12)

ax.set_title('K-means Clustering')
ax.set_xlabel('t-SNE Dimension 1')
ax.set_ylabel('t-SNE Dimension 2')
ax.set_zlabel('t-SNE Dimension 3')
plt.show()
```

Keywords in user cluster {'vegasdude71', 'bergasorio', 'GlennHinzman'}

```
{'joebiden', 'go', 'racist', 'years', 'likes', 'big', 'male', 'politicians', 'chinese', 'worl
d', '100', 'government', 'time', 'congress', 'idiot', 'problems', 'quit', 'border', 'money',
'gop', 'crack', 'biden', 'america', 'see', 'dead', 'place', 'better', 'https', 'woman', 'fami
lies', 'read', 'business', 'ban', 'china', 'always', 'piece', 'president', 'endwokeness', 'mu
ch', 'look', 'elonmusk', 'traitor', 'news', 'little', 'potus', 'country', 'back', 'day', 'mus
t', 'take', 'people', 'krassenstein', 'know', 'senschumer', 'like', 'hey', 'seems', 'george',
'rest', 'smoking', 'getting', 'house', 'ukraine', 'first', 'start', 'would', 'black', 'histor
y', 'class', 'got', 'us', 'term', 'get', 'co', 'thedemocrats', 'watch', 'ok', 'problem', 'dic
k', 'american', 'still', 'ass', 'inflation', 'poor', 'puppet', 'hunter', 'make', 'shit', 'ban
k', 'real', 'run', 'smart', 'criminal', 'want', 'bed', 'whitehouse', 'bill'}
```

Based on the above key words we can infer that the users in these clusters talk about politics and government related issues such as recession, crimes, bills etc. So, we can consider this group is focused on politics.

Keywords in user cluster {'oznev2', 'DiforTruth', 'chris_tondre', 'stormindentist'}

Based of the above key words we can infer that the users in these clusters talk about a wide variety of things, politics, government related issues, social concerns such as abortion, love, deaths, taxes etc. So, we can consider this group is focused on politics and social issues.

4.3.1 Visualizing clusters:

t-SNE

T-distributed stochastic neighbor embedding (t-SNE) is a machine-learning algorithm for data visualization and dimensionality reduction. This technique is used for high-dimension datasets if the starlight use of linear dimensionality techniques like SVD or PCA could be more effective. The advantage of using t-SNE is that it preserves the pairwise distances between the data points. It is achieved by minimizing the divergence between the probability distribution.t-SNE is widely used in fields such as bioinformatics, computer vision, and natural language processing and has successfully revealed complex patterns and relationships in large datasets.

We have used the t-SNE technique to plot the clusters in two and three dimensions ***(refer to 6.1 Visualization)***. We can observe that cluster points in 2D seem scattered, but if we consider 3D, they will form a group, so in the higher dimensions, the clusters will be more accurate, but it is hard to comprehend .

5. Challenges

1. The primary challenge faced is the limited access to the data from Twitter corpus due to the privacy concerns of the users. Multiple people on social media prefer not to disclose their data as prioritize their privacy. This is also one of the primary goals of these social media companies, i.e., to safeguard their user's privacy. This poses a challenging sector for the NLP models we train to detect cyberbullying taking place.
   This can be mitigated by the usage of anonymized data or obtaining consent from users to collect their data.

2. Secondly, the language and cultural barrier can pose a huge challenge. When cyberbullying is done in other languages other than English, it will be difficult for the NLP models to recognize it and act on it. It will not be identifiable by the NLP models as its trained on English language data. This can also be due to different languages or phrases or metaphors used in the tweet widely depending on the culture, context, and region. For example, the words used in one language can have multiple meanings in other languages and can lead to misunderstanding if one points towards Offensive language.
3.
4. Twitter has set a limit on the characters in each tweet, which may provide limited context to researchers in dissecting and analyzing the meaning or context of the tweet. Currently, it has a 280-c-character limit which can lead to users using shorthand or abbreviations to express their opinions on the platform. This in turn will make it more difficult for the NLP models to analyze to understand the full meaning of a tweet. Moreover, the intent of the tweet can be misunderstood as bullying or harassment if taken out of context. For addressing the challenge, we can collect more data on the user such as their history, activity on other social media platforms, and preferences to come to respective conclusions.

5. The Twitter API (tweepy) provided for researchers and developers has a major drawback. It is the restriction on the data collection process, where we can collect data only every 15 minutes. This limit on the frequency of data collection can pose a great challenge for developers who are attempting to develop a real-time cyberbully solution system as they will not be able to train the NLP models with sufficient data. It may be necessary to use other data sources or to develop more efficient data collection methods.

6. The definition of cyberbullying lacks clarity and uniformity across cultures and languages. What is categorized as bullying in one culture can be different in others. A few such examples are: Teasing and rude remarks are seen as normal and as a sign of friendly social interaction while some can take it as harassment or bullying. The sarcasm is processed varies as one may perceive it as a joke and others can mistake it as rudeness. This varied

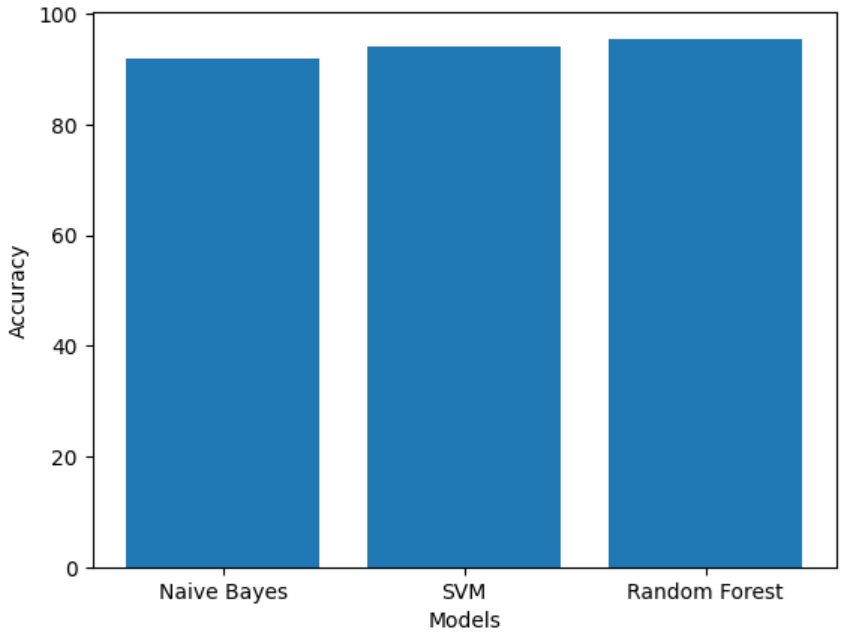
sentiment takes on such matters will pose a difficulty to develop NLP models of high accuracy and efficiency. To address this issue, we can consult cultural and language experts from varied backgrounds which can bolster the development of our NLP models and classifiers and to develop culturally sensitive algorithms.

Lastly, the challenge is for developing a rapid response solution required to deescalate the horrible side effects of cyberbullying. False news can spread quickly in social media which makes it necessary to take mitigation measures as soon as possible. Delayed or inadequate solutions can lead to severe consequences for the victim. Therefore, finding a solution to this is extremely critical and crucial. One feasible way to address this is to detect cyberbullying in real time and notify the appropriate authorities or organizations. For example, such activities can be flagged and brought to the attention of a human moderator.

6. CONCLUSION

6.1 Visualization

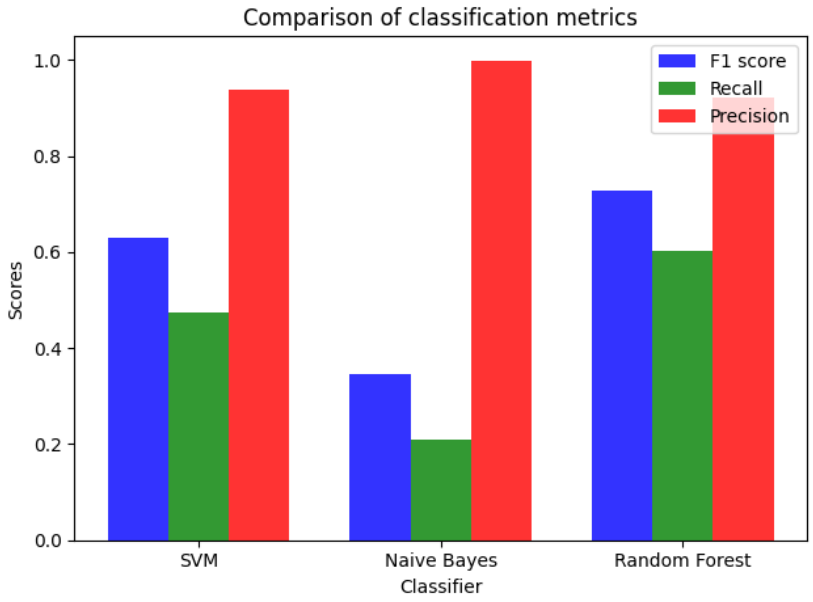
A comparison between the accuracies of the models:



Based on the above depiction of accuracies for the three models used to classify toxic tweets, we can infer that:

- The Random Forest model performs the best with an accuracy of 95.52%.
- The SVM model performs better than the Naive Bayes model with an accuracy of 94.17% compared to 91.82% respectively.
- The Random Forest model outperforms both the SVM and Naive Bayes models in terms of accuracy.

Now, as our dataset is a highly imbalanced one with only $1/8^{th}$ of the results being positive, we are aware that going by accuracy alone is not ideal to determine the best performing model. Hence, we have also computed the precision, recall and F1 score of each model to identify the best model.



From the given graph of F1 score, precision, and recall of the 3 models, we can infer that the random forest model is performing better than the other two models.

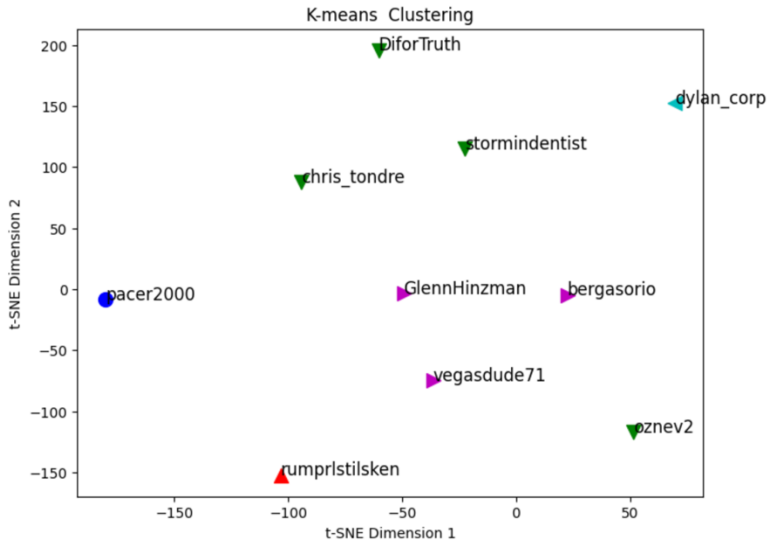
The F1 score of the random forest model is 0.728, which is higher than the F1 score of the SVM model (0.630) and the Naive Bayes model (0.345). The F1 score is a harmonic means of precision and recall, which gives equal importance to both metrics, and therefore provides an overall measure of the model's performance.

Similarly, the recall and precision values of the random forest model are higher than those of the SVM and Naive Bayes models. This means that the random forest model is better at correctly identifying true positive cases and avoiding false positives and false negatives, which are critical for classification tasks.

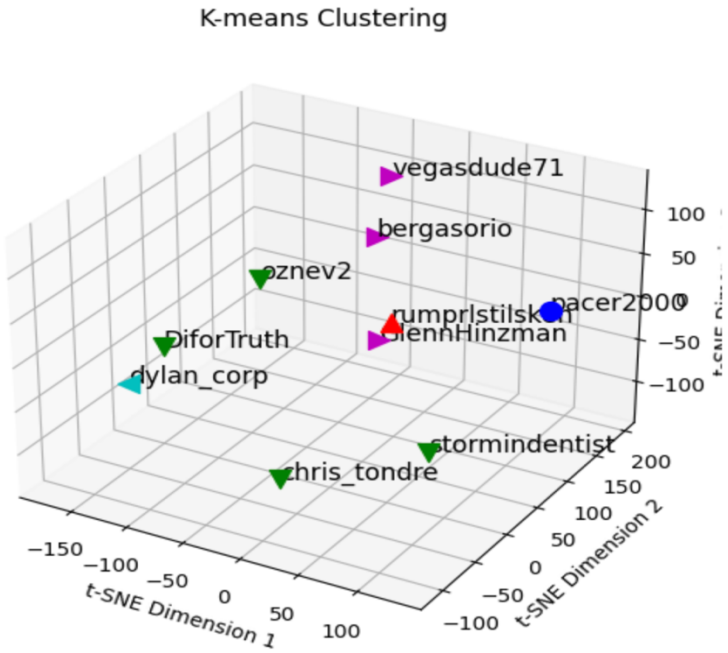
Therefore, based on these performance metrics, we can conclude that the random forest model is performing better than the other two models in this classification task.

In general, combining multiple classifiers can improve classification performance by leveraging the strengths of each individual model and reducing their weaknesses. Combining different models to create a hybrid classifier can lead to improved performance rather than using one classifier alone. Hence, we have coupled all three models to build a hybrid classifier. This will ensure that the tweets that go undetected by one classifier will be filtered by one of the other 2 classifiers.

2-D Visualization of Clustering for K = 5



3-D Visualization of Clustering for K = 5



In addition to classifying tweets, we have also employed the K-Means clustering algorithm to group the toxic users into clusters based on the similarities between their toxic comments. After performing repeated number of iterations, with different values of k, we have identified the value 5 to be the one with the best results. Hence, k is set to 5, that is, the toxic users are categorized into 5 clusters.

7. FUTURE SCOPE AND IMPROVEMENTS

7.1 Location-based Tracking

To further enhance the cyberbullying detection system, location-based services can be integrated. By utilizing IP address information, geolocation APIs, and user-provided location data, the proposed framework can identify the geographical locations of users involved in cyberbullying incidents. This information can help platform administrators and law enforcement agencies to take appropriate actions, such as issuing warnings or blocking access to the platform.

7.2 Other Platforms

The proposed framework can be tailored to suit various social media platforms by considering their unique features and modes of communication.

7.3 Real-time Detection

Developing algorithms to detect toxic tweets allows for faster response and a deeper understanding of the dynamics of the Twitter platform.

7.4 Optimization Algorithms

In the context of toxic tweets classification, evolutionary machine learning (ML) and metaheuristic optimization algorithms can still provide significant benefits when used alongside traditional classifiers like Naive Bayes, Random Forest, Decision Trees, and Support Vector Machines (SVM). These optimization algorithms can help improve classification performance by performing feature selection, hyperparameter tuning, model optimization, and ensemble learning specific to the task of detecting toxic tweets.

REFERENCES

1. V. Jain, V. Kumar, V. Pal and D. K. Vishwakarma, "Detection of Cyberbullying on Social Media Using Machine learning," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2021, pp. 1091-1096, doi: 10.1109/ICCMC51019.2021.9418254.

2. W. N. Hamiza Wan Ali, M. Mohd and F. Fauzi, "Cyberbullying Detection: An Overview," 2018 Cyber Resilience Conference (CRC), Putrajaya, Malaysia, 2018, pp. 1-3, doi: 10.1109/CR.2018.8626869.

3. N. Tsapatsoulis and V. Anastasopoulou, "Cyberbullies in Twitter: A focused review," 2019 14th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP), Larnaca, Cyprus, 2019, pp. 1-6, doi: 10.1109/SMAP.2019.8864918.

4. Shetty, J., Chaithali, K.N., Shetty, A.M., Varsha, B., Puthran, V. (2021). Cyber-Bullying Detection: A Comparative Analysis of Twitter Data. In: Chiplunkar, N., Fukao, T. (eds) Advances in Artificial Intelligence and Data Engineering. Advances in Intelligent Systems and Computing, vol 1133. Springer, Singapore. https://doi.org/10.1007/978-981-15-3514-7_62

5. Mukhopadhyay, D., Mishra, K., Mishra, K., Tiwari, L. (2021). Cyber Bullying Detection Based on Twitter Dataset. In: Joshi, A., Khosravy, M., Gupta, N. (eds) Machine Learning for Predictive Analysis. Lecture Notes in Networks and Systems, vol 141. Springer, Singapore

6. J. O. Atoum, "Cyberbullying Detection Through Sentiment Analysis," 2020 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2020, pp. 292-297, doi: 10.1109/CSCI51800.2020.00056.