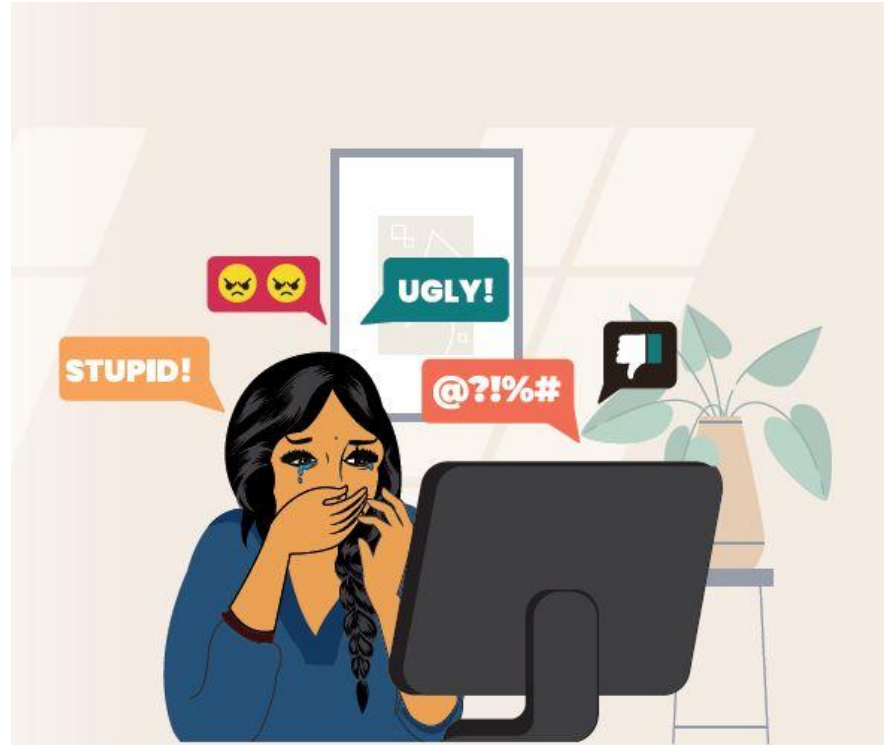


Cyberbullying Detection using Hybrid RNN and LSTM



Introduction

- Cyberbullying is a pervasive issue in today's digital era, particularly on social media platforms. This project aims to develop a model to detect cyberbullying based on comments extracted **[Web Scraping]** from *social media platforms* such as YouTube and Reddit.
- To accomplish this, **text cleaning** methods will be applied to remove noise from the data, including *special characters, emojis, and URLs*, followed by **tokenization** to break down the comments into analyzable components.
- The processed dataset will then be used to train a machine learning model, experimenting with algorithms like ***Logistic Regression, Support Vector Machines (SVM), and Decision Trees and neural networks*** to identify the most effective approach for detecting offensive and harmful content accurately.

Data Collection and Data Preprocessing

We collected data by web scraping comments from YouTube and Reddit. We used the **YouTube Data API** to extract comments and **PRAW** libraries for scraping Reddit. The collected data was stored as **CSV files** for further processing.

After collecting the data, we performed several **text cleaning** operations to remove unwanted characters, URLs, emojis, and punctuation. We also removed extra whitespaces to prepare the text for further analysis.

- **Module Used - Regular Expression (import re)**

Tokenization is the process of splitting text into smaller units, like words or phrases, to make it analyzable for machine learning models. It helps convert raw text into structured data that models can interpret effectively.

- **Modules Used - Natural Language Toolkit (Nltk) , tokenize , stopwords**

Removing **stopwords** involves eliminating common words like "and," "the," and "is" that don't add significant meaning to text analysis. This process helps improve model efficiency by focusing on more relevant words in the data.

Data Labeling

Manual Labeling

After cleaning the data, we manually labelled the comments as 'cyberbullying' or 'non-cyberbullying'. This labeling process was crucial in providing labeled training data for the machine learning model.

In labeling,

- 1 for Cyberbullying
- 0 for Non Cyberbullying

Initially, comments scraped from a single platform resulted in a ***biased dataset*** that didn't fully represent the diversity of online interactions. To create a more balanced and unbiased dataset, we *included comments from both YouTube and Reddit*, ensuring the model captures a broader range of language patterns and behaviors.

Automatic labeling

To automate labeling, we used the "**unitary/toxic-bert**" model to classify comments as "cyberbullying" or "non-cyberbullying" based on toxicity scores. Comments with scores above a set threshold are labeled as "cyberbullying".

Modules Used - AutoTokenizer, AutoModelForSequenceClassification , torch

Phase 1: Web Scraping & Data Preprocessing

- Retrieved data using the YouTube Data API v3 to gather video comments.
- Processed and cleaned the data by removing unwanted elements, normalizing text, and breaking down sentences into tokens.
- Performed initial data analysis to understand the comment patterns and potential challenges.

Phase 3: Model Training using Neural Network

- Trained the LSTM and RNN models on the updated dataset to improve the accuracy of cyberbullying detection.
- Incorporated additional data to refine the model's ability to classify comments effectively.
- Evaluated model performance and accuracy to assess detection improvements.

Phase 2: Data Splitting & Model Training

- Classified the data into two groups: Cyberbullying and Non-Cyberbullying.
- Used several classifiers, including Logistic Regression, SVM, and Random Forest, to train the model and assess performance.
- Tuned hyperparameters to optimize each model's accuracy and efficiency.

Phase 4: GUI Development with Flask & Gradio

- Built a user interface with Flask for interaction and model evaluation.
- Integrated the trained model within the web interface and performed tests with new data.
- Added functionalities for real-time feedback and error handling to improve user experience.



Model Training

- **Feature Extraction and Model Selection**

The preprocessed text data was converted into numerical format using **TF-IDF Vectorization**. Multiple models, including **Logistic Regression, Random Forest, and LSTM**, were trained to identify the most effective approach.

- **Evaluation and Integration**

The best-performing model, **LSTM**, was selected based on metrics like accuracy and F1-score. It was then integrated into the system to provide real-time, personalized career recommendations.

Modules Used for Model Training

- **Scikit-learn**: For traditional machine learning models like Logistic Regression and Random Forest, and evaluation metrics.
 - **TensorFlow/Keras**: For building and training deep learning models such as LSTM.
 - **Pandas**: For data manipulation and preparation.
 - **NumPy**: For numerical computations and matrix operations.
 - **Matplotlib/Seaborn**: For visualizing model performance metrics.
-



Advanced models used RNN-LSTM

- RNN is a type of neural network designed to process sequential data, such as time series, text, or audio.
- LSTM is a special type of RNN designed to overcome the limitations of standard RNNs.

WHY RNN-LSTM

- They are excellent at analyzing and predicting patterns in data.
 - LSTM is particularly effective in text-based tasks, as it can capture the context and relationships between words over long sentences.
 - LSTM captures subtle details in comments or messages by analyzing context, tone, and word sequence, thereby enhancing detection accuracy,
-

Statistical Analysis of Model training data

Model	Test Size (20%)	Test Size (25%)	Test Size (30%)
Logistic Regression	78.17	78.16	78.05
Random Forest Classifier	76.21	75.70	75.58
Support Vector Machine(SVM)	78.68	78.61	78.57
Decision Tree Classifier	74.88	75.33	74.45
K-Nearest Neighbours (K-NN)	65.29	59.11	64.74
Long Short-Term Memory(LSTM)	81.56	79.84	78.81
Recurrent Neural Network(RNN)	68.57	73.59	73.42

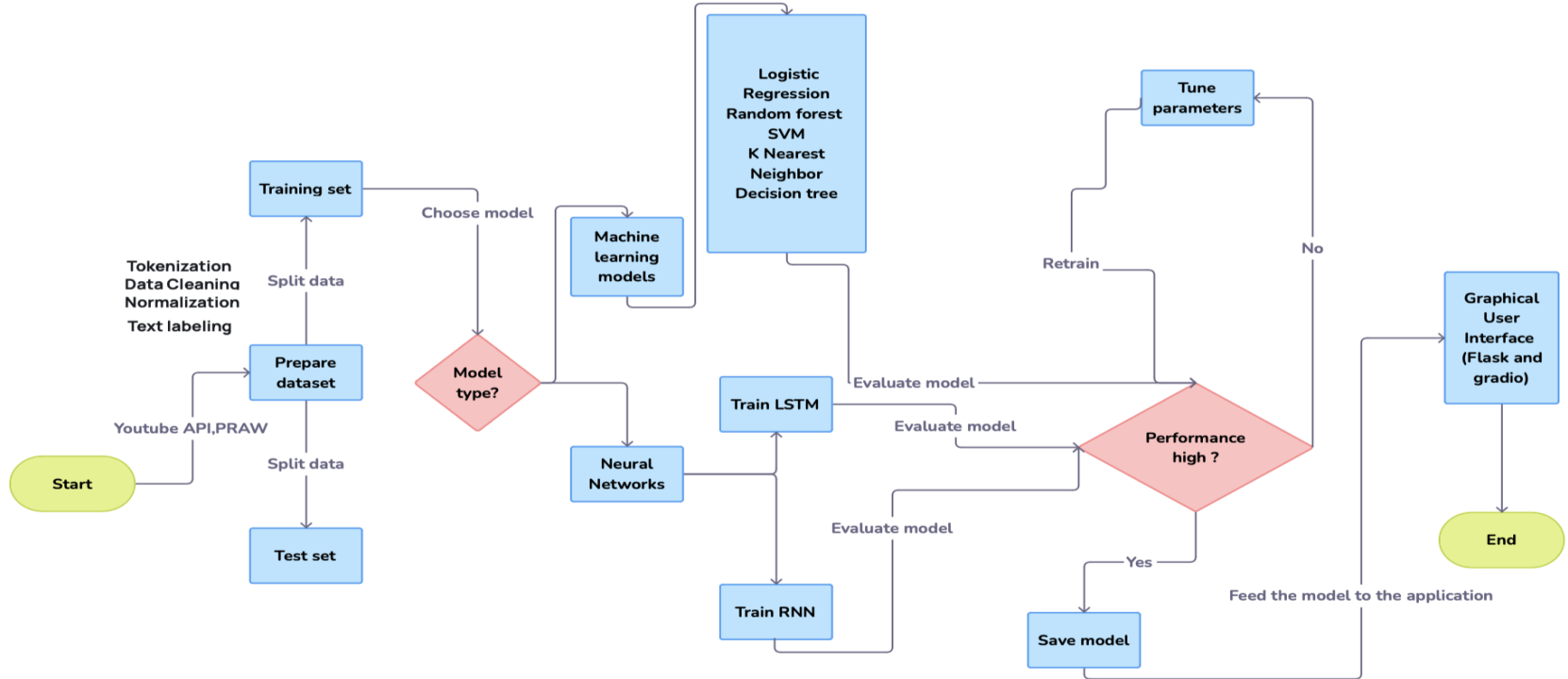
Label	Data_Count	Percentage
Non cyberbullying	9095	54.71%
Cyberbullying	7530	45.29%



Advantages of different models

- **Logistic Regression** is simple, efficient, and suitable for real-time applications, especially with large features.
 - **Random Forest classifier** offers high accuracy, is robust against overfitting, and handles noisy or missing data effectively.
 - **Support Vector Machine (SVM)** excels in high-dimensional data, is robust to overfitting, and flexible for both linear and non-linear classification.
 - **K-Nearest Neighbors (KNN)** is easy to implement, intuitive, and adaptable for varied datasets without a training phase.
 - **Decision Tree classifier** is highly interpretable, captures non-linear relationships, and efficiently handles both numerical and categorical data.
 - **Long Short-Term Memory (LSTM)**: Excels in processing and predicting sequential data, retaining long-term dependencies, and addressing the vanishing gradient problem effectively
 - **Recurrent Neural Network (RNN)**: Suitable for sequential and time-series data, capturing temporal patterns efficiently, and is adaptable for tasks like text, audio, and video analysis.
-

Workflow of the process



GUI development with flask & gradio

- **Flask for Backend:** Use Flask to handle user requests, manage routes, and integrate with the trained model for predictions and evaluations.
- **Gradio for Interface Prototyping:** Incorporate Gradio for rapid prototyping of the model's UI, enabling real-time testing of predictions with minimal effort.
- **Seamless Model Integration:** Both Flask and Gradio can easily integrate the trained machine learning model for smooth interaction and testing.

<https://cyberbullying-detection-app.onrender.com/>

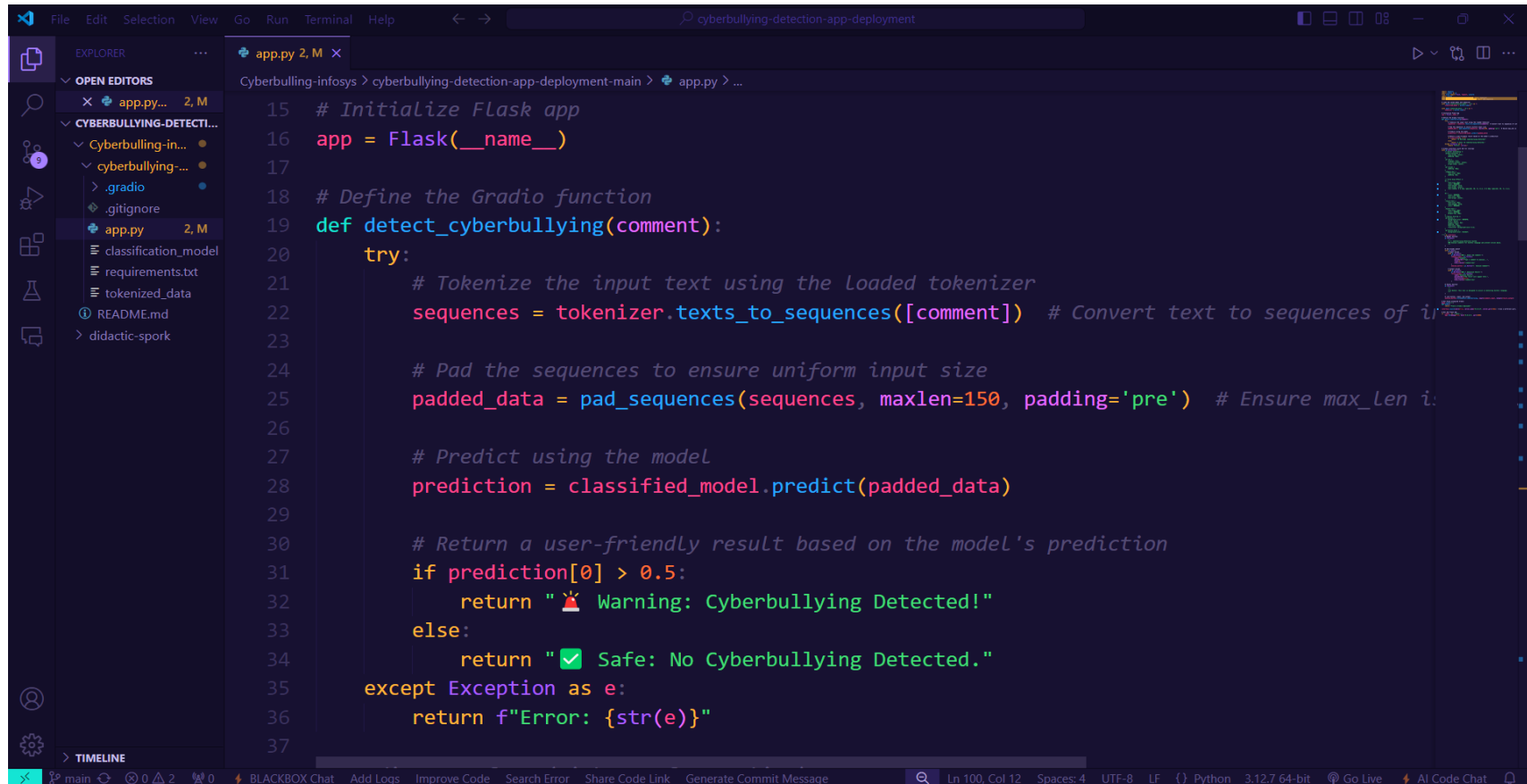
Software tools Used

- **Google Colab** - provide an online, cloud-based platform for writing and executing the code
- **TensorFlow/Keras** - For building deep learning models
- **scikit-learn** - For data preprocessing and model evaluation
- **pandas** - For data manipulation
- **Flask** - For deploying the trained model
- **matplotlib & seaborn** - For data visualization and model evaluation
- **Git LFS** - For managing large files like the trained model

VERSIONS:

- Python == 3.12.3
- Flask == 3.1.0
- Tensorflow == 2.18.0
- Keras == 3.7.0
- Scikit-learn == 1.5.2
- Pandas == 2.2.3
- Gradio == 5.6.0

Working of the frontend



The image shows a Visual Studio Code editor window with a Python file named `app.py` open. The file is part of a project named `cyberbullying-detection-app-deployment`. The code is a Flask application that uses a pre-trained model to detect cyberbullying in text comments. The code is written in Python and uses the Flask framework for web development. The code is as follows:

```
15 # Initialize Flask app
16 app = Flask(__name__)
17
18 # Define the Gradio function
19 def detect_cyberbullying(comment):
20     try:
21         # Tokenize the input text using the Loaded tokenizer
22         sequences = tokenizer.texts_to_sequences([comment]) # Convert text to sequences of integers
23
24         # Pad the sequences to ensure uniform input size
25         padded_data = pad_sequences(sequences, maxlen=150, padding='pre') # Ensure max_len is 150
26
27         # Predict using the model
28         prediction = classified_model.predict(padded_data)
29
30         # Return a user-friendly result based on the model's prediction
31         if prediction[0] > 0.5:
32             return "🚨 Warning: Cyberbullying Detected!"
33         else:
34             return "✅ Safe: No Cyberbullying Detected."
35     except Exception as e:
36         return f"Error: {str(e)}"
37
```

The code is written in Python and uses the Flask framework for web development. The code is as follows:

Working of the backend



Cyberbullying Detection using Lstm & RNN.ipynb ☆

File Edit View Insert Runtime Tools Help [Last edited on 26 November](#)



Connect ▾



+ Code + Text



Model training using LSTM

```
[ ] import numpy as np
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

```
[ ] # Load dataset
df = pd.read_csv('/content/cyberbullying_dataset.csv') # Replace with your dataset file
comments = df['Comments'].values
labels = df['label'].values
```

df.shape

(16625, 2)



```
[ ] comments = df['Comments'].astype(str).values # Convert the 'Comments' column to string type
labels = df['label'].values
```



```
[ ] tokenizer = Tokenizer(num_words=20000, oov_token="<OOV>") # Using 10k most frequent words
tokenizer.fit_on_texts(comments)
```



Cyberbullying Detection System

Analyze comments for harmful language and prevent online abuse.



Enter the Comment:

Comment

Type a comment to analyze...



Analyze Comment



Detection Result:



Note: This tool is designed to assist in detecting harmful language.

Final User Interface



Cyberbullying Detection System

Analyze comments for harmful language and prevent online abuse.



Enter the Comment:

Comment

you are so dumb!



Analyze Comment



Detection Result:



Warning: Cyberbullying Detected!



Note: This tool is designed to assist in detecting harmful language.

Cyberbullying detected



Cyberbullying Detection System

Analyze comments for harmful language and prevent online abuse.



Enter the Comment:

Comment

This looks interesting



Analyze Comment



Detection Result:



Safe: No Cyberbullying Detected.



Note: This tool is designed to assist in detecting harmful language.

No Cyberbullying detected

Conclusion

- **Web Scraping:** Data was collected from online platforms like YouTube and Reddit to gather real-world comments for analysis.
- **Labeling and Tokenization:** Comments were labeled as cyberbullying or non-cyberbullying, followed by text preprocessing, including tokenization and stopword removal.
- **Model Training:** Various machine learning models, including Logistic Regression, Random Forest, KNN, and SVM, were trained and evaluated for accurate cyberbullying detection.
- **GUI:** A user interface was built using Flask to facilitate interaction and model evaluation. The trained model was integrated within the web interface, enabling tests with new data. Additional functionalities, such as real-time feedback and error handling, were incorporated to enhance the user experience.

Team Members

- Sruthi Sai Prabha K S
- Kanika B
- Billa Nithin Reddy
- Mohan Prasath R
- Lakshmi Sowmya Mallela