| Name: Subrato Tapaswi | Class/Roll No.: D16AD/60 | Grade: |
|---|---|---|

➢ **Title of Experiment:** Mini Project Implementation

➢ **Topic:** Toxic Comments Classifier

## Introduction:

- In the realm of Natural Language Processing (NLP), toxic comment classification holds significant importance, as it addresses the critical task of identifying and flagging potentially harmful or offensive comments in online environments.
- The model aims to identify a toxic comment by giving a "toxicity score" to the comments.

## Problem Statement:

- Develop a robust toxic comment classification model in Python.
- Automatically identify and categorize offensive, harmful, or inappropriate content in usergenerated comments.
- Enhance online community safety by implementing a machine learning solution.
- Filter and flag toxic comments in real-time.

**Dataset:** A dataset of 54,313 unique tweets was found on Kaggle.

## Working of the Model:

- TF-IDF Vectorization: Convert text to numerical features using TF-IDF. Save TF-IDF vectorizer for future use.
- Split Data: Divide the dataset into training and testing sets.
- Model Creation: Build a Multinomial Naive Bayes model.
- Train the model with TF-IDF vectors.
- Model Evaluation: Assess model performance with ROC AUC score.
- Classification Testing: Test the model on sample toxic comments and display predictions.
- Model Persistence: Save the trained model and TFIDF vectorizer for future applications.

## ➢ Program:

## Data Pre-processing:

In [11]:
```
def prepare_text(text):
    def get_wordnet_pos(treebank_tag):
        if treebank_tag.startswith('J'):
            return wordnet.ADJ
        elif treebank_tag.startswith('V'):
            return wordnet.VERB
        elif treebank_tag.startswith('N'):
            return wordnet.NOUN
        elif treebank_tag.startswith('R'):
            return wordnet.ADV
        else:
            return wordnet.NOUN
    text = re.sub(r'[^a-zA-Z\']', ' ', text)
    text = text.split()
    text = ' '.join(text)
    text = word_tokenize(text)
    text = pos_tag(text)
    lemma = []
    for i in text: lemma.append(wordnet_lemmatizer.lemmatize(i[0], pos = get_wordnet_pos(i[1])))
    lemma = ' '.join(lemma)
    return lemma
```

In [12]:
```
data['clean_tweets'] = data['tweet'].apply(lambda x: prepare_text(x))
```

In [13]:
```
data.head(5)
```

Out[13]:

| | Toxicity | tweet | clean_tweets |
|---|---|---|---|
| 0 | 0 | @user when a father is dysfunctional and is s... | user when a father be dysfunctional and be so ... |
| 1 | 0 | @user @user thanks for #lyft credit i can't us... | user user thanks for lyft credit i ca n't use ... |
| 2 | 0 | bihday your majesty | bihday your majesty |
| 3 | 0 | #model i love u take with u all the time in ... | model i love u take with u all the time in ur |
| 4 | 0 | factsguide: society now #motivation | factsguide society now motivation |

## TF-IDF:

In [14]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
```

In [15]:
```
corpus = data['clean_tweets'].values.astype('U')
```

In [16]:
```
stopwords = set(nltk_stopwords.words('english'))
```

In [17]:
```
count_tf_idf = TfidfVectorizer(stop_words = stopwords)
tf_idf = count_tf_idf.fit_transform(corpus)
```

In [18]:
```
import pickle
```

In [19]:
```
pickle.dump(count_tf_idf, open("tf_idf.pkt", "wb"))
```

In [20]:
```
tf_idf_train, tf_idf_test, target_train, target_test = train_test_split(
    tf_idf, data['Toxicity'], test_size = 0.8, random_state= 42, shuffle=True
)
```

# Binary Classification Model:

```
In [21]: model_bayes = MultinomialNB()

In [22]: model_bayes = model_bayes.fit(tf_idf_train, target_train)

In [23]: y_pred_proba = model_bayes.predict_proba(tf_idf_test)[::, 1]

In [24]: y_pred_proba

Out[24]: array([0.90152453, 0.27916787, 0.79021827, ..., 0.09487729, 0.20555162,
                0.32090192])

In [25]: fpr, tpr, _ = roc_curve(target_test, y_pred_proba)

In [26]: final_roc_auc = roc_auc_score(target_test, y_pred_proba)
```

## ➢ Output:

```
In [27]: final_roc_auc

Out[27]: 0.9658691315317345

In [28]: test_text = "I hate you moron"
         test_tfidf = count_tf_idf.transform([test_text])
         display(model_bayes.predict_proba(test_tfidf))
         display(model_bayes.predict(test_tfidf))

         array([[0.39920068, 0.60079932]])

         array([1], dtype=int64)

In [30]: test_text = "you look ugly"
         test_tfidf = count_tf_idf.transform([test_text])
         display(model_bayes.predict_proba(test_tfidf))
         display(model_bayes.predict(test_tfidf))

         array([[0.24391577, 0.75608423]])

         array([1], dtype=int64)

In [31]: test_text = "you are fat"
         test_tfidf = count_tf_idf.transform([test_text])
         display(model_bayes.predict_proba(test_tfidf))
         display(model_bayes.predict(test_tfidf))

         array([[0.22187227, 0.77812773]])

         array([1], dtype=int64)
```

# Toxic Comment Classifier

```
you are fat
you are ugly
```

Classify

Toxicity Score: 0.90

V.E.S. INSTITUTE OF TECHNOLOGY
DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE

**NAME OF LAB TEACHER:** Dr. (Mrs.) Anjali Yeole

# Toxic Comments Classifier


- Surabhi Tambe- 59
- Subrato Tapaswi- 60
- Abhishek Thorat- 61

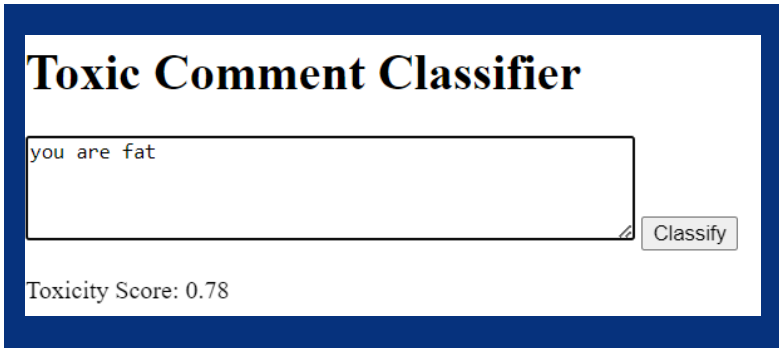

## INTRODUCTION

In the realm of Natural Language Processing (NLP), toxic comment classification holds significant importance, as it addresses the critical task of identifying and flagging potentially harmful or offensive comments in online environments.

The model aims to identify a toxic comment by giving a “toxicity score” to the comments

# PROBLEM STATEMENT

- Develop a robust toxic comment classification model in Python.
- Automatically identify and categorize offensive, harmful, or inappropriate content in user-generated comments.
- Enhance online community safety by implementing a machine learning solution.
- Filter and flag toxic comments in real-time.

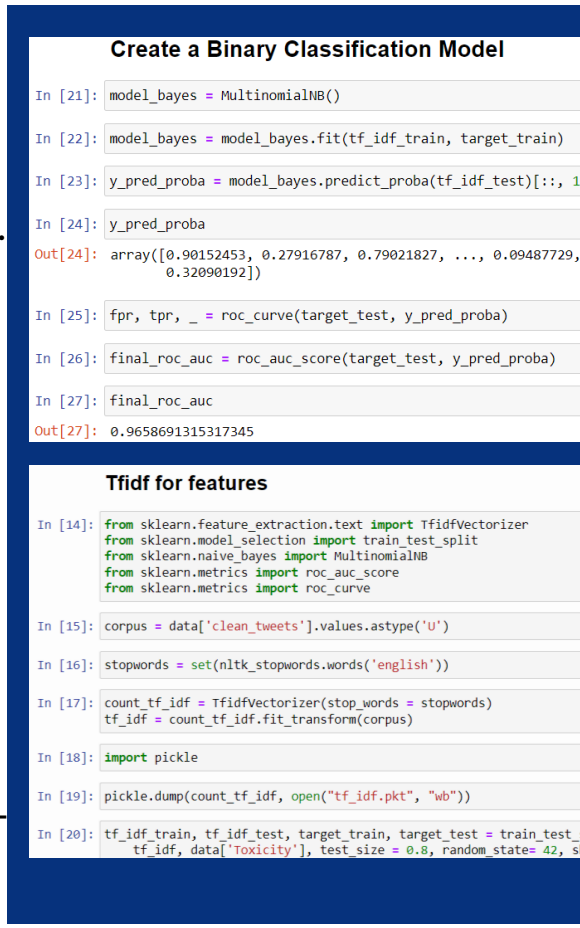



# DATASET

- We found this dataset of 54,313 unique tweets on **Kaggle.**
- A concise, balanced dataset using Tweets containing hate speech and offensive language to help build sentiment analysis/toxicity detection models.
- This dataset combines various original datasets to address class imbalance, preserving data integrity, and achieving a balanced class distribution for improved analysis and modeling.

## Working of the Model

- **TF-IDF Vectorization:**
  - Convert text to numerical features using TF-IDF.
  - Save TF-IDF vectorizer for future use.
- **Split Data:** Divide the dataset into training and testing sets.
- **Model Creation:**
  - Build a Multinomial Naive Bayes model.
  - Train the model with TF-IDF vectors.
- **Model Evaluation:** Assess model performance with ROC AUC score.
- **Classification Testing:** Test the model on sample toxic comments and display predictions.
- **Model Persistence:** Save the trained model and TF-IDF vectorizer for future applications.

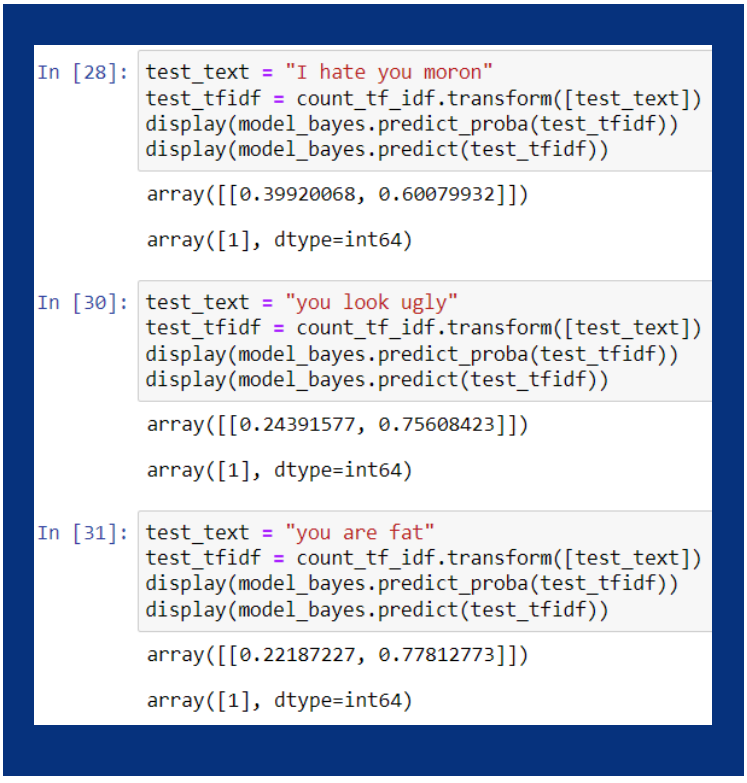


**Create a Binary Classification Model**

```
In [21]: model_bayes = MultinomialNB()

In [22]: model_bayes = model_bayes.fit(tf_idf_train, target_train)

In [23]: y_pred_proba = model_bayes.predict_proba(tf_idf_test)[::, 1

In [24]: y_pred_proba
Out[24]: array([0.90152453, 0.27916787, 0.79021827, ..., 0.09487729,
                0.32090192])

In [25]: fpr, tpr, _ = roc_curve(target_test, y_pred_proba)

In [26]: final_roc_auc = roc_auc_score(target_test, y_pred_proba)

In [27]: final_roc_auc
Out[27]: 0.9658691315317345
```

**Tfidf for features**

```
In [14]: from sklearn.feature_extraction.text import TfidfVectorizer
         from sklearn.model_selection import train_test_split
         from sklearn.naive_bayes import MultinomialNB
         from sklearn.metrics import roc_auc_score
         from sklearn.metrics import roc_curve

In [15]: corpus = data['clean_tweets'].values.astype('U')

In [16]: stopwords = set(nltk_stopwords.words('english'))

In [17]: count_tf_idf = TfidfVectorizer(stop_words = stopwords)
         tf_idf = count_tf_idf.fit_transform(corpus)

In [18]: import pickle

In [19]: pickle.dump(count_tf_idf, open("tf_idf.pkt", "wb"))

In [20]: tf_idf_train, tf_idf_test, target_train, target_test = train_test_
             tf_idf, data['Toxicity'], test_size = 0.8, random_state= 42, s
```



```
In [28]: test_text = "I hate you moron"
         test_tfidf = count_tf_idf.transform([test_text])
         display(model_bayes.predict_proba(test_tfidf))
         display(model_bayes.predict(test_tfidf))

         array([[0.39920068, 0.60079932]])

         array([1], dtype=int64)

In [30]: test_text = "you look ugly"
         test_tfidf = count_tf_idf.transform([test_text])
         display(model_bayes.predict_proba(test_tfidf))
         display(model_bayes.predict(test_tfidf))

         array([[0.24391577, 0.75608423]])

         array([1], dtype=int64)

In [31]: test_text = "you are fat"
         test_tfidf = count_tf_idf.transform([test_text])
         display(model_bayes.predict_proba(test_tfidf))
         display(model_bayes.predict(test_tfidf))

         array([[0.22187227, 0.77812773]])

         array([1], dtype=int64)
```

# CONCLUSION

- A final ROC AUC score of 0.9659 indicates that the model performs exceptionally well in distinguishing between toxic and non-toxic comments.
- This high score demonstrates the model's robustness and effectiveness in classifying toxic language, making it a valuable tool for content moderation and online safety efforts.

# THANK YOU

## SIMPLE FRONTEND

- We have created a simple frontend using basic **HTML** and **JavaScript.**
- We have integrated our model using **Flask.**
- The website has a “**comment**” area, in which the user can type any text, and upon clicking the “**classify**” button, it will give the “**toxicity**” score for it.