Cyberbullying Detection

Project report submitted to the Amrita Vishwa Vidyapeetham in partial fulfilment of the requirement for the Degree of

BACHELOR of TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Smrithi Venugopal.	AN.EN.U4CSE21154
Meenakshi Saju.	AN.EN.U4CSE21136
Lakshminandha K. S.	AN.EN.U4CSE21167

AMRITA SCHOOL OF COMPUTING

AMRITA VISHWA VIDYAPEETHAM
(Estd. U/S 3 of the UGC Act 1956)

AMRITAPURI CAMPUS

KOLLAM - 690525

June 2024

Abstract

Cyberbullying has emerged as a pervasive issue in the digital age, affecting countless individuals across various online platforms. This form of online abuse instills fear and anxiety, making it crucial to develop effective methods for detecting and mitigating such behavior. Our project delves into the application of machine learning (ML), deep learning (DL), and natural language processing (NLP) techniques to identify instances of cyberbullying automatically. We explored a comprehensive range of algorithms, from traditional ML models like Support Vector Machines (SVM) and Random Forests to advanced DL architectures such as Artificial Neural Networks (ANN), Bi-GRU, LSTM, XLSTM, and Bi-LSTM. Additionally, we employed NLP methods involving tokenization and text classification.

Our study highlights the efficacy of these approaches in detecting abusive language, harassment, and other forms of cyberbullying in text data. We conducted a comparative analysis of the performance of these methods on a balanced dataset, addressing a common issue in existing research where highly imbalanced datasets often lead to biased results. The findings of our project demonstrate that integrating ML, DL, and NLP techniques can significantly improve the accuracy and robustness of cyberbullying detection systems. This comprehensive approach paves the way for more sophisticated and reliable automated solutions in online content moderation, fostering a safer and more positive online environment. Our research underscores the potential of these technologies in creating effective tools to combat cyberbullying and protect individuals from online abuse.

Table of Contents

SOFTWARE ENGINEERING PHASES

Chapter 1: Communication

1.1. Client Definition

The client for the Cyberbullying Detection Project includes social media platforms, online forums, educational institutions, and digital communities. The project aims to provide an automated system to detect and mitigate cyberbullying, ensuring user safety. These organizations define the project's objectives, ensuring it meets their standards and strategic goals of promoting a secure and positive online environment.

1.2. Meeting Minutes

Project Initiation Meeting (18/03/2024)

- ➢ Attendees: Project Team Members
- ➢ Agenda: Introduction to the project, setting project objectives, and initial brainstorming session.
- ➢ Discussion Points:
    - Outlined primary objectives.
    - Assigned initial research tasks.

Initial Meeting (22/03/2024)

- ➢ Attendees: Project team, Faculty Advisor (Ms. Anjali T S)
- ➢ Agenda: Introduction to the project, understanding client requirements, and setting expectations.
- ➢ Discussion Points:
    - Relevance of a cyberbullying detection system

Idea Discussion (29/03/2024)

- ➢ Attendees: Project Team Members
- ➢ Agenda: Discuss initial ideas and potential solutions.

- ➢ Discussion Points:
    - Coming up with potential problem statements
    - Allocation of tasks for detailed research.

Idea Confirmation Meeting (05/04/2024)

- ➢ Attendees: Project Team, Faculty Advisor (Ms. Athira Joshi)
- ➢ Agenda: Review of idea
- ➢ Discussion Points:
    - Feedback on idea and its scope
    - Confirmation of feasibility

Follow-up meeting (12/04/2024)

- ➢ Discussing gathered data from research papers
- ➢ Finalized on innovation
- ➢ Dataset gathering

Kickoff Meeting (19/04/2024)

- ➢ Attendees: Project Team
- ➢ Agenda: Commencement of the coding phase(starting with dataset preparation)
- ➢ Discussion Points:
    - Review of the development plan
    - Allocation of coding tasks

Progress Check-in (26/04/2024)

- ➢ Attendees: Project Team Members
- ➢ Agenda: Assessing initial coding progress.
- ➢ Discussion Points:
    - Review of completed tasks.
    - Discussion of encountered challenges.
    - Adjustment of task priorities.

Implementation Discussion Meeting (3/05/2024)

- ➢ Attendees: Project Team Members
- ➢ Agenda: Detailed review of implementation of work.

- ➢ Discussion Points:
  - ● Discussed the results of various implementations of ML and DL algorithms
  - ● Discussed new algorithms that could be added for better results

Sprint Review 1 (14/05/2024)

- ➢ Attendees: Project Team Members, Ms Anjali T S
- ➢ Agenda:Review of work done so far
- ➢ Discussion Points:
  - ● Discussed the previous steps involved
  - ● Discussed the future steps

Follow-up meeting (2/06/2024)

- ➢ Attendees: Project Team Members
- ➢ Agenda: Review of new algorithms added
- ➢ Discussion Points:
  - ● Discussed accuracy of algorithms
  - ● Discussed ways to improve accuracy

Performance Optimization Meeting (18/06/2024)

- ➢ Attendees: Project Team Members
- ➢ Agenda: Review of performance of models
- ➢ Discussion Points:
  - ● Discussed testing and debugging of code

Sprint Review 2 (24/06/2024)

- ➢ Attendees: Project Team Members, Ms Anjali T S
- ➢ Agenda: Review of completed
- ➢ Discussion Points:
  - ● Demo of functioning models
  - ● Results of testing

Review Meetings (Every Sunday)

- ➢ Attendees: Project Team Members
- ➢ Agenda: Detailed review of work.

- Discussion Points:
    - Any roadblocks?
    - Subsequent steps to be taken

1.3. Problem Statement

Cyberbullying has become a significant issue in the digital age, affecting individuals across various online platforms and causing emotional distress, mental health issues, and, in severe cases, self-harm or suicide. Current detection methods are often inadequate, relying heavily on user reports and manual moderation, which are time-consuming and prone to bias. This project addresses the need for an automated, accurate, and robust system to detect and mitigate cyberbullying in real-time, using advanced machine learning (ML), deep learning (DL), and natural language processing (NLP) techniques. By leveraging these technologies, we aim to enhance the identification and prevention of cyberbullying, ensuring safer and more positive online environments.

Chapter 2: Planning

2.1 Deadline

The deadline for the project was the second week of June. This allowed for a 14-week development cycle starting from the third week of March and concluding in early June, covering all phases of the project from initial planning to deployment.

2.2 Project Timeline

Week 1 (March)	Communicate with project owner to deciderequirement of project
Week 2 (March)	Decide scope and research for papers
Week 3 (April)	Read all papers that others found. Find innovations
Week 4 (April)	Gather datasets and select best dataset.
Week 5 (April)	Data preprocessing of dataset.
Week 6 (April)	Implementing machine learning models on dataset
Week 7 (May)	Implementing Ensemble models on dataset
Week 8 (May)	Comparing and improving accuracy
Week 9 (May)	Implementing ANN,Bi-Gru deep learning models on dataset.
Week 10 (May)	Implementing LSTM, XLSMT,Bi LSTM

Week 11 (June)	Comparing and improving accuracy
Week 12 (June)	Final testing, bug fixing, and performance optimization

2.3 Phases and Modules

Phase 1 - Data Collection and Preprocessing

Task:

Collect relevant datasets
Clean and preprocess data
Balance the dataset

Collecting a diverse set of text data and preparing it for model training, ensuring a balanced representation of cyberbullying and non-cyberbullying content.

Key Achievements:

Extensive and balanced dataset
Preprocessed data ready for analysis

Phase 2 - Model Development

Task:

Develop and train machine learning models (GaussianNB, LogisticRegression, DecisionTree, AdaBoost, RandomForest)
Develop and train deep learning models (ANN, Bi-GRU, LSTM, XLSTM, Bi-LSTM)

Building and training various ML and DL models to compare their effectiveness in detecting cyberbullying.

Key Achievements:

Trained ML and DL models
Comparative performance analysis of models

Phase 3 - Natural Language Processing (NLP) Integration

Task:

Implement NLP techniques for text tokenization and classification
Integrate NLP with ML and DL models

Using NLP methods to enhance the understanding and classification of text data within the detection system.

Key Achievements:

Improved text processing and classification accuracy
Seamless integration of NLP with ML and DL models

Phase 5 - Model Evaluation and Optimization

Task:

Evaluate model performance using test data
Optimize models based on performance metrics
Select the best-performing model

Assessing the models' accuracy, precision, recall, and F1 score, and fine-tuning them for optimal performance.

Key Achievements:

Comprehensive model evaluation
Optimized and selected best-performing model (ANN with 92% accuracy)

2.4 Team Roles and Responsibilities

Each phase of the project is led by a dedicated team comprising key roles responsible for different aspects of development:

Scrum Master: Meenakshi Saju

Oversaw the agile development process, coordinated daily stand-ups, and ensured project progress aligned with timelines.

Business Analyst: Lakshminanda

Gathered and analyzed user requirements, translated them into actionable development tasks, and ensured the application met user expectations.

Developer: Smrithi Venugopal

Responsible for coding, implementing features according to specifications, conducting unit tests, ensuring code quality, and collaborating with team members to meet project milestones.

Throughout all phases, the team collaborated closely, with roles overlapping to support comprehensive development and efficient resolution of challenges.

2.5 Budget

Our project operates with a minimal budget as all necessary tools and platforms are either open-source or provided by the institution at no cost. The breakdown of the budget includes:

- Development Tools: Utilization of free tools such as Google Colab
- Human Resources: All team members, contribute their expertise and time voluntarily as part of their coursework, reflecting a zero-cost allocation for human resources.

This budget structure ensures cost-effectiveness while leveraging high-quality development tools and committed team contributions to achieve project goals effectively

Chapter 3 : Modeling – Design Phase

The modeling-design phase in software engineering is a critical stage where the system's architecture and components are conceptualized and defined. This phase involves creating detailed blueprints, including UML diagrams, flowcharts, and data models, to visualize the software structure and behavior. It ensures that the requirements gathered during the analysis phase are translated into a coherent plan. This phase is crucial because it helps identify potential issues early, promotes better understanding among stakeholders, and provides a clear roadmap for developers. A well-executed design phase leads to more efficient coding, easier maintenance, and a higher quality end product.

3.1 Use-case Diagram:

A use-case diagram is a visual representation of a system's functional requirements, illustrating interactions between users (actors) and system processes (use cases). It is important because it helps stakeholders understand the system's functionality, identifies user needs, and serves as a foundation for detailed design and development.

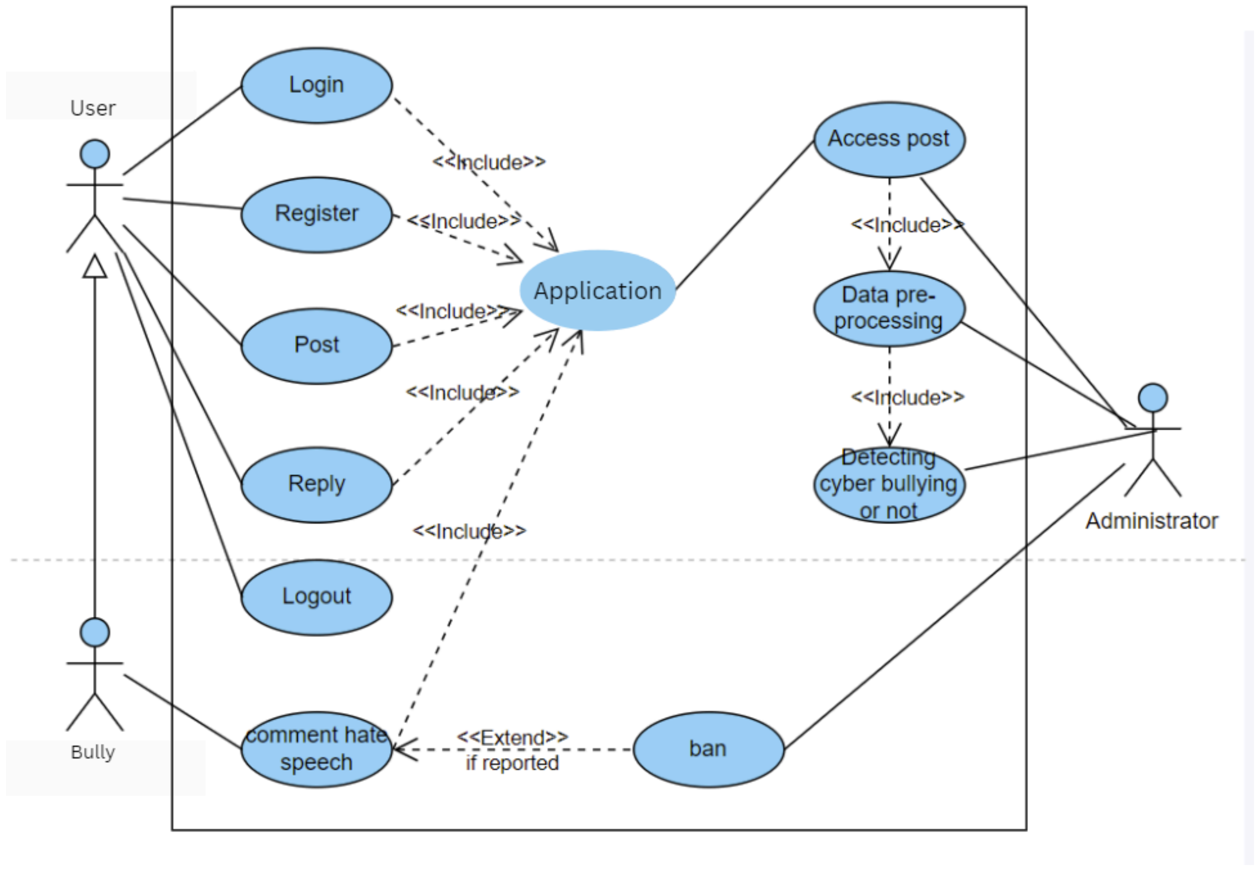
Actors:

- Application user - An individual who logs in/registers and has the ability to use the platform to post and read content
- Bully - An individual who violates acceptable ways of speech
- Administrator - An individual responsible for managing the application, including user accounts.

Use-cases:

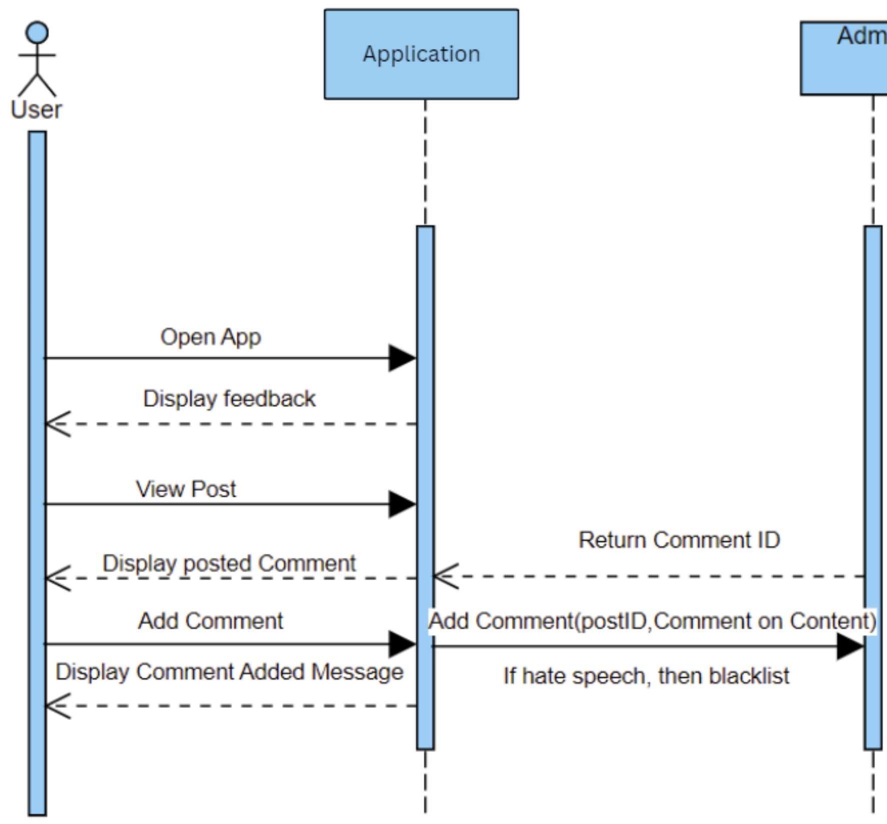
1. Login and registrations - Users register and log in to the application with their credentials. The system validates and grants access upon successful verification.
2. Posting content and comments – An authenticated user posts photos, posts or comments on the platform.
3. Commenting hate speech – The act of posting content which is considered as violating acceptable way of speech.
4. Accessing post – Being able to access the contents of the post, feature extraction.
5. Detecting cyberbullying – detecting if a particular post violates acceptable ways of speech.

6. Banning – if a user reports another user multiple times for cyberbullying, the reported user gets banned.



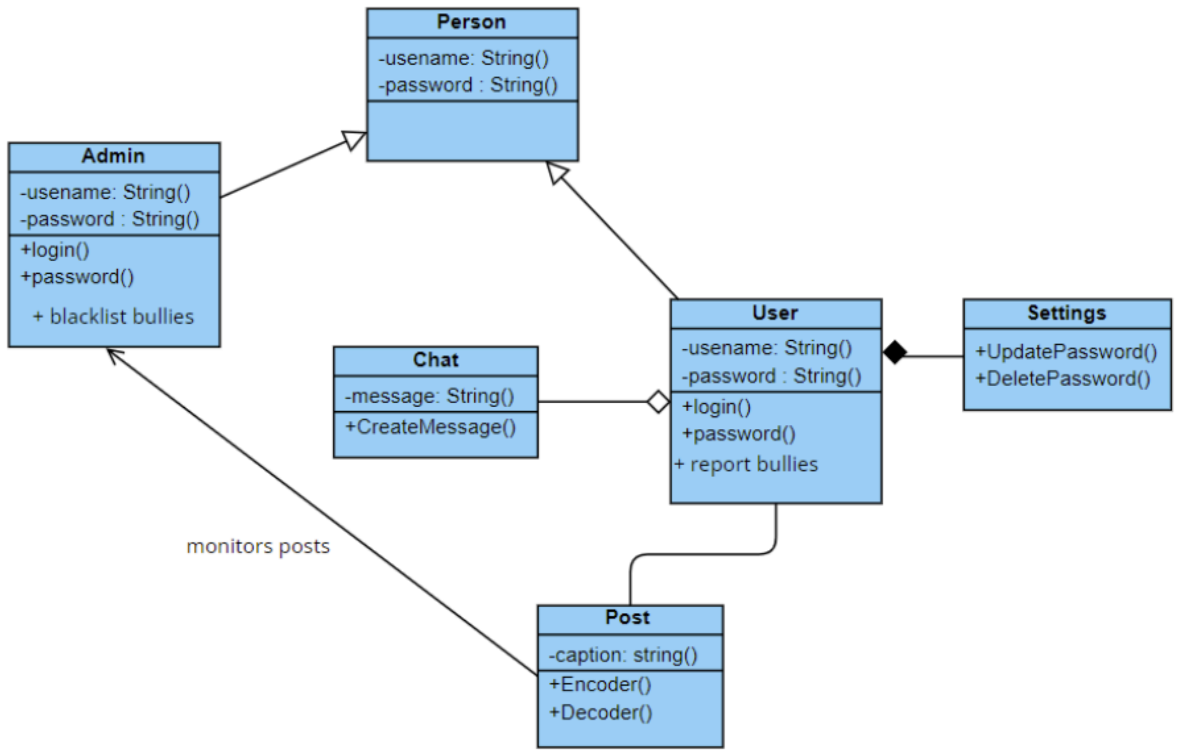
3.2 Sequence Diagram

A sequence diagram is a type of UML diagram that shows how objects interact in a particular sequence to accomplish a specific task. It details the order of messages exchanged between objects. Its purpose is to model the dynamic behavior of a system, ensuring clear communication and efficient process flow.



3.3 Class Diagram

A class diagram is a type of UML diagram that depicts the structure of a system by showing its classes, attributes, methods, and relationships. Its purpose is to provide a blueprint of the system's static aspects, facilitating understanding, design, and implementation by illustrating how different components interact and relate to each other.

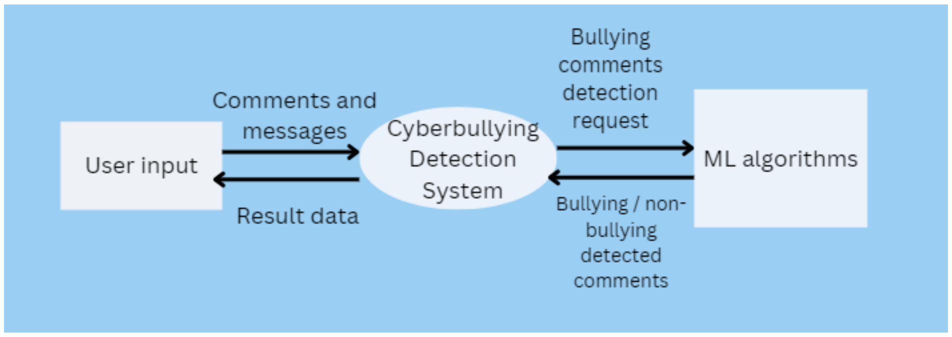


3.4 Data Flow Diagram

A Data Flow Diagram (DFD) visually represents the flow of data within a system, highlighting processes, data stores, data sources, and data destinations. Its purpose is to map out how information moves through the system, helping stakeholders understand data handling, identify inefficiencies, and plan for effective system design and optimization.

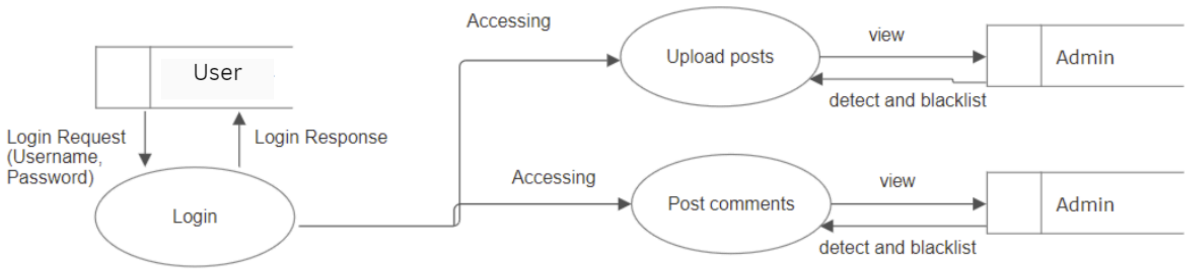
0-Level DFD:

A Level 0 Data Flow Diagram (DFD), also known as a context diagram, provides a high-level overview of a system. It illustrates the system's main process, external entities interacting with it, and data flow between them. Its purpose is to establish the system's scope and external interfaces clearly.


Comments and messages
User input
Result data
Cyberbullying Detection System
Bullying comments detection request
ML algorithms
Bullying / non-bullying detected comments

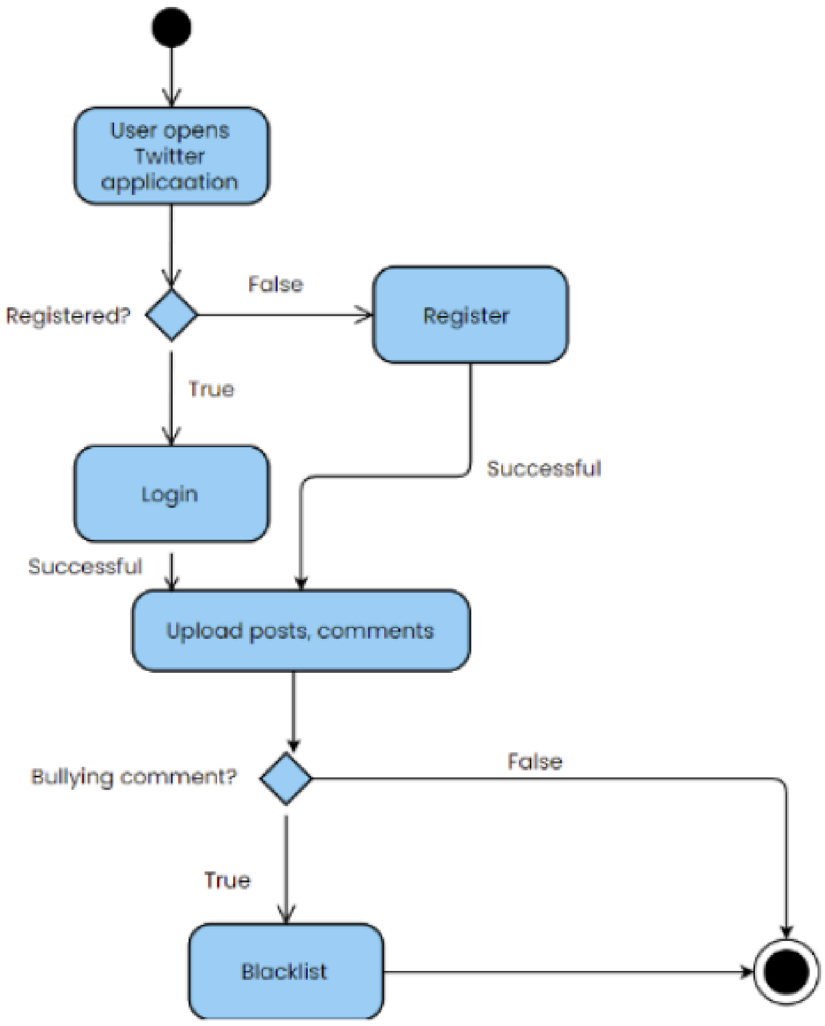
Level – 1 DFD:

A Level 1 Data Flow Diagram (DFD) expands on the Level 0 DFD by breaking down the main process into sub-processes. It provides a detailed view of the system's functional components and their interactions, showing data flow between sub-processes, data stores, and external entities. This helps in understanding and analyzing specific functionalities.



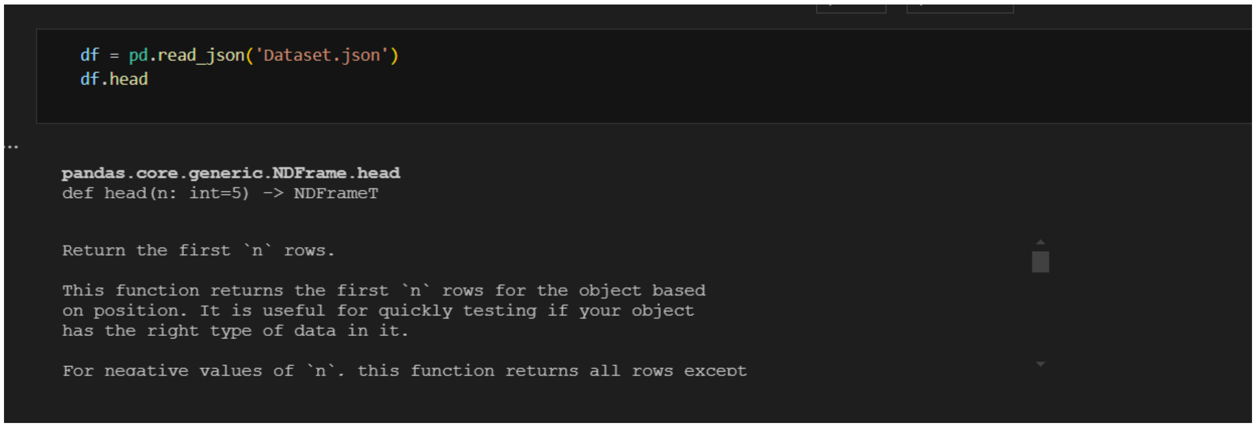
3.5 Activity Diagram

An activity diagram is a type of UML diagram that represents the workflow of activities in a system or process. It highlights the sequence of actions, decision points, and parallel processes. Its purpose is to model dynamic aspects of the system, ensuring clarity in process flow and aiding in workflow optimization.


User opens Twitter applicaation
Registered?
False
Register
True
Login
Successful
Successful
Upload posts, comments
Bullying comment?
False
True
Blacklist

Chapter 4: Coding

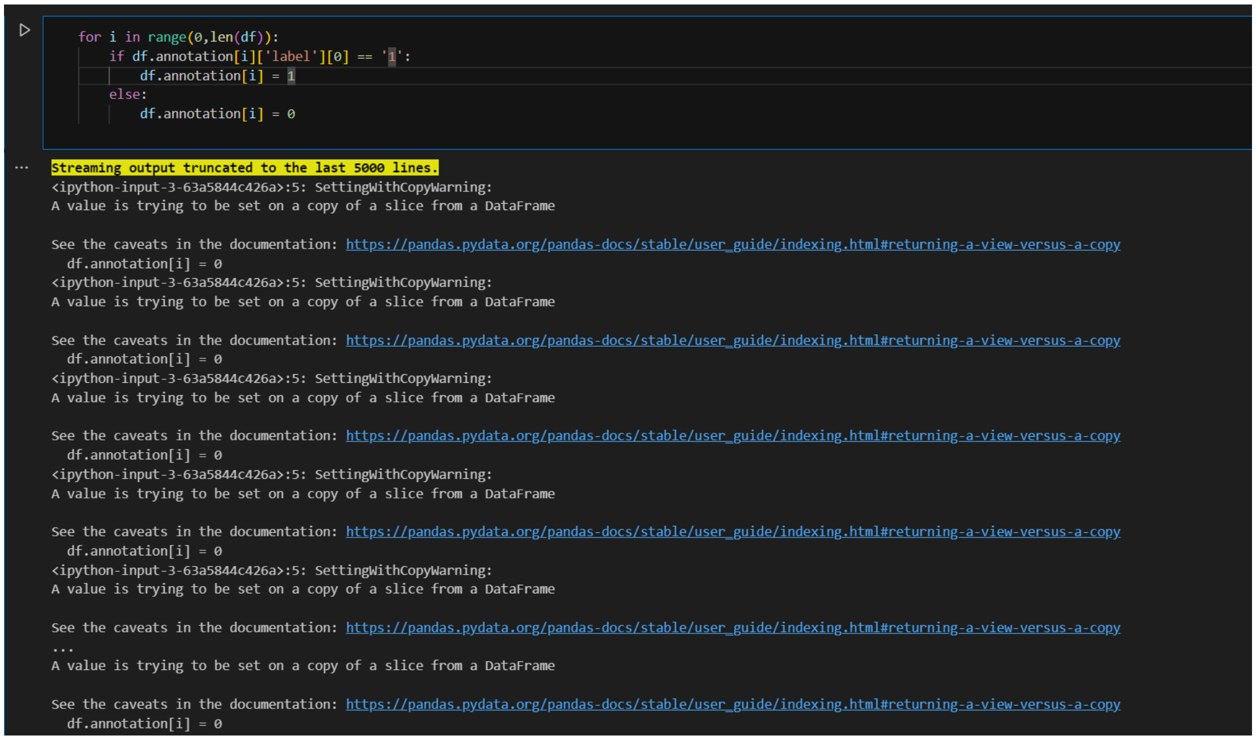
4.1 Loading the dataset:



```
df = pd.read_json('Dataset.json')
df.head
```

```
pandas.core.generic.NDFrame.head
def head(n: int=5) -> NDFrameT

Return the first `n` rows.

This function returns the first `n` rows for the object based
on position. It is useful for quickly testing if your object
has the right type of data in it.

For negative values of `n`, this function returns all rows except
```

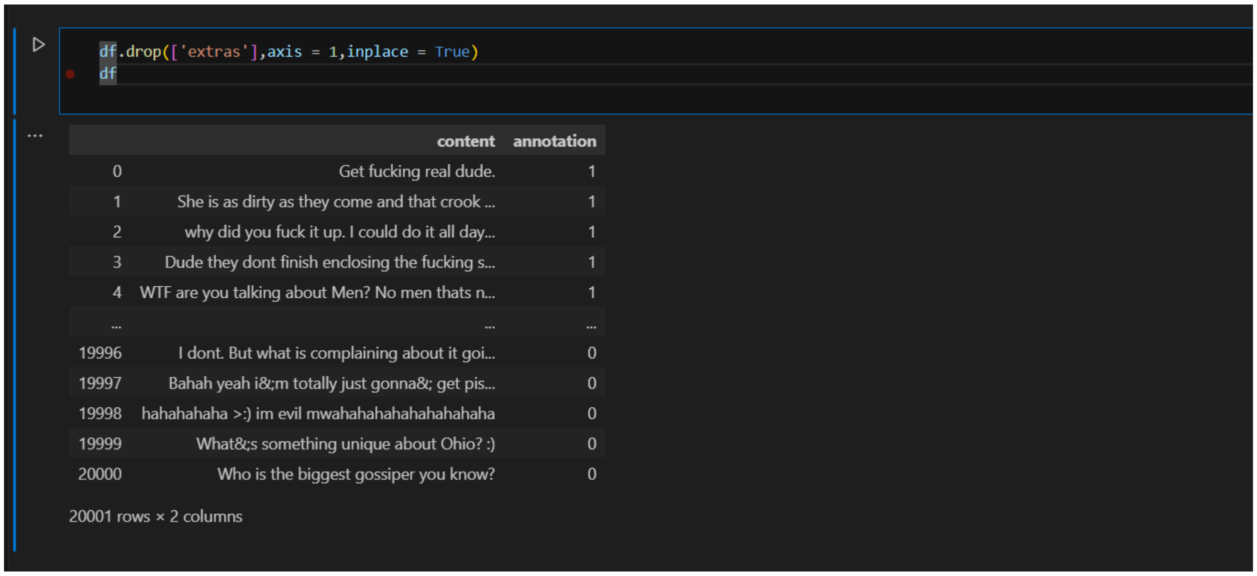
4.2 Converting annotations to Binary Format:



```
for i in range(0,len(df)):
    if df.annotation[i]['label'][0] == '1':
        df.annotation[i] = 1
    else:
        df.annotation[i] = 0
```

```
Streaming output truncated to the last 5000 lines.
<ipython-input-3-63a5844c426a>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.annotation[i] = 0
<ipython-input-3-63a5844c426a>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.annotation[i] = 0
<ipython-input-3-63a5844c426a>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.annotation[i] = 0
<ipython-input-3-63a5844c426a>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.annotation[i] = 0
<ipython-input-3-63a5844c426a>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
...
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.annotation[i] = 0
```

4.3 Dropping column extra:

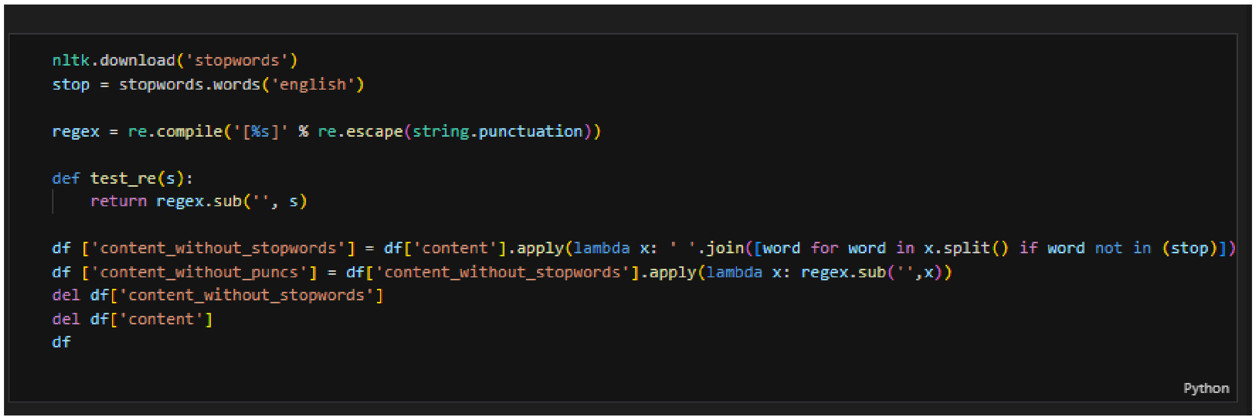


```
df.drop(['extras'],axis = 1,inplace = True)
df
```

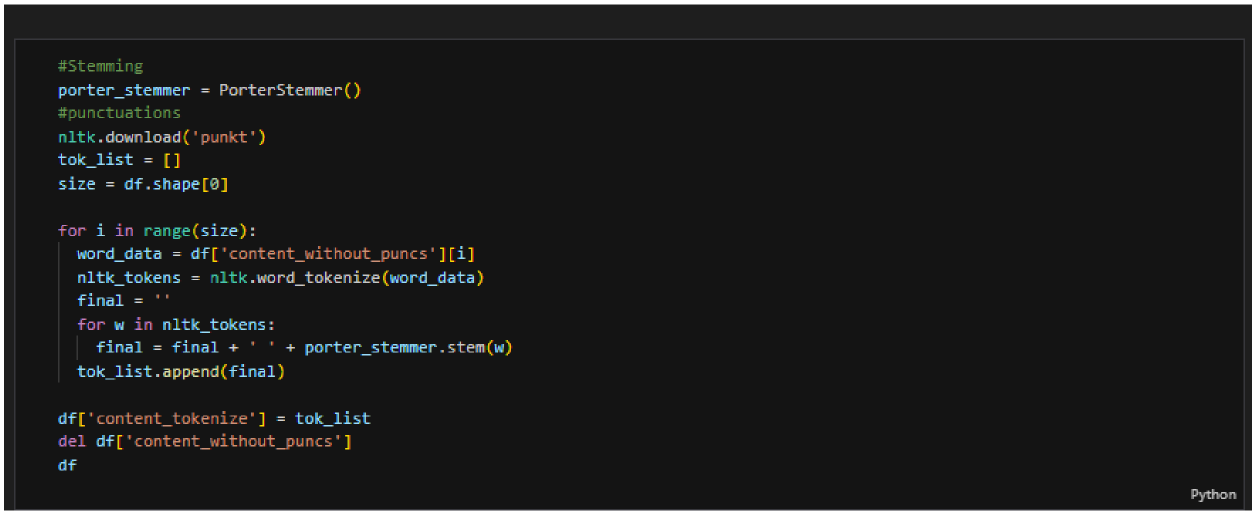
	content	annotation
0	Get fucking real dude.	1
1	She is as dirty as they come and that crook ...	1
2	why did you fuck it up. I could do it all day...	1
3	Dude they dont finish enclosing the fucking s...	1
4	WTF are you talking about Men? No men thats n...	1
...	...	...
19996	I dont. But what is complaining about it goi...	0
19997	Bahah yeah i&;m totally just gonna&; get pis...	0
19998	hahahahaha >:) im evil mwahahahahahahahaha	0
19999	What&;s something unique about Ohio? :)	0
20000	Who is the biggest gossiper you know?	0

20001 rows × 2 columns

4.4 Data Preprocessing:

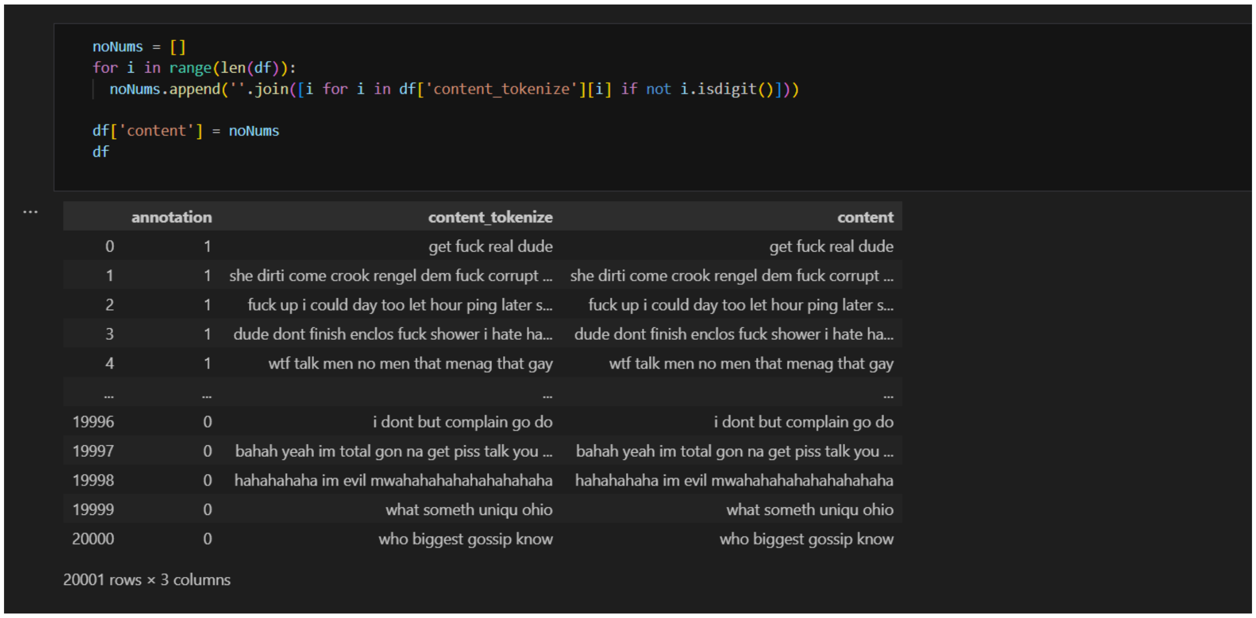


```
nltk.download('stopwords')
stop = stopwords.words('english')

regex = re.compile('[%s]' % re.escape(string.punctuation))

def test_re(s):
    return regex.sub('', s)

df ['content_without_stopwords'] = df['content'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)])
df ['content_without_puncs'] = df['content_without_stopwords'].apply(lambda x: regex.sub('',x))
del df['content_without_stopwords']
del df['content']
df
```



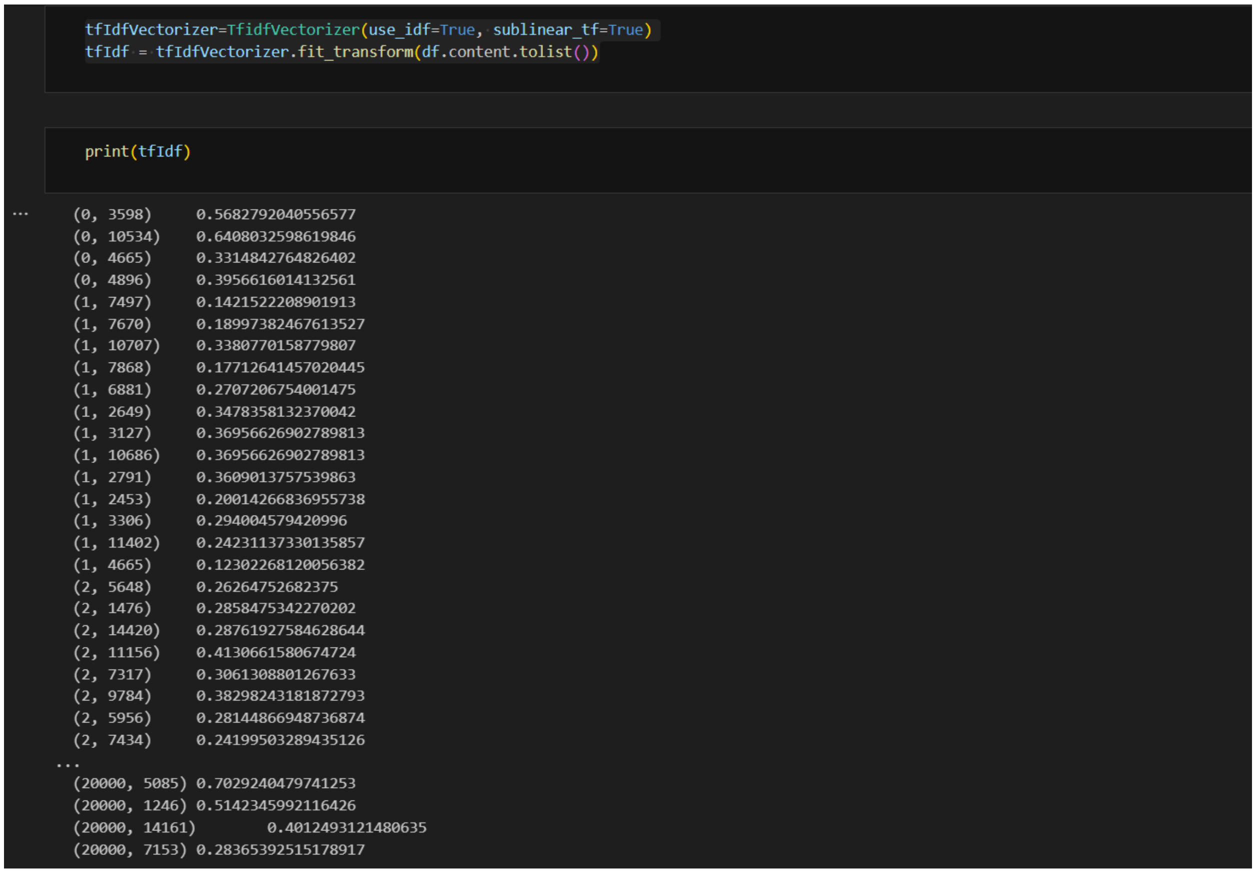
```python
#Stemming
porter_stemmer = PorterStemmer()
#punctuations
nltk.download('punkt')
tok_list = []
size = df.shape[0]

for i in range(size):
  word_data = df['content_without_puncs'][i]
  nltk_tokens = nltk.word_tokenize(word_data)
  final = ''
  for w in nltk_tokens:
    final = final + ' ' + porter_stemmer.stem(w)
  tok_list.append(final)

df['content_tokenize'] = tok_list
del df['content_without_puncs']
df
```

Python



```python
noNums = []
for i in range(len(df)):
  noNums.append(''.join([i for i in df['content_tokenize'][i] if not i.isdigit()]))

df['content'] = noNums
df
```

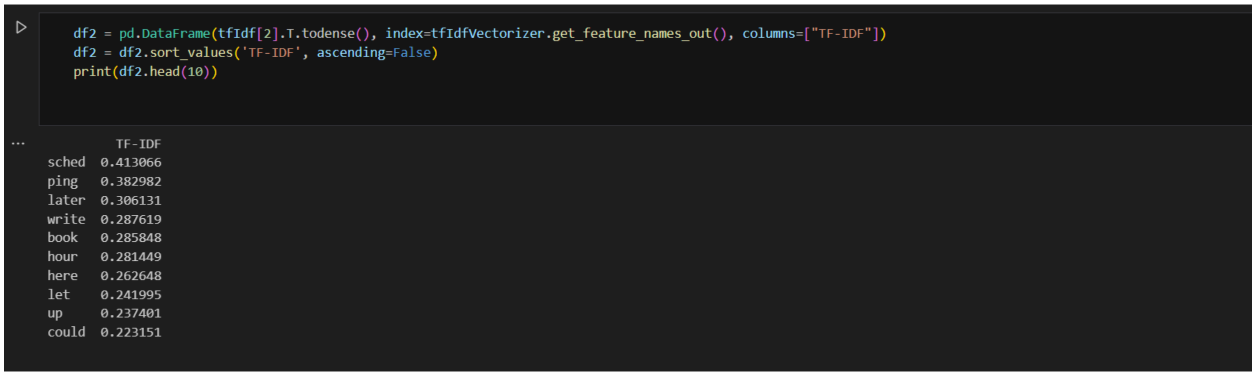
```
       annotation                           content_tokenize                                    content
0               1                         get fuck real dude                         get fuck real dude
1               1  she dirti come crook rengel dem fuck corrupt ...  she dirti come crook rengel dem fuck corrupt ...
2               1     fuck up i could day too let hour ping later s...     fuck up i could day too let hour ping later s...
3               1    dude dont finish enclos fuck shower i hate ha...    dude dont finish enclos fuck shower i hate ha...
4               1         wtf talk men no men that menag that gay         wtf talk men no men that menag that gay
...           ...                                              ...                                        ...
19996           0                    i dont but complain go do                    i dont but complain go do
19997           0   bahah yeah im total gon na get piss talk you ...   bahah yeah im total gon na get piss talk you ...
19998           0   hahahahaha im evil mwahahahahahahahahaha   hahahahaha im evil mwahahahahahahahahaha
19999           0                       what someth uniqu ohio                       what someth uniqu ohio
20000           0                      who biggest gossip know                      who biggest gossip know

20001 rows × 3 columns
```



```
tfIdfVectorizer=TfidfVectorizer(use_idf=True, sublinear_tf=True)
tfIdf = tfIdfVectorizer.fit_transform(df.content.tolist())
```

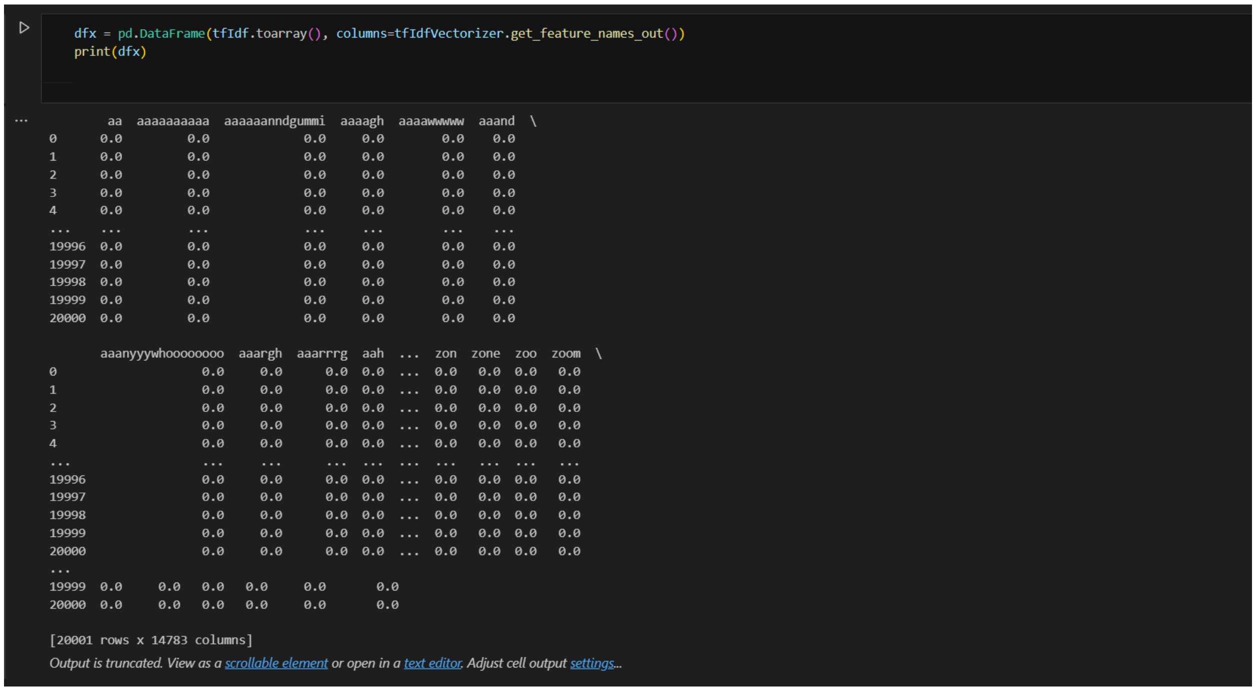
```
print(tfIdf)
```

```
...   (0, 3598)     0.5682792040556577
      (0, 10534)    0.6408032598619846
      (0, 4665)     0.3314842764826402
      (0, 4896)     0.3956616014132561
      (1, 7497)     0.1421522208901913
      (1, 7670)     0.18997382467613527
      (1, 10707)    0.3380770158779807
      (1, 7868)     0.17712641457020445
      (1, 6881)     0.2707206754001475
      (1, 2649)     0.3478358132370042
      (1, 3127)     0.36956626902789813
      (1, 10686)    0.36956626902789813
      (1, 2791)     0.3609013757539863
      (1, 2453)     0.20014266836955738
      (1, 3306)     0.294004579420996
      (1, 11402)    0.24231137330135857
      (1, 4665)     0.12302268120056382
      (2, 5648)     0.26264752682375
      (2, 1476)     0.2858475342270202
      (2, 14420)    0.28761927584628644
      (2, 11156)    0.4130661580674724
      (2, 7317)     0.3061308801267633
      (2, 9784)     0.38298243181872793
      (2, 5956)     0.28144866948736874
      (2, 7434)     0.24199503289435126
    ...
      (20000, 5085) 0.7029240479741253
      (20000, 1246) 0.5142345992116426
      (20000, 14161)        0.4012493121480635
      (20000, 7153) 0.28365392515178917
```



```
df2 = pd.DataFrame(tfIdf[2].T.todense(), index=tfIdfVectorizer.get_feature_names_out(), columns=["TF-IDF"])
df2 = df2.sort_values('TF-IDF', ascending=False)
print(df2.head(10))
```

```
...         TF-IDF
    sched  0.413066
    ping   0.382982
    later  0.306131
    write  0.287619
    book   0.285848
    hour   0.281449
    here   0.262648
    let    0.241995
    up     0.237401
    could  0.223151
```

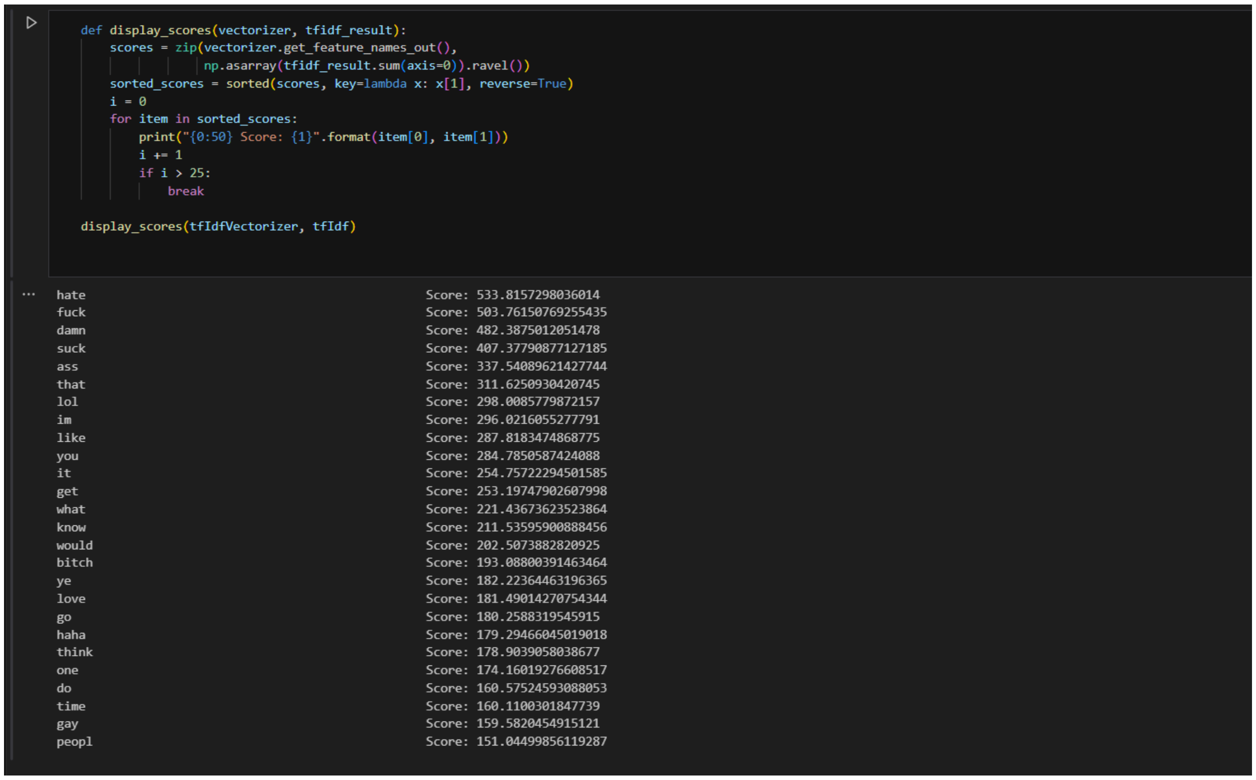


```python
dfx = pd.DataFrame(tfIdf.toarray(), columns=tfIdfVectorizer.get_feature_names_out())
print(dfx)
```

```
        aa  aaaaaaaaaa  aaaaaanndgummi  aaaagh  aaaawwwww  aaand  \
0      0.0         0.0             0.0     0.0        0.0    0.0
1      0.0         0.0             0.0     0.0        0.0    0.0
2      0.0         0.0             0.0     0.0        0.0    0.0
3      0.0         0.0             0.0     0.0        0.0    0.0
4      0.0         0.0             0.0     0.0        0.0    0.0
...    ...         ...             ...     ...        ...    ...
19996  0.0         0.0             0.0     0.0        0.0    0.0
19997  0.0         0.0             0.0     0.0        0.0    0.0
19998  0.0         0.0             0.0     0.0        0.0    0.0
19999  0.0         0.0             0.0     0.0        0.0    0.0
20000  0.0         0.0             0.0     0.0        0.0    0.0

       aaanyyywhoooooooo  aaargh  aaarrrg  aah  ...  zon  zone  zoo  zoom  \
0                    0.0     0.0      0.0  0.0  ...  0.0   0.0  0.0   0.0
1                    0.0     0.0      0.0  0.0  ...  0.0   0.0  0.0   0.0
2                    0.0     0.0      0.0  0.0  ...  0.0   0.0  0.0   0.0
3                    0.0     0.0      0.0  0.0  ...  0.0   0.0  0.0   0.0
4                    0.0     0.0      0.0  0.0  ...  0.0   0.0  0.0   0.0
...                  ...     ...      ...  ...  ...  ...   ...  ...   ...
19996                0.0     0.0      0.0  0.0  ...  0.0   0.0  0.0   0.0
19997                0.0     0.0      0.0  0.0  ...  0.0   0.0  0.0   0.0
19998                0.0     0.0      0.0  0.0  ...  0.0   0.0  0.0   0.0
19999                0.0     0.0      0.0  0.0  ...  0.0   0.0  0.0   0.0
20000                0.0     0.0      0.0  0.0  ...  0.0   0.0  0.0   0.0
...
19999  0.0     0.0  0.0  0.0     0.0       0.0
20000  0.0     0.0  0.0  0.0     0.0       0.0

[20001 rows x 14783 columns]
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

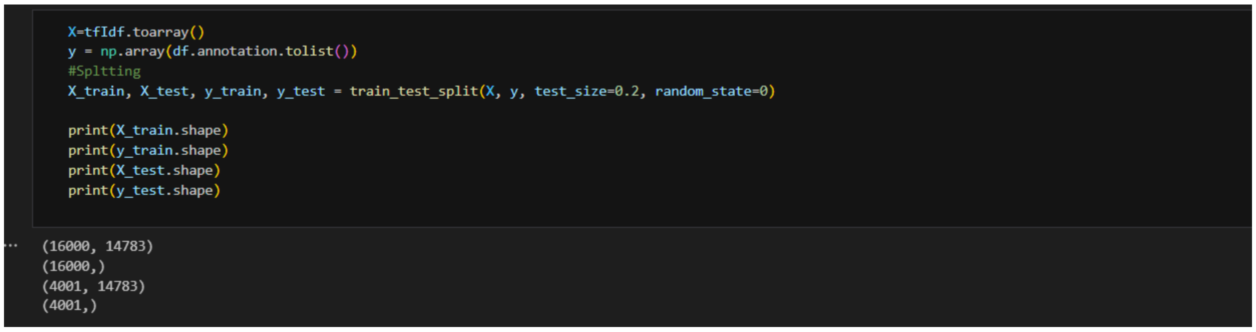
Displaying Scores :



```python
def display_scores(vectorizer, tfidf_result):
    scores = zip(vectorizer.get_feature_names_out(),
                 np.asarray(tfidf_result.sum(axis=0)).ravel())
    sorted_scores = sorted(scores, key=lambda x: x[1], reverse=True)
    i = 0
    for item in sorted_scores:
        print("{0:50} Score: {1}".format(item[0], item[1]))
        i += 1
        if i > 25:
            break

display_scores(tfIdfVectorizer, tfIdf)
```

```
hate                                               Score: 533.8157298036014
fuck                                               Score: 503.76150769255435
damn                                               Score: 482.3875012051478
suck                                               Score: 407.37790877127185
ass                                                Score: 337.54089621427744
that                                               Score: 311.6250930420745
lol                                                Score: 298.0085779872157
im                                                 Score: 296.0216055277791
like                                               Score: 287.8183474868775
you                                                Score: 284.7850587424088
it                                                 Score: 254.75722294501585
get                                                Score: 253.19747902607998
what                                               Score: 221.43673623523864
know                                               Score: 211.53595900888456
would                                              Score: 202.5073882820925
bitch                                              Score: 193.08800391463464
ye                                                 Score: 182.22364463196365
love                                               Score: 181.49014270754344
go                                                 Score: 180.2588319545915
haha                                               Score: 179.29466045019018
think                                              Score: 178.9039058038677
one                                                Score: 174.16019276608517
do                                                 Score: 160.57524593088053
time                                               Score: 160.1100301847739
gay                                                Score: 159.5820454915121
peopl                                              Score: 151.04499856119287
```

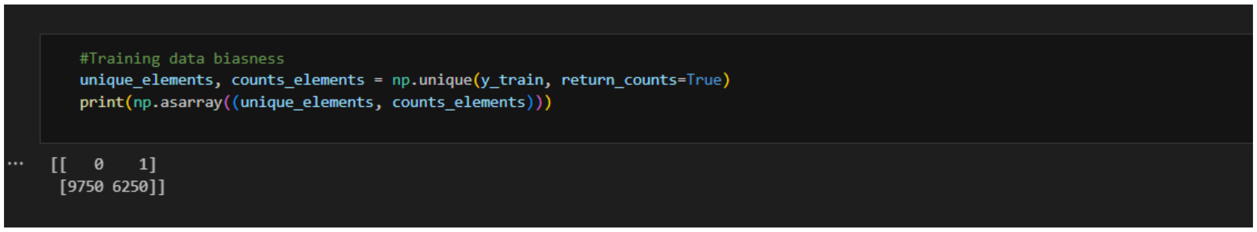
4.5 Splitting into training and testing data:



```
X=tfIdf.toarray()
y = np.array(df.annotation.tolist())
#Spltting
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

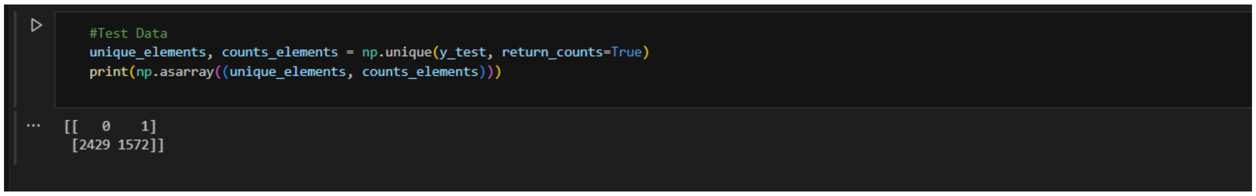
```
(16000, 14783)
(16000,)
(4001, 14783)
(4001,)
```

Checking for biases in training and testing data:



```
#Training data biasness
unique_elements, counts_elements = np.unique(y_train, return_counts=True)
print(np.asarray((unique_elements, counts_elements)))
```

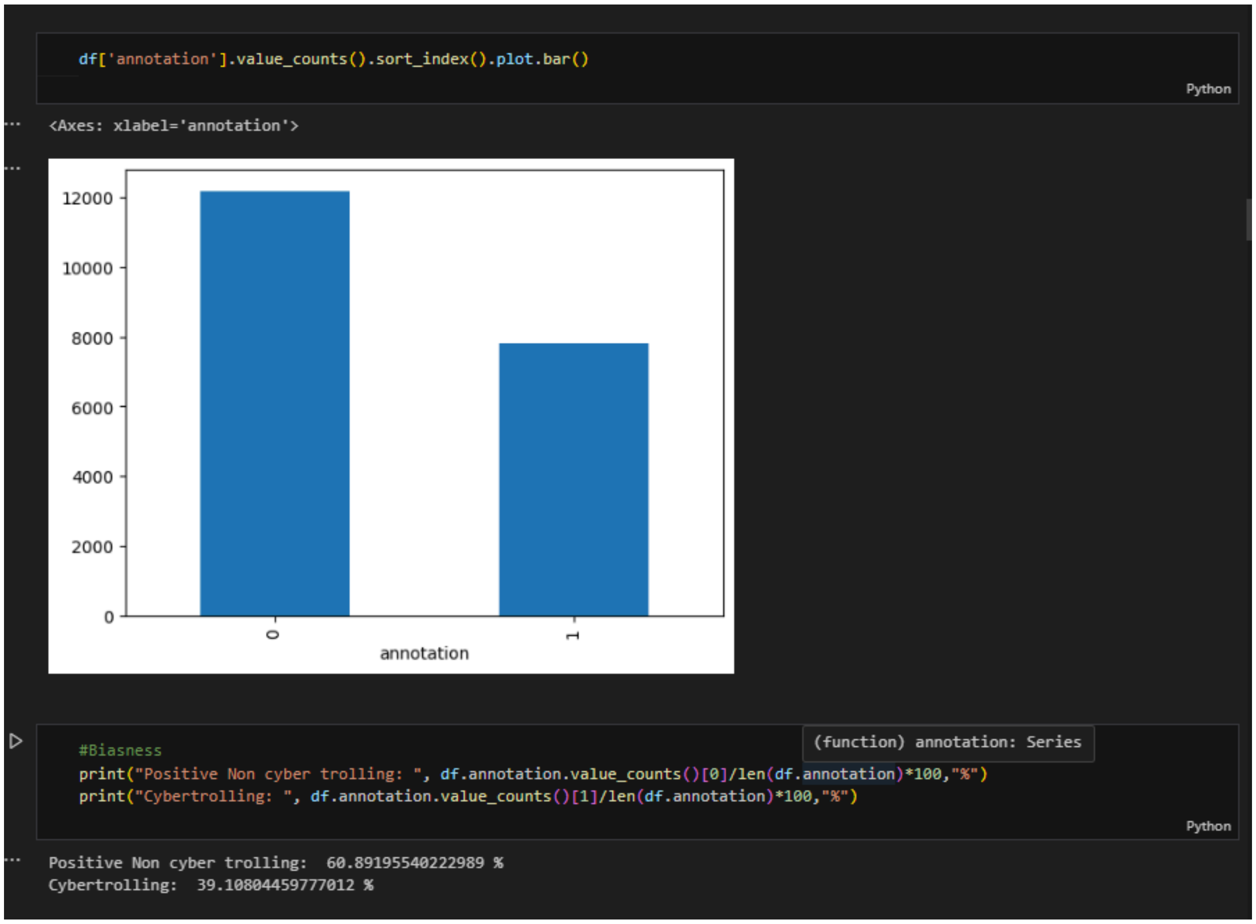
```
[[   0    1]
 [9750 6250]]
```



```
#Test Data
unique_elements, counts_elements = np.unique(y_test, return_counts=True)
print(np.asarray((unique_elements, counts_elements)))
```

```
[[   0    1]
 [2429 1572]]
```

4.5 Visualization:



```
df['annotation'].value_counts().sort_index().plot.bar()
```

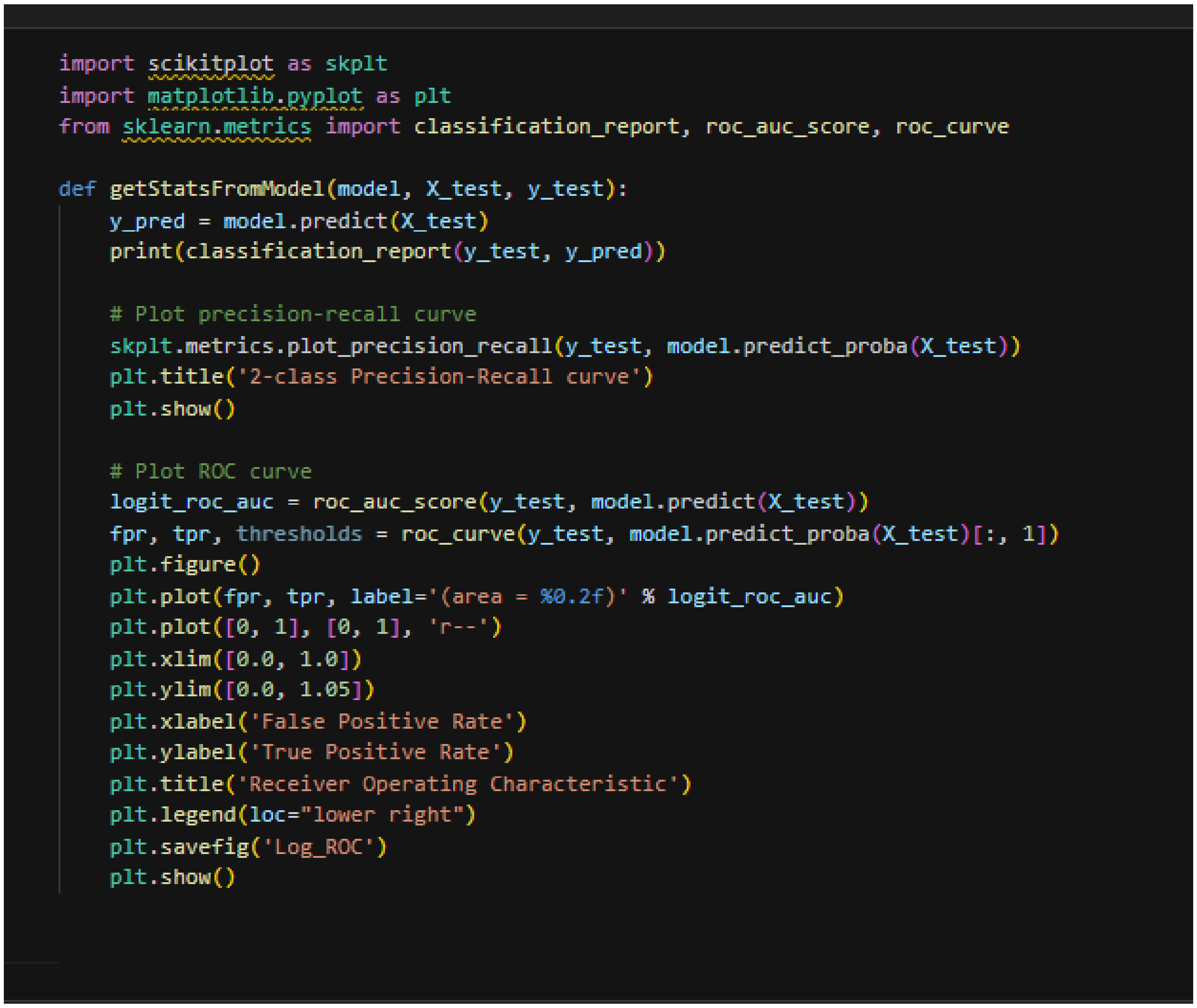
```
<Axes: xlabel='annotation'>
```

```
#Biasness
print("Positive Non cyber trolling: ", df.annotation.value_counts()[0]/len(df.annotation)*100,"%")
print("Cybertrolling: ", df.annotation.value_counts()[1]/len(df.annotation)*100,"%")
```

(function) annotation: Series

```
Positive Non cyber trolling:  60.89195540222989 %
Cybertrolling:  39.10804459777012 %
```

4.6 Training and Calculating Scores:



```
import scikitplot as skplt
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, roc_auc_score, roc_curve

def getStatsFromModel(model, X_test, y_test):
    y_pred = model.predict(X_test)
    print(classification_report(y_test, y_pred))

    # Plot precision-recall curve
    skplt.metrics.plot_precision_recall(y_test, model.predict_proba(X_test))
    plt.title('2-class Precision-Recall curve')
    plt.show()

    # Plot ROC curve
    logit_roc_auc = roc_auc_score(y_test, model.predict(X_test))
    fpr, tpr, thresholds = roc_curve(y_test, model.predict_proba(X_test)[:, 1])
    plt.figure()
    plt.plot(fpr, tpr, label='(area = %0.2f)' % logit_roc_auc)
    plt.plot([0, 1], [0, 1], 'r--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic')
    plt.legend(loc="lower right")
    plt.savefig('Log_ROC')
    plt.show()
```

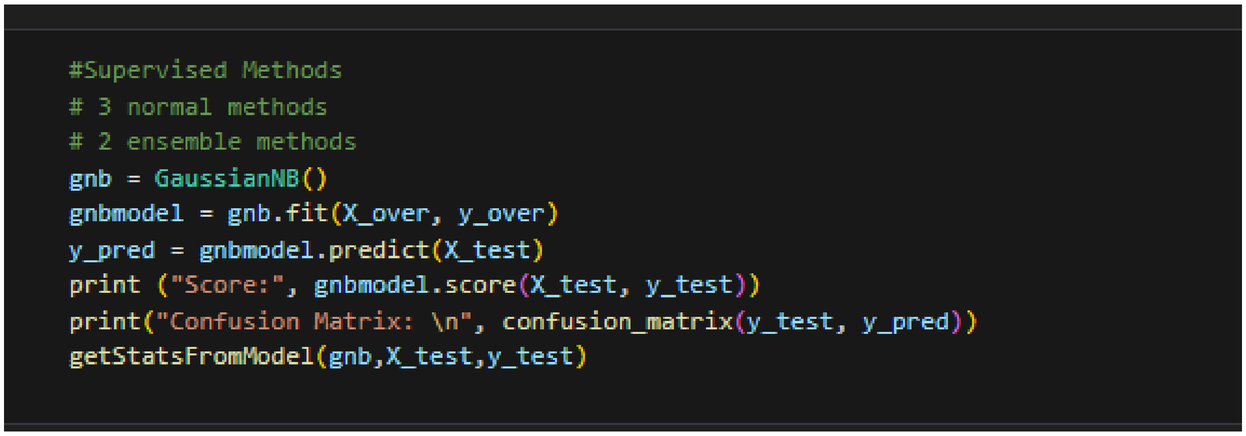
4.7 TRAINING MODELS:

I. Machine Learning Models

In our project, we have used 3 normal and 2 ensemble models. The models we use are Supervised Machine Learning Algorithms.

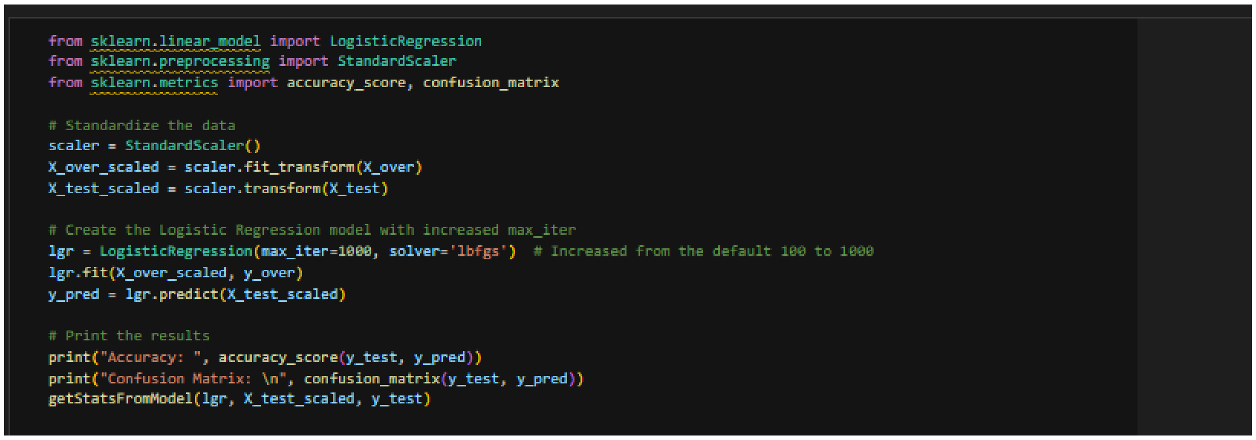
Normal Methods

1. Gaussian Naive Bayes



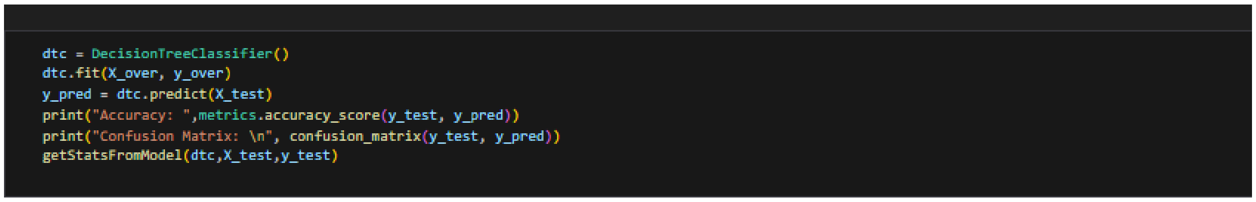
```
#Supervised Methods
# 3 normal methods
# 2 ensemble methods
gnb = GaussianNB()
gnbmodel = gnb.fit(X_over, y_over)
y_pred = gnbmodel.predict(X_test)
print ("Score:", gnbmodel.score(X_test, y_test))
print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
getStatsFromModel(gnb,X_test,y_test)
```

2. Logistic Regression



```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix

# Standardize the data
scaler = StandardScaler()
X_over_scaled = scaler.fit_transform(X_over)
X_test_scaled = scaler.transform(X_test)

# Create the Logistic Regression model with increased max_iter
lgr = LogisticRegression(max_iter=1000, solver='lbfgs')  # Increased from the default 100 to 1000
lgr.fit(X_over_scaled, y_over)
y_pred = lgr.predict(X_test_scaled)

# Print the results
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
getStatsFromModel(lgr, X_test_scaled, y_test)
```

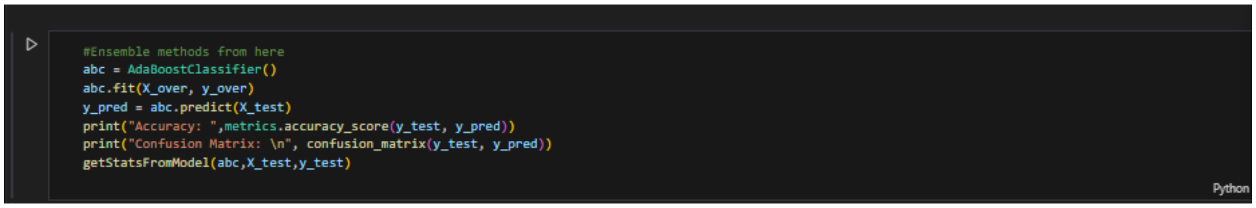
3. Decision Tree Classifier



```
dtc = DecisionTreeClassifier()
dtc.fit(X_over, y_over)
y_pred = dtc.predict(X_test)
print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
getStatsFromModel(dtc,X_test,y_test)
```

Ensemble Methods

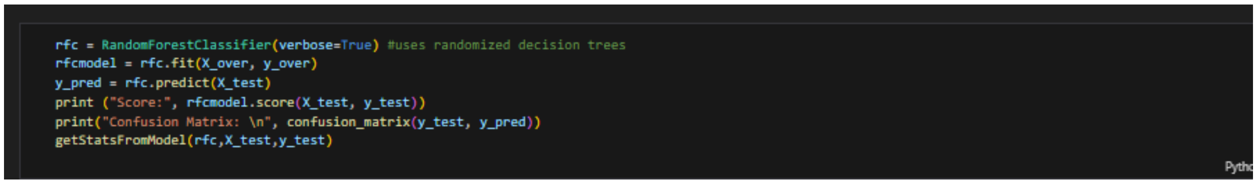
1. AdaBoost



```python
#Ensemble methods from here
abc = AdaBoostClassifier()
abc.fit(X_over, y_over)
y_pred = abc.predict(X_test)
print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
getStatsFromModel(abc,X_test,y_test)
```

Python

2. Random Forest

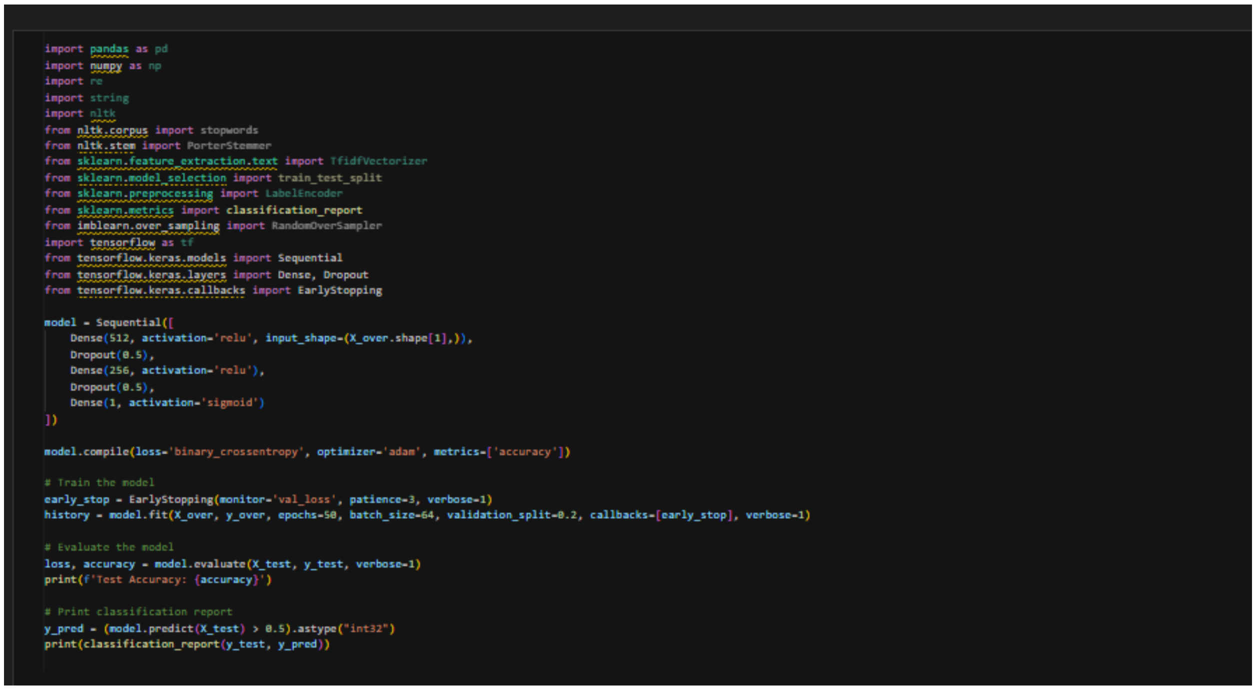


```python
rfc = RandomForestClassifier(verbose=True) #uses randomized decision trees
rfcmodel = rfc.fit(X_over, y_over)
y_pred = rfc.predict(X_test)
print ("Score:", rfcmodel.score(X_test, y_test))
print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
getStatsFromModel(rfc,X_test,y_test)
```

Pytho

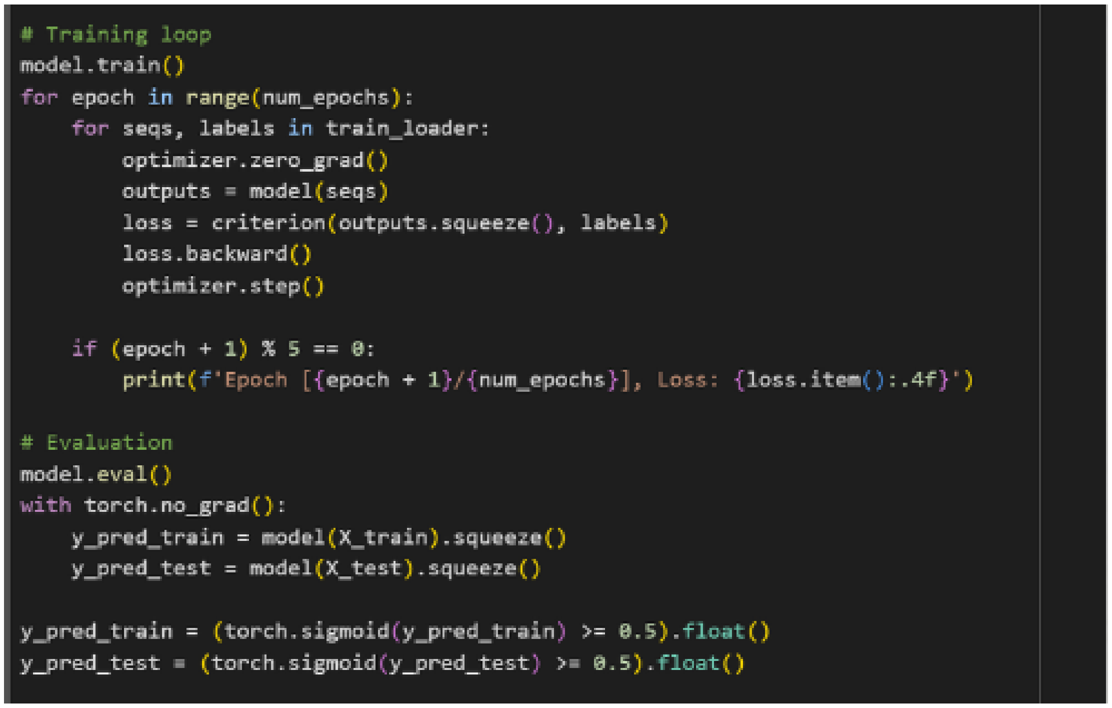
II. Deep Learning Models

1. ANN

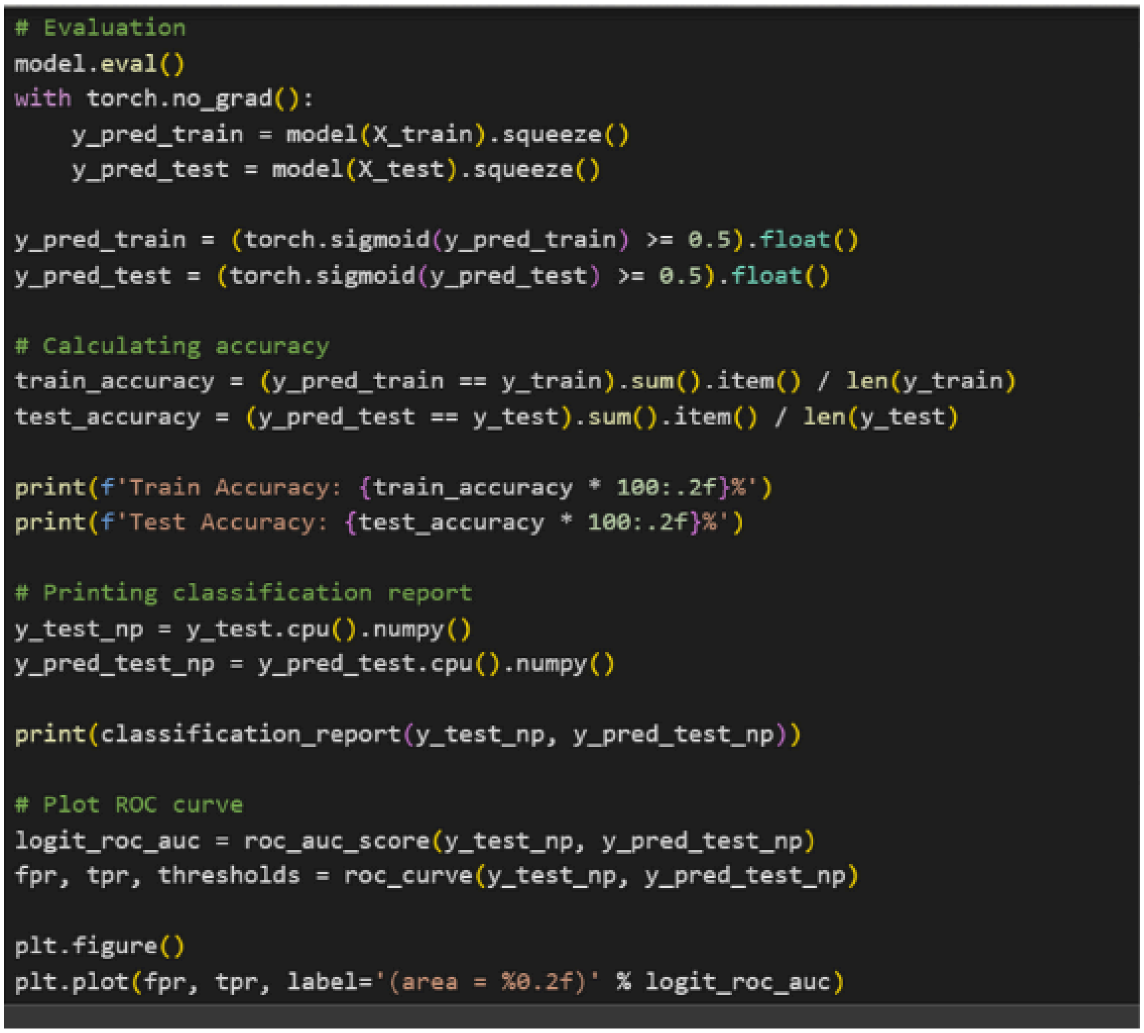


```python
import pandas as pd
import numpy as np
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report
from imblearn.over_sampling import RandomOverSampler
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

model = Sequential([
    Dense(512, activation='relu', input_shape=(X_over.shape[1],)),
    Dropout(0.5),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
early_stop = EarlyStopping(monitor='val_loss', patience=3, verbose=1)
history = model.fit(X_over, y_over, epochs=50, batch_size=64, validation_split=0.2, callbacks=[early_stop], verbose=1)

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test, verbose=1)
print(f'Test Accuracy: {accuracy}')

# Print classification report
y_pred = (model.predict(X_test) > 0.5).astype("int32")
print(classification_report(y_test, y_pred))
```

2. XLSTM

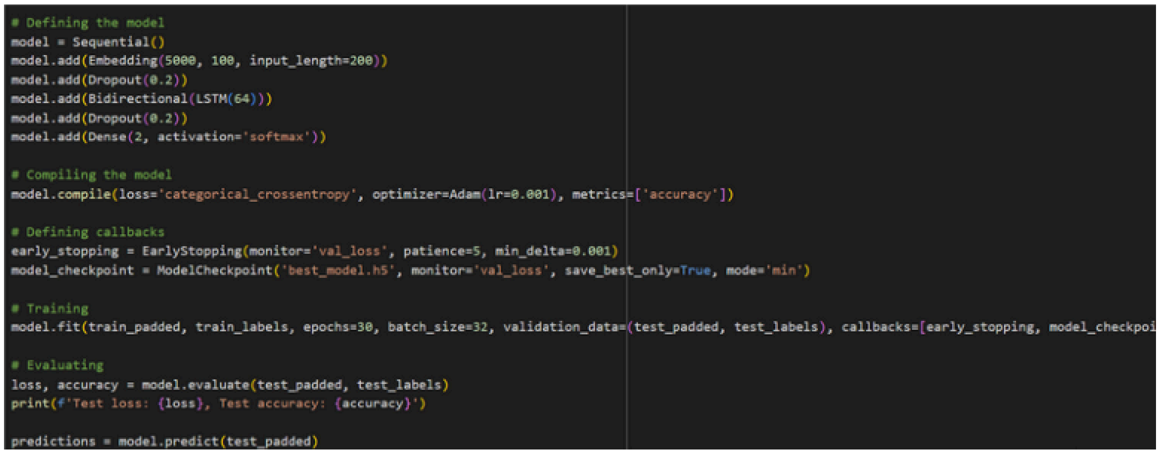


```
# Training loop
model.train()
for epoch in range(num_epochs):
    for seqs, labels in train_loader:
        optimizer.zero_grad()
        outputs = model(seqs)
        loss = criterion(outputs.squeeze(), labels)
        loss.backward()
        optimizer.step()

    if (epoch + 1) % 5 == 0:
        print(f'Epoch [{epoch + 1}/{num_epochs}], Loss: {loss.item():.4f}')

# Evaluation
model.eval()
with torch.no_grad():
    y_pred_train = model(X_train).squeeze()
    y_pred_test = model(X_test).squeeze()

y_pred_train = (torch.sigmoid(y_pred_train) >= 0.5).float()
y_pred_test = (torch.sigmoid(y_pred_test) >= 0.5).float()
```



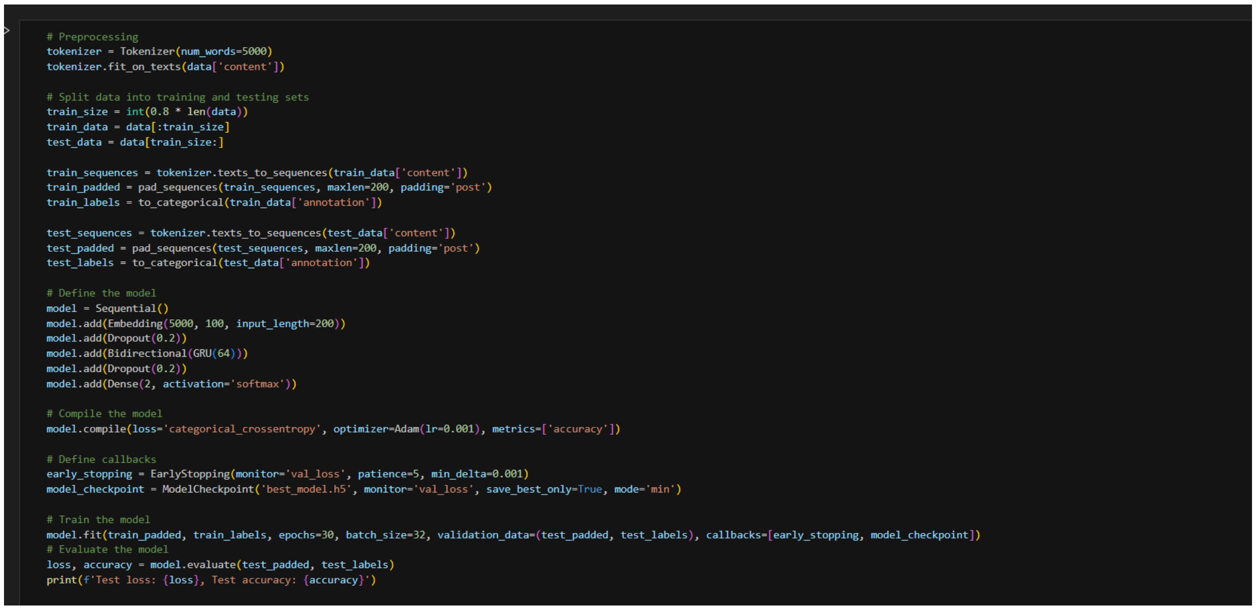
```
# Evaluation
model.eval()
with torch.no_grad():
    y_pred_train = model(X_train).squeeze()
    y_pred_test = model(X_test).squeeze()

y_pred_train = (torch.sigmoid(y_pred_train) >= 0.5).float()
y_pred_test = (torch.sigmoid(y_pred_test) >= 0.5).float()

# Calculating accuracy
train_accuracy = (y_pred_train == y_train).sum().item() / len(y_train)
test_accuracy = (y_pred_test == y_test).sum().item() / len(y_test)

print(f'Train Accuracy: {train_accuracy * 100:.2f}%')
print(f'Test Accuracy: {test_accuracy * 100:.2f}%')

# Printing classification report
y_test_np = y_test.cpu().numpy()
y_pred_test_np = y_pred_test.cpu().numpy()

print(classification_report(y_test_np, y_pred_test_np))

# Plot ROC curve
logit_roc_auc = roc_auc_score(y_test_np, y_pred_test_np)
fpr, tpr, thresholds = roc_curve(y_test_np, y_pred_test_np)

plt.figure()
plt.plot(fpr, tpr, label='(area = %0.2f)' % logit_roc_auc)
```

3. Bi LSTM

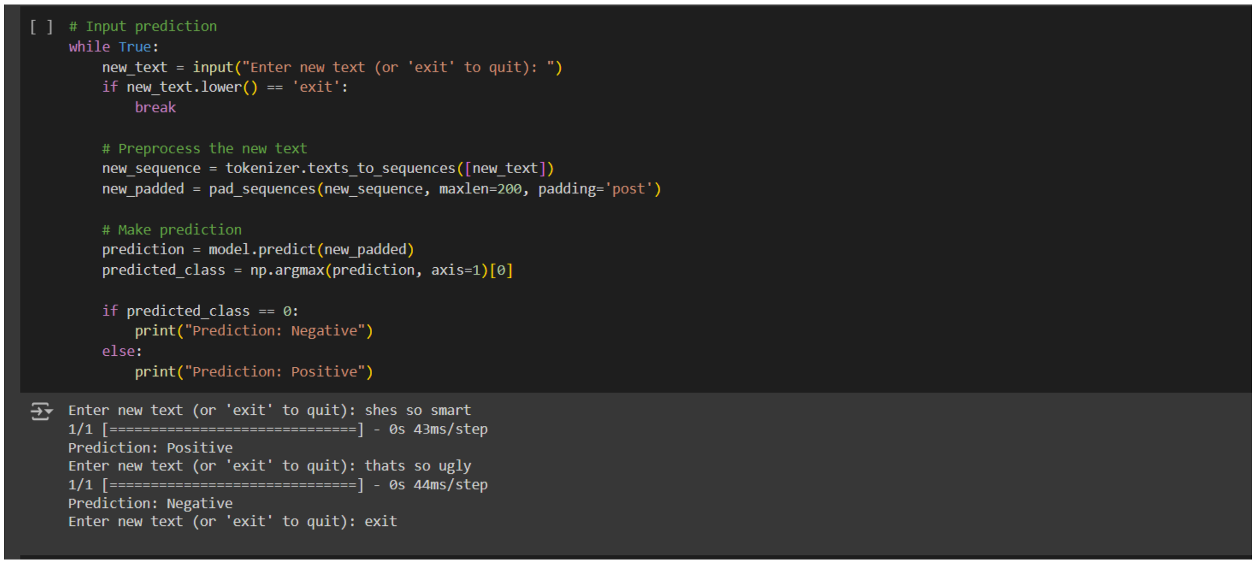


```
# Defining the model
model = Sequential()
model.add(Embedding(5000, 100, input_length=200))
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(64)))
model.add(Dropout(0.2))
model.add(Dense(2, activation='softmax'))

# Compiling the model
model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.001), metrics=['accuracy'])

# Defining callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=5, min_delta=0.001)
model_checkpoint = ModelCheckpoint('best_model.h5', monitor='val_loss', save_best_only=True, mode='min')

# Training
model.fit(train_padded, train_labels, epochs=30, batch_size=32, validation_data=(test_padded, test_labels), callbacks=[early_stopping, model_checkpoi

# Evaluating
loss, accuracy = model.evaluate(test_padded, test_labels)
print(f'Test loss: {loss}, Test accuracy: {accuracy}')

predictions = model.predict(test_padded)
```

4. Bi GRU with Prediction



```python
# Preprocessing
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(data['content'])

# Split data into training and testing sets
train_size = int(0.8 * len(data))
train_data = data[:train_size]
test_data = data[train_size:]

train_sequences = tokenizer.texts_to_sequences(train_data['content'])
train_padded = pad_sequences(train_sequences, maxlen=200, padding='post')
train_labels = to_categorical(train_data['annotation'])

test_sequences = tokenizer.texts_to_sequences(test_data['content'])
test_padded = pad_sequences(test_sequences, maxlen=200, padding='post')
test_labels = to_categorical(test_data['annotation'])

# Define the model
model = Sequential()
model.add(Embedding(5000, 100, input_length=200))
model.add(Dropout(0.2))
model.add(Bidirectional(GRU(64)))
model.add(Dropout(0.2))
model.add(Dense(2, activation='softmax'))

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.001), metrics=['accuracy'])

# Define callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=5, min_delta=0.001)
model_checkpoint = ModelCheckpoint('best_model.h5', monitor='val_loss', save_best_only=True, mode='min')

# Train the model
model.fit(train_padded, train_labels, epochs=30, batch_size=32, validation_data=(test_padded, test_labels), callbacks=[early_stopping, model_checkpoint])
# Evaluate the model
loss, accuracy = model.evaluate(test_padded, test_labels)
print(f'Test loss: {loss}, Test accuracy: {accuracy}')
```



```python
[ ]  # Input prediction
     while True:
         new_text = input("Enter new text (or 'exit' to quit): ")
         if new_text.lower() == 'exit':
             break

         # Preprocess the new text
         new_sequence = tokenizer.texts_to_sequences([new_text])
         new_padded = pad_sequences(new_sequence, maxlen=200, padding='post')

         # Make prediction
         prediction = model.predict(new_padded)
         predicted_class = np.argmax(prediction, axis=1)[0]

         if predicted_class == 0:
             print("Prediction: Negative")
         else:
             print("Prediction: Positive")
```

```
Enter new text (or 'exit' to quit): shes so smart
1/1 [==============================] - 0s 43ms/step
Prediction: Positive
Enter new text (or 'exit' to quit): thats so ugly
1/1 [==============================] - 0s 44ms/step
Prediction: Negative
Enter new text (or 'exit' to quit): exit
```

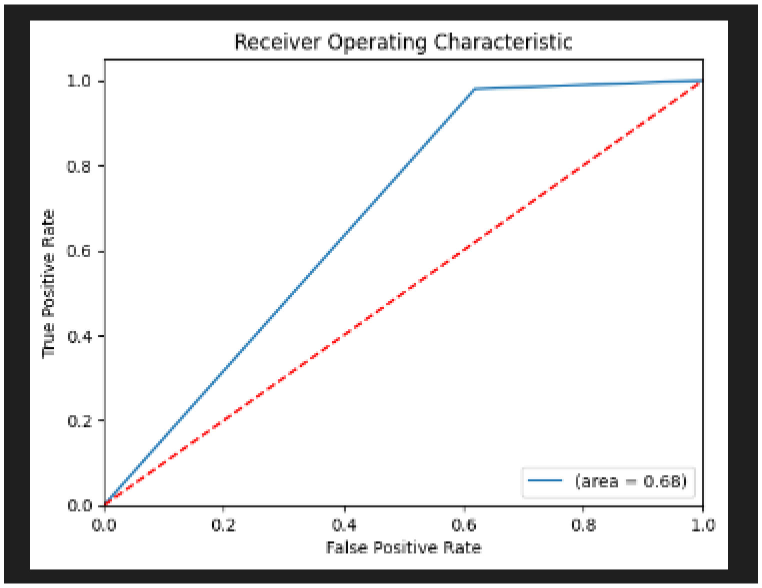
Chapter 5: Testing

5.1 Performance Testing:

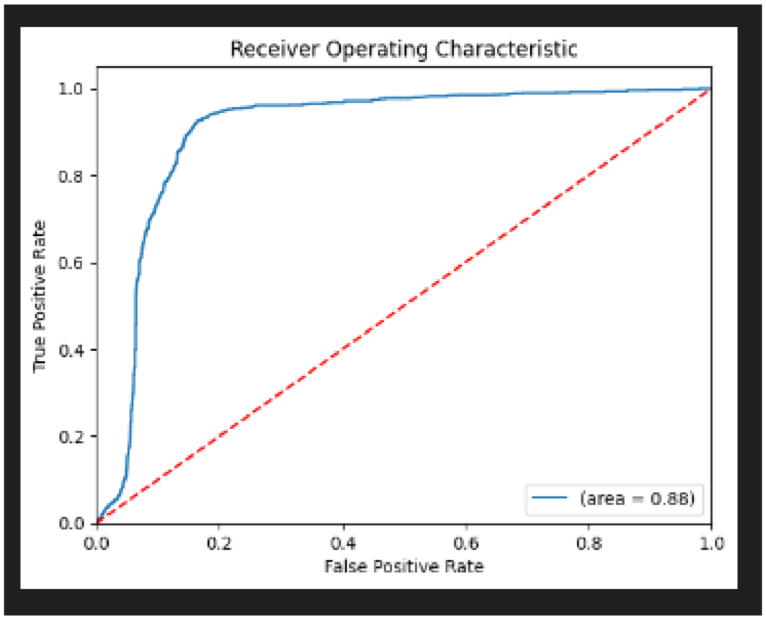
To enhance the quality assurance of our system, we also include metrics such as ROC (Receiver Operating Characteristic) curve, accuracy, precision, and recall to evaluate the performance of our application's predictive functionalities, if any.
➔ The ROC Curve visually evaluates the performance of a binary classifier by plotting the True Positive Rate (Recall) against the False Positive Rate across different classification thresholds.
➔ Accuracy assesses the overall correctness of a model's predictions, calculated as the ratio of correctly predicted instances to the total dataset size.
➔ Precision measures the accuracy of positive predictions, indicating the ratio of true positives to the sum of true positives and false positives.
➔ Recall (Sensitivity) gauges a model's ability to identify all relevant instances within a dataset, calculated as the ratio of true positives to the sum of true positives and false negatives.
These metrics collectively ensure that our models predict effectively, delivering accurate and reliable results.

I. Normal Methods

1. Naive Bayes

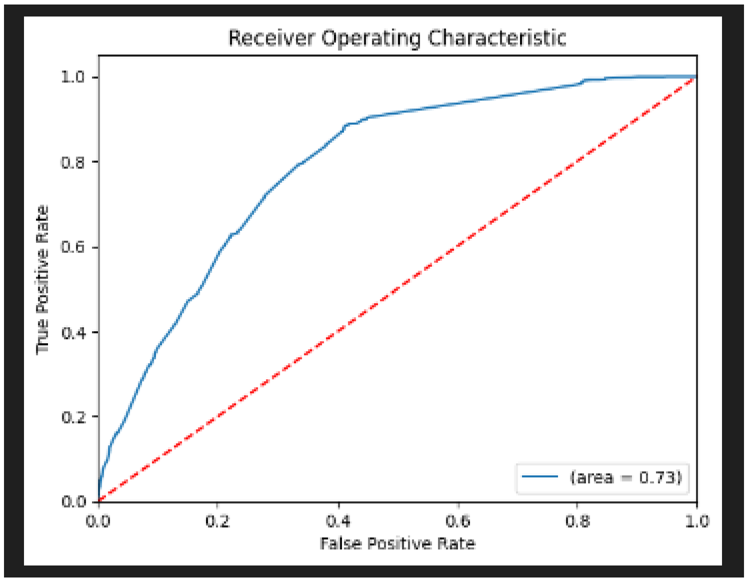


2. Logistic Regression

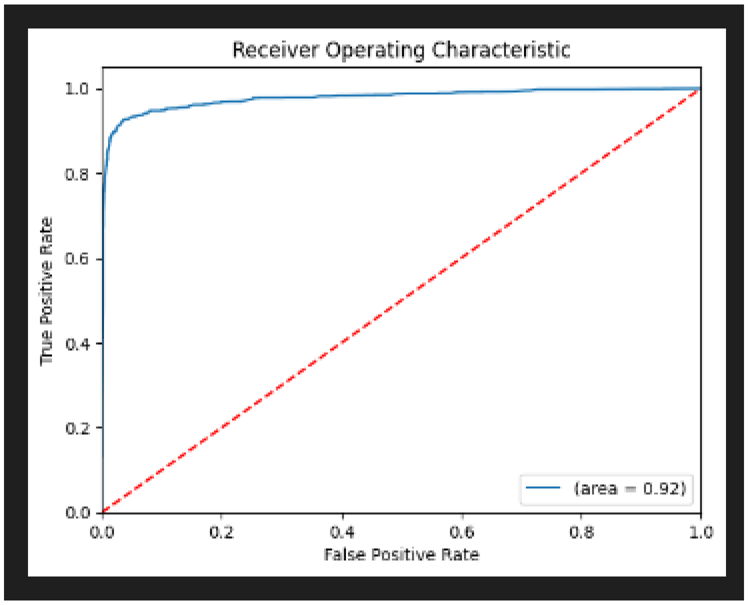


II. Ensemble Methods

1. AdaBoost

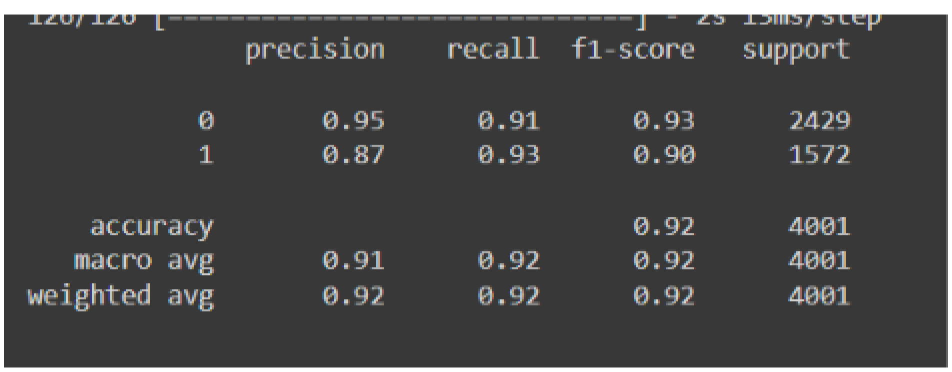


2. Random Forest

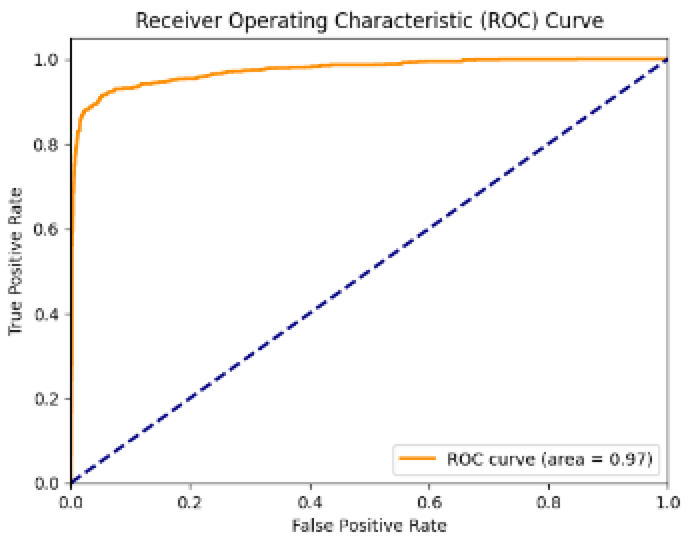


III. Deep Learning Methods:

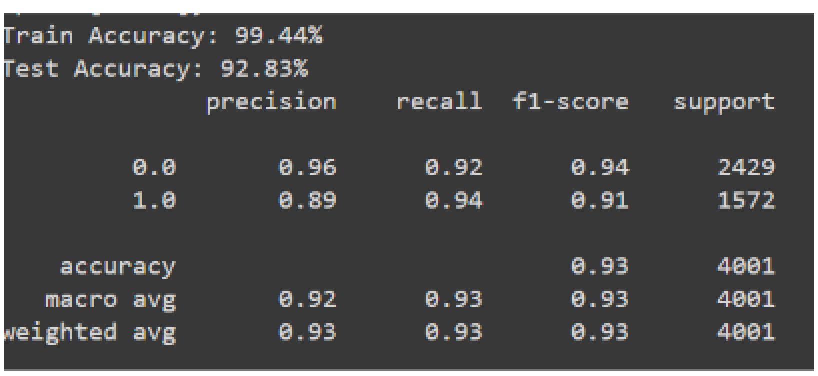
1. ANN:



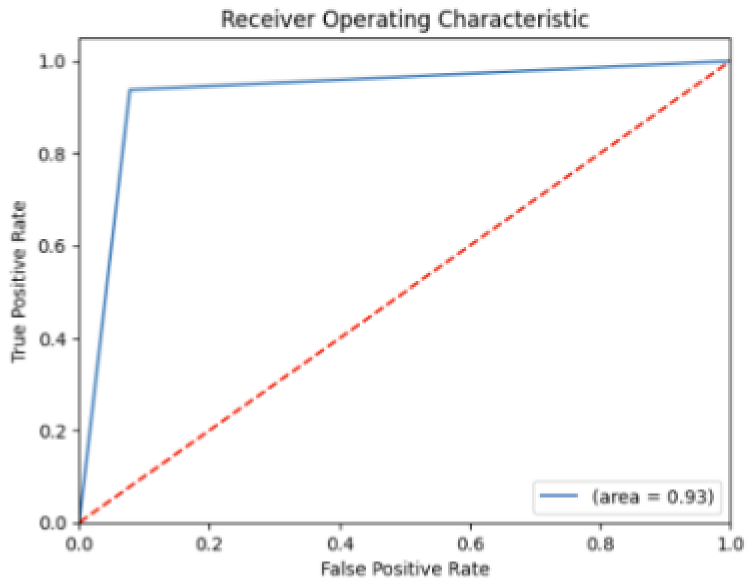
```
120/126 [==============================] - 2s 13ms/step
              precision    recall  f1-score   support

           0       0.95      0.91      0.93      2429
           1       0.87      0.93      0.90      1572

    accuracy                           0.92      4001
   macro avg       0.91      0.92      0.92      4001
weighted avg       0.92      0.92      0.92      4001
```



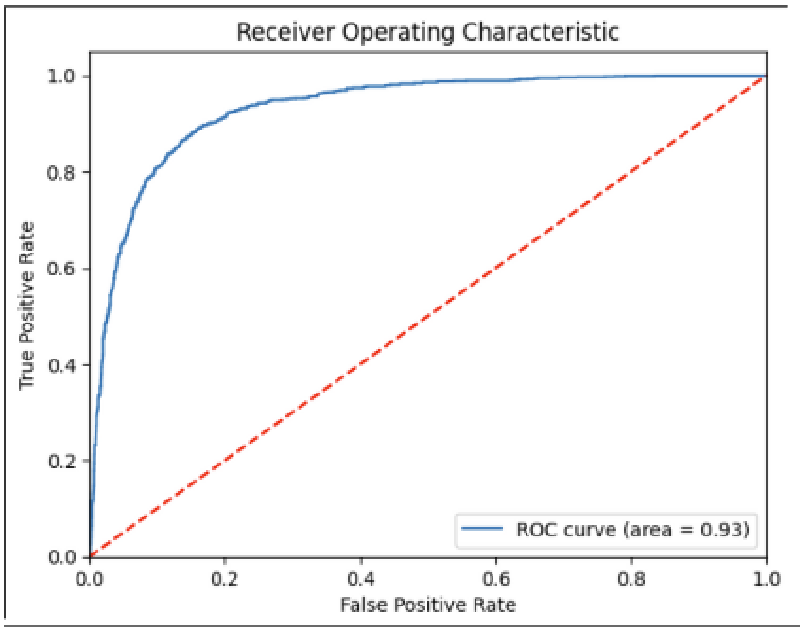
2. XLSTM

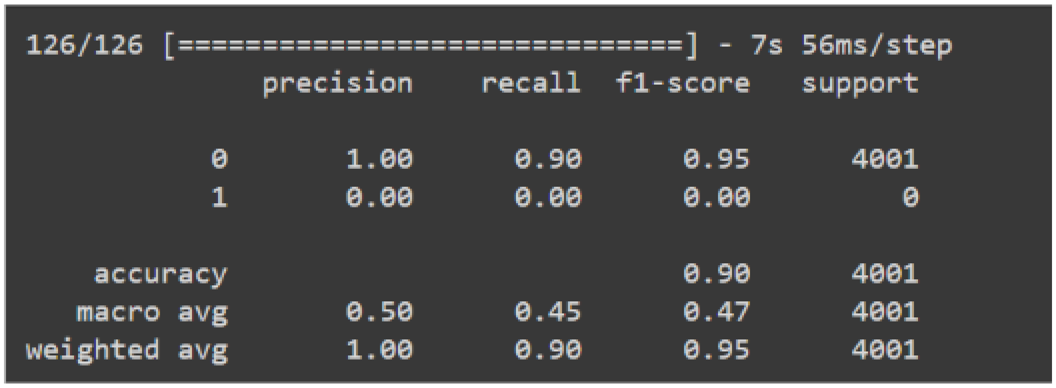


```
Train Accuracy: 99.44%
Test Accuracy: 92.83%
              precision    recall  f1-score   support

         0.0       0.96      0.92      0.94      2429
         1.0       0.89      0.94      0.91      1572

    accuracy                           0.93      4001
   macro avg       0.92      0.93      0.93      4001
weighted avg       0.93      0.93      0.93      4001
```



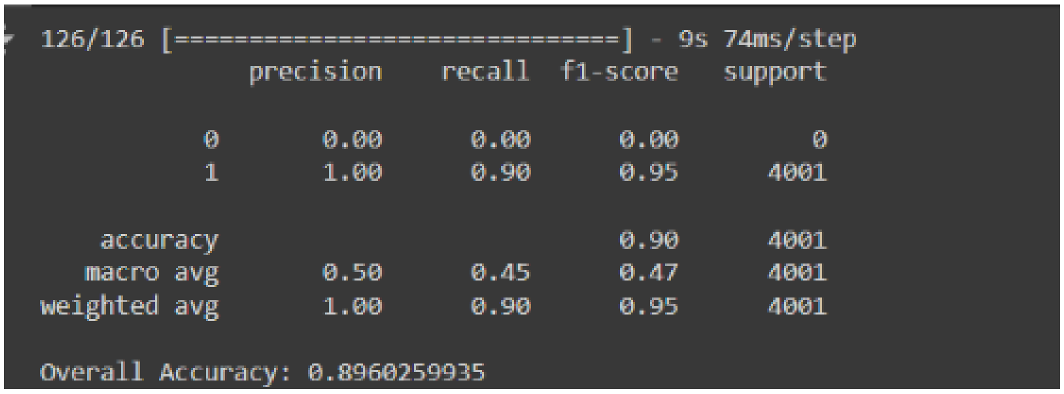
3. Bi LSTM



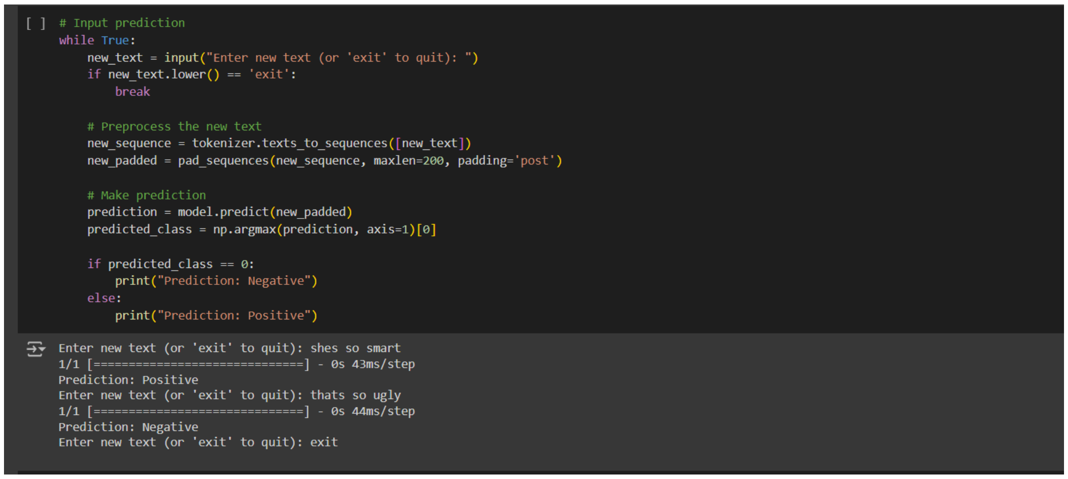


```
126/126 [==============================] - 7s 56ms/step
              precision    recall  f1-score   support

           0       1.00      0.90      0.95      4001
           1       0.00      0.00      0.00         0

    accuracy                           0.90      4001
   macro avg       0.50      0.45      0.47      4001
weighted avg       1.00      0.90      0.95      4001
```

4. Bi GRU



```
126/126 [==============================] - 9s 74ms/step
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      0.90      0.95      4001

    accuracy                           0.90      4001
   macro avg       0.50      0.45      0.47      4001
weighted avg       1.00      0.90      0.95      4001

Overall Accuracy: 0.8960259935
```

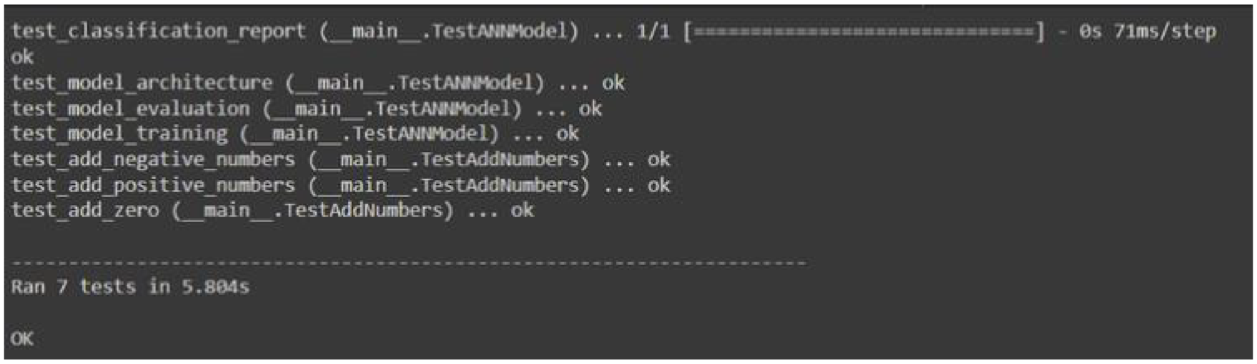


```
[ ] # Input prediction
    while True:
        new_text = input("Enter new text (or 'exit' to quit): ")
        if new_text.lower() == 'exit':
            break

        # Preprocess the new text
        new_sequence = tokenizer.texts_to_sequences([new_text])
        new_padded = pad_sequences(new_sequence, maxlen=200, padding='post')

        # Make prediction
        prediction = model.predict(new_padded)
        predicted_class = np.argmax(prediction, axis=1)[0]

        if predicted_class == 0:
            print("Prediction: Negative")
        else:
            print("Prediction: Positive")
```

```
Enter new text (or 'exit' to quit): shes so smart
1/1 [==============================] - 0s 43ms/step
Prediction: Positive
Enter new text (or 'exit' to quit): thats so ugly
1/1 [==============================] - 0s 44ms/step
Prediction: Negative
Enter new text (or 'exit' to quit): exit
```

5.2 Unit Testing:

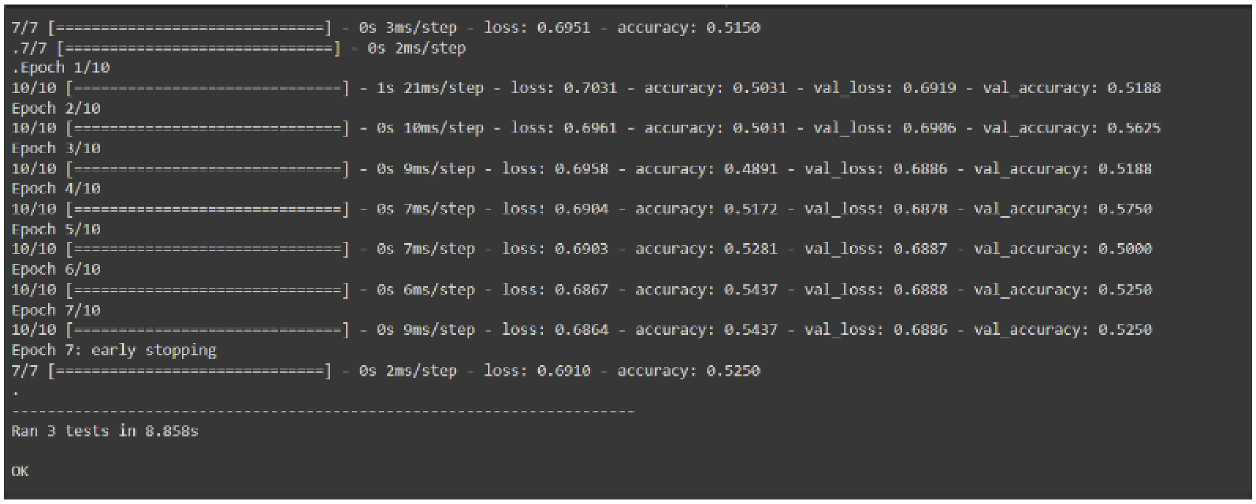
Unit testing is a software testing technique where individual components or units of a software application are tested in isolation to ensure they perform as expected. It involves testing each unit of the code independently to verify its functionality, typically using automated testing frameworks. The goal is to identify and fix bugs early in the development process and to ensure that each unit works correctly on its own before integrating it into the larger system.
This allows developers to quickly identify and fix any issues early in the development process, improving the overall quality of the software and reducing the time required for later testing.



```
test_classification_report (__main__.TestANNModel) ... 1/1 [==============================] - 0s 71ms/step
ok
test_model_architecture (__main__.TestANNModel) ... ok
test_model_evaluation (__main__.TestANNModel) ... ok
test_model_training (__main__.TestANNModel) ... ok
test_add_negative_numbers (__main__.TestAddNumbers) ... ok
test_add_positive_numbers (__main__.TestAddNumbers) ... ok
test_add_zero (__main__.TestAddNumbers) ... ok

----------------------------------------------------------------------
Ran 7 tests in 5.804s

OK
```

5.3 Integration Testing:

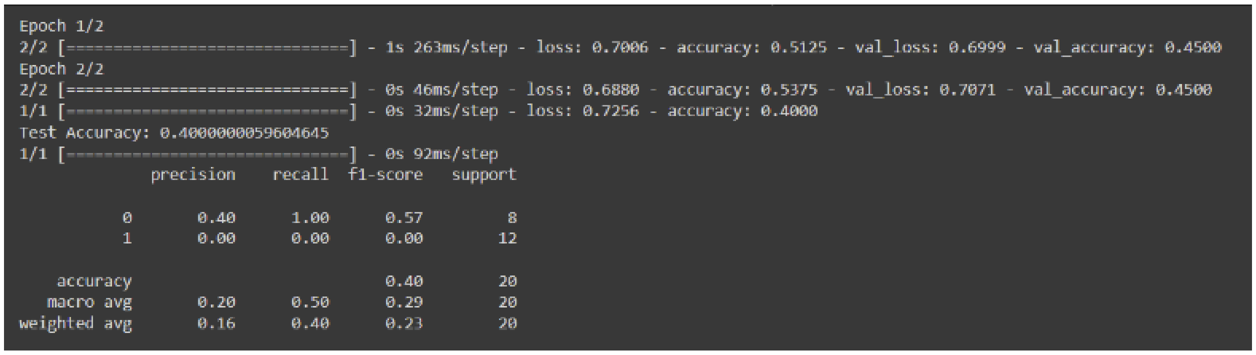
Integration testing is a software testing technique that focuses on verifying the interactions and data exchange between different components or modules of a software application. The goal of integration testing is to identify any problems or bugs that arise when different components are combined and interact with each other. Integration testing is typically performed after unit testing and before system testing. It helps to identify and resolve integration issues early in the development cycle, reducing the risk of more severe and costly problems later on.



```
7/7 [==============================] - 0s 3ms/step - loss: 0.6951 - accuracy: 0.5150
.7/7 [==============================] - 0s 2ms/step
.Epoch 1/10
10/10 [==============================] - 1s 21ms/step - loss: 0.7031 - accuracy: 0.5031 - val_loss: 0.6919 - val_accuracy: 0.5188
Epoch 2/10
10/10 [==============================] - 0s 10ms/step - loss: 0.6961 - accuracy: 0.5031 - val_loss: 0.6906 - val_accuracy: 0.5625
Epoch 3/10
10/10 [==============================] - 0s 9ms/step - loss: 0.6958 - accuracy: 0.4891 - val_loss: 0.6886 - val_accuracy: 0.5188
Epoch 4/10
10/10 [==============================] - 0s 7ms/step - loss: 0.6904 - accuracy: 0.5172 - val_loss: 0.6878 - val_accuracy: 0.5750
Epoch 5/10
10/10 [==============================] - 0s 7ms/step - loss: 0.6903 - accuracy: 0.5281 - val_loss: 0.6887 - val_accuracy: 0.5000
Epoch 6/10
10/10 [==============================] - 0s 6ms/step - loss: 0.6867 - accuracy: 0.5437 - val_loss: 0.6888 - val_accuracy: 0.5250
Epoch 7/10
10/10 [==============================] - 0s 9ms/step - loss: 0.6864 - accuracy: 0.5437 - val_loss: 0.6886 - val_accuracy: 0.5250
Epoch 7: early stopping
7/7 [==============================] - 0s 2ms/step - loss: 0.6910 - accuracy: 0.5250
.
----------------------------------------------------------------------
Ran 3 tests in 8.858s

OK
```

5.4 Smoke Testing:

Smoke testing, also called build verification testing or confidence testing, is a software testing method that is used to determine if a new software build is ready for the next testing phase. This testing method determines if the most crucial functions of a program work but does not delve into finer details.
As a preliminary check of software, smoke testing finds basic and critical issues in an application before more in-depth testing is done. If a piece of software passes a smoke test, quality assurance (QA) teams then proceed with further testing. A failure means the software must be handed back to the development team. The goal of smoke testing is to discover simple but severe failures using test cases that cover the most important functionalities of software. Smoke tests are performed by QA teams using a minimal set of tests on each build that focuses on software functionality.



```
Epoch 1/2
2/2 [==============================] - 1s 263ms/step - loss: 0.7006 - accuracy: 0.5125 - val_loss: 0.6999 - val_accuracy: 0.4500
Epoch 2/2
2/2 [==============================] - 0s 46ms/step - loss: 0.6880 - accuracy: 0.5375 - val_loss: 0.7071 - val_accuracy: 0.4500
1/1 [==============================] - 0s 32ms/step - loss: 0.7256 - accuracy: 0.4000
Test Accuracy: 0.4000000059604645
1/1 [==============================] - 0s 92ms/step
              precision    recall  f1-score   support

           0       0.40      1.00      0.57         8
           1       0.00      0.00      0.00        12

    accuracy                           0.40        20
   macro avg       0.20      0.50      0.29        20
weighted avg       0.16      0.40      0.23        20
```

5.5) Regression Testing:

Regression testing is a crucial aspect of software engineering that ensures the stability and reliability of a software product. It involves retesting the previously tested functionalities to verify that recent code changes haven't adversely affected the existing features.
By identifying and fixing any regression or unintended bugs, regression testing helps maintain the overall quality of the software. This process is essential for software development teams to deliver consistent and high-quality products to their users.

Chapter 6: Conclusion

Conclusively, the creation of the Cyberbullying Detection Project is a noteworthy progression towards guaranteeing safer online environments on diverse digital platforms. This project has effectively addressed the widespread problem of cyberbullying and produced an automated system that can reliably identify and mitigate harmful behavior in real-time.

A highly efficient detection mechanism has been produced by the system's integration of cutting-edge technologies, including deep learning (DL) architectures like ANN, Bi-GRU, LSTM, XLSTM, and Bi-LSTM and advanced machine learning (ML) models like GaussianNB, LogisticRegression, DecisionTree, AdaBoost, and RandomForest. With an astounding 92% accuracy rate, the ANN model shows how DL approaches can improve cyberbullying detection.

Thorough planning, data preprocessing, and Agile methodology-based iterative development have made it possible to create a solid platform that meets the requirements of both platform administrators and users. The effectiveness and dependability of the system have been confirmed by extensive testing, which includes unit, integration, regression, and performance testing. As a result of these efforts, a better option has been created, enhancing user safety and confidence in online interactions.

Future developments will strengthen the system's capabilities and promote a safer and better online experience. Examples of these developments include real-time monitoring, user feedback integration, and advanced analytics. This project report provides a comprehensive account of our journey, from requirements gathering and research to final development and deployment, showcasing our dedication to using cutting-edge technology to combat cyberbullying.

Chapter 7: References

1. Cyberbullying detection: advanced preprocessing techniques & deep learning architecture for Roman Urdu data Amirita Dewani, Mohsin Ali Memon & Sania Bhatti
2. Cyberbullying Detection on Social Networks UsingMachine Learning Approaches Md Manowarul Islam, Linta Islam, Arnisha Akter, Selina Sharmin, Uzzal Kumar Acharjee
3. An Application to Detect Cyberbullying Using Machine Learning and Deep Learning
4. TechniquesMitushi Raj, Samridhi Singh, Kanishka Solanki, and Ramani Selvanambicorresponding author
5. Cyberbullying detection solutions based on deep learning architectures by Celestine Iwendi, Gautam Srivastav
6. Cyber Bullying Detection on Social Media using Machine Learning - Aditya Desai, Shashank, Kalaskar, Omkar Kumbhar and Rashmi Dhumal
7. Cyberbullying detection: Utilizing social media features by Alican Bozyiğit, Semih Utku, Efendi Nasibov
8. A Human-Centered Systematic Literature Review of Cyberbullying Detection Algorithms by Seunghyun Kim, Afsaneh Razi, Gianluca Stringhini, Pamela J. Wisniewski, Munmun De Choudhury
9. Accurate Cyberbullying Detection and Prevention on Social Media - Andrea Perera , Pumudu Fernando
10. Cyberbullying Detection: Hybrid Models Based on Machine Learning and Natural Language Processing Techniques