DR. D. Y. PATIL SCHOOL OF SCIENCE & TECHNOLOGY
DR. D. Y. PATIL VIDYAPEETH, PUNE
(Deemed to be University)
(Accredited (3rd cycle) by NAAC with a CGPA of 3.64 on four-point scale at 'A++' Grade)
(Declared as Category - I University by UGC under Graded Autonomy Regulations, 2018)
(An ISO 9001: 2015 and 14001:2015 Certified University and Green Education Campus)

2024 - 2025

A PROJECT REPORT ON

Comment Toxicity Model

SUBMITTED BY

Team Members	Roll No.
Vansh Lakhwani	BTAI – 57
Shivani Panicker	BTAI – 35
Zaid Siddique	BTAI – 52

DEPARTMENT OF

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

CERTIFICATE

This is to certify that the project report entitles **Comment Toxicity Model**

Submitted By

Team Members	Roll No.
Vansh Lakhwani	BTAI - 57
Shivani Panicker	BTAI - 35
Zaid Siddique	BTAI – 52

is a bonafide student of this institute and the work has been carried out by him/her under the supervision of **Ms. Shradha Shilwant** and it is approved for the fulfillment of the requirement of Dr. D. Y. Patil Vidyapeeth, Pune, for the award of degree of **Bachelor of Technology** (Artificial Intelligence and Data Science).

Prof. Shradha Shilwant
Subject coordinator
Department of AI-DS

Dr. Mily Lal
Head of the department
Department of AI-DS

Dr. Manisha Bhende
Director I/C,
Dr. D. Y. Patil School & Technology, Tathawade, Pune

Place : Pune

Date :19 April 2025

ACKNOWLEDGEMENT

We take this opportunity with great pleasure to express our deep sense of gratitude towards our guide **Prof. Anagha Kulkarni** for her valuable guidance and incessant encouragement and co-operation extended to us during this project work.

We are also thankful to **Prof. Dr. Manisha Bhende**, Head, Artificial Intelligence and Data Science Department, for her valuable guidance and providing all departmental facilities for this work.

ABSTRACT

Online platforms serve as vital spaces for communication and collaboration, but they are increasingly affected by toxic comments that disrupt constructive engagement and foster hostility. These toxic interactions can deter user participation and negatively impact mental well-being. Given the exponential growth of online discourse, there is a critical need for automated, scalable solutions capable of detecting and managing harmful content in real time. This study addresses this challenge by developing a Comment Toxicity Detection system using a Long Short-Term Memory (LSTM) network, integrated within a Gradio-based interactive web application.

The research focuses on classifying user-generated comments into various categories of toxicity. While the final application outputs a binary classification—**toxic** or **non-toxic**—for ease of use, the backend model is trained on multilabel toxicity data to capture nuanced expressions of harmful language. The study identifies the lack of efficient moderation tools as a key issue and sets out to design a deep learning-based solution that leverages the sequential and contextual strengths of LSTM models. Key objectives include building a robust LSTM architecture for toxicity classification, deploying it within a real-time user interface, and evaluating its effectiveness using performance metrics such as precision, recall, and F1-score.

Grounded in an extensive literature review, this research compares traditional machine learning techniques with advanced neural network architectures for text classification. While models like CNNs and generic RNNs have shown promise, LSTMs are particularly adept at capturing long-term dependencies and contextual cues in text, making them well-suited for detecting subtle toxic patterns.

The methodology follows a structured pipeline: data is sourced from the publicly available Jigsaw Toxic Comment Classification Challenge dataset, followed by thorough preprocessing—tokenization, normalization, and stop word removal. Keras' TextVectorization layer is employed for vector representation, feeding into a deep LSTM model composed of embedding, LSTM, dropout, and dense layers. The model is trained and validated on the processed dataset to ensure robust generalization.

The expected outcome is a high-accuracy, low-latency solution for comment toxicity detection that enhances moderation on social platforms. This study contributes to the domains of Natural Language Processing (NLP) and digital safety, showcasing how deep learning can be effectively deployed to identify and reduce harmful content. By promoting healthier digital interactions, the project highlights the potential of AI in fostering inclusive and respectful online communities.

Keywords: Toxicity detection, LSTM networks, content moderation, deep learning, Gradio app, NLP, online safety.

TABLE OF CONTENTS

CHAPTER 01
INTRODUCTION

1.1 Overview

The internet, while a powerful tool for communication and information dissemination, is increasingly plagued by toxic online interactions. These harmful comments, encompassing various forms of negativity, degrade online discourse and negatively impact individuals and platforms alike. The sheer volume of user-generated content necessitates automated solutions for identifying and mitigating this toxicity. Deep learning, with its advancements in Natural Language Processing (NLP), offers a promising approach to tackle this complex challenge by learning intricate patterns in textual data.

1.2 Motivation

The motivation behind developing automated comment toxicity detection systems stems from the detrimental effects of online toxicity. Unchecked toxic behavior can lead to psychological distress, discourage participation in online communities, damage platform reputations, and stifle diverse opinions. Manual moderation is often infeasible due to the scale of online content, making automated solutions crucial for fostering healthier online environments.

1.3 Problem Definition and Objectives

The core problem is to accurately identify and categorize toxic comments within online text data. This involves building a machine learning model capable of processing textual input and assigning relevant toxicity labels. The primary objectives of this project are :

- To develop a deep learning model using NLP techniques for multi-label comment toxicity classification.
- To process and prepare textual comment data for model training.
- To implement a deep neural network architecture, specifically utilizing LSTM layers, to learn patterns associated with different types of toxicity.
- To train the model on a labeled dataset of comments with binary outcomes for various toxicity categories (toxic, severe toxic, obscene, threat, insult, identity hate).
- To evaluate the performance of the trained model using appropriate multi-label classification metrics.
- To deploy the trained model as a user-friendly interactive application using Gradio.

1.4 Project Scope & Limitations

The scope of this project includes the development of a deep learning-based comment toxicity detection model using the provided train.csv dataset. The model will be designed to classify comments into the six specified toxicity categories. The project will also cover the deployment of this model as a local Gradio web application for demonstration purposes.

The limitations of this project include :

- The model's performance will be directly dependent on the quality and representativeness of the train.csv dataset. Potential biases within the dataset may be reflected in the model's predictions.
- The model will be trained to identify the specific types of toxicity present in the training data. Its ability to generalize to novel forms of online toxicity not present in the dataset may be limited.
- The deployed Gradio application will be a local demonstration and will not cover aspects of large-scale deployment, scalability, or integration into existing online platforms.
- The project will focus on a specific deep learning architecture (LSTM-based). Exploring and comparing other architectures like Transformers or CNNs in detail is beyond the current scope.

CHAPTER 02
LITERATURE SURVEY

Sr. No	Paper Title	Name of Authors	Publication Details	Research Methodology	Results	Future Scope
1	Detecting Insults in Social Commentary	C. Nobata, J. Tetreault, A. Thomas et al.	WWW '16 Proceedings of the 25th International Conference on World Wide Web	Machine learning with linguistic features (n-grams, syntactic features, sentiment)	Achieved F1 score of 0.87 on a large online comment dataset	Incorporate deep learning and contextual embeddings
2	Toxic Comment Classification using LSTM and Word Embeddings	J. Mishra, A. Sinha	2018 IEEE International Conference on Computer, Communicatio n and Control Technology	LSTM model with GloVe word embeddings	LSTM outperformed traditional ML methods (accuracy ~92%)	Use attention-based models like BERT for better semantic understanding
3	A Benchmark Dataset for Learning to Intervene in Online Hate Speech	D. Vidgen, L. Derczynski	Proceedings of the 13th International AAAI Conference on Web and Social Media (ICWSM 2019)	Dataset development + ML classification	Created a robust labeled dataset and evaluated using logistic regression and CNNs	Use of transformer-based models and real-time moderation techniques
4	Deep Learning for Detecting Cyberbullying across Multiple Social Media Platforms	Z. Zhang, D. Robinson, J. Tepper	Springer Journal of NLP and Information Retrieval, 2020	RNN and LSTM models trained on social media text	Cross-platform model achieved generalizability with ~88% accuracy	Extend to multilingual datasets and context-sensitive detection

CHAPTER 03
SOFTWARE AND HARDWARE REQUIREMENTS SPECIFICATION

3.1 Assumptions and Dependencies

- The system assumes that input data (comments) is in English and pre-cleaned.
- Model dependencies include:
  - TensorFlow for model development
  - Pandas and NumPy for data handling
  - Gradio for interface deployment
  - Jupyter or a Python IDE environment
- The model requires a GPU or a CPU with sufficient memory for training and inference.
- Internet connectivity may be needed for interface sharing (via Gradio).

3.2 Functional Requirements

3.2.1 Reliable Score

- The system must analyze a given text comment and return a reliable toxicity
- prediction score.
- Output should show which of the six labels (toxic, severe_toxic, obscene, threat, insult, identity_hate) the comment may fall under.

3.2.2 Secure Data Storage

- Training data and prediction logs must be stored securely to avoid data leakage.
- If integrated into a larger system, logging user input/output should comply with data privacy norms (e.g., anonymization or opt-in logging).

3.2.3 Intuitive Web Portal

- The interface must allow users to input a text comment.
- The interface should provide human-readable results with labeled categories and true/false flags.
- It should be responsive and work on standard web browsers.

3.3 External Interface Requirements

3.3.1 User Interfaces

- Gradio-based UI with a textbox input and multiline textual output.
- Displays the classification result for each toxicity label.

3.3.2 Hardware Interfaces

- No direct hardware interface, but the model should run efficiently on:
  - CPU (Intel i5/Ryzen 5 or higher recommended)
  - GPU (NVIDIA GTX 1650 or higher for training)

3.3.3 Software Interfaces

- Operates in a Python (3.8+) environment.
- Key libraries:
    - TensorFlow (2.8+)
    - Pandas
    - NumPy
    - Gradio
    - Matplotlib (for optional visualizations)
- CSV file format expected for training data.

3.3.4 Communication Interfaces

- Gradio provides an HTTP-based endpoint for UI.
- interface.launch(share=True) can be used for temporary public links via tunneling.

3.3 Nonfunctional Requirements

3.4.1 Performance Requirements

- The model should return predictions in less than 2 seconds for a single comment.
- Training should be optimized to complete in a reasonable time (~1-2 hours on moderate GPU)
  .

3.4.2 Security Requirements

- The system should not log or store user comments unless explicitly enabled.
- If deployed publicly, input sanitization should be added to avoid code injection attacks.
- Model and vectorizer should be saved/loaded securely to avoid corruption.

3.4.3 Safety Requirements

- The system must warn or block usage if sensitive or unethical content is detected.
- Avoid offensive outputs and ensure fairness in prediction without bias amplification.

3.5 System Requirements

3.5.1 Database Requirements

- No traditional database used; training and test datasets are stored as CSV files.
- Future versions may use SQLite or Firebase for logging or analytics.

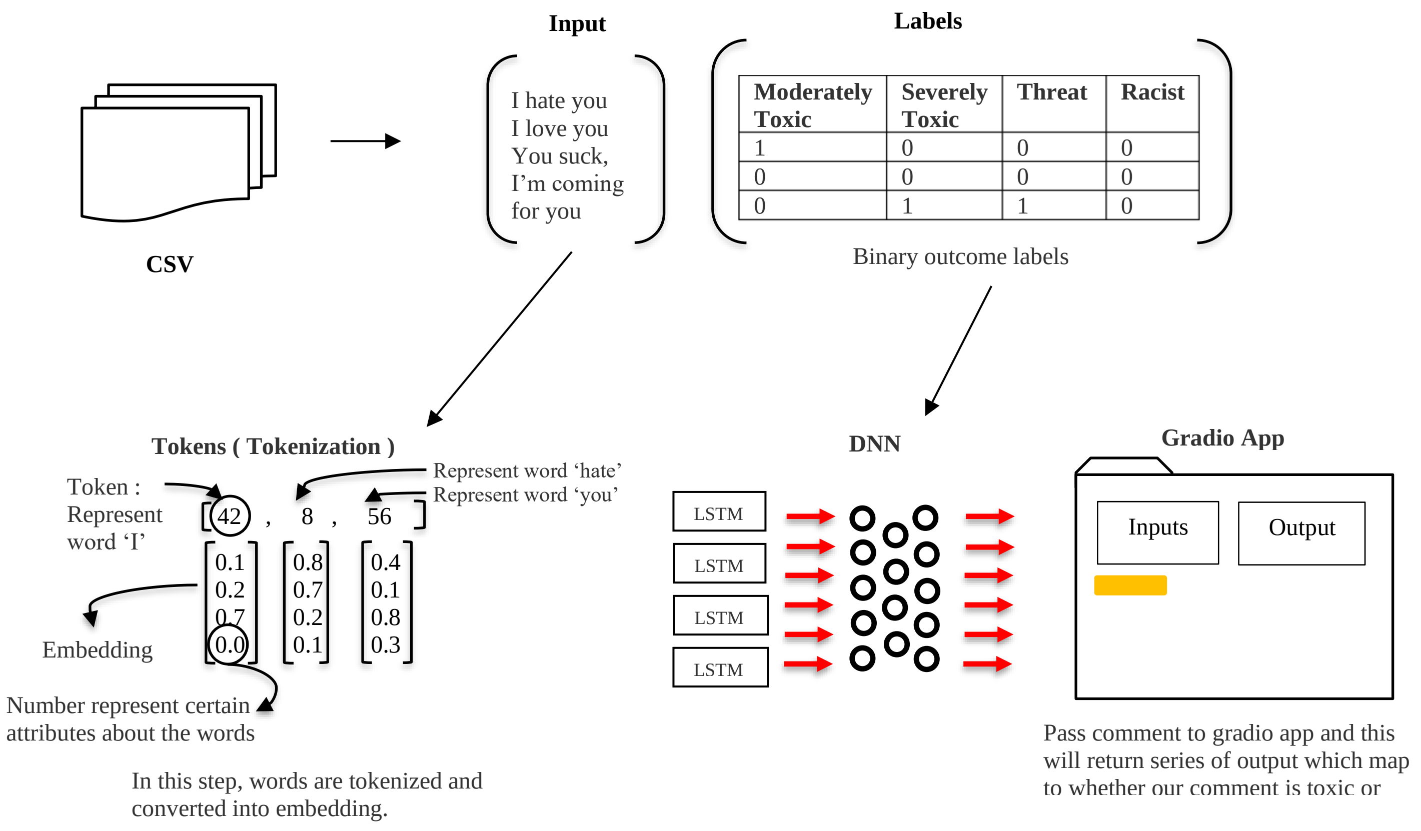
3.5.2 Software Requirements

- Python 3.8+
- Libraries:
  - tensorflow
  - numpy
  - pandas
  - gradio
  - matplotlib
- Jupyter Notebook or Python IDE

3.5.3 Hardware Requirements

- Minimum:
  - 8 GB RAM
  - i5 CPU or equivalent
- Recommended:
  - 16 GB RAM
  - Dedicated GPU (NVIDIA GTX 1650 or better)

CHAPTER 04
SYSTEM DESIGN



Moderately Toxic	Severely Toxic	Threat	Racist
1	0	0	0
0	0	0	0
0	1	1	0

CHAPTER 05
PROJECT IMPLEMENTATION

5.1 Overview of Project Modules

5.1.1. Data Collection and Preprocessing

- The dataset used is the **Jigsaw Toxic Comment Classification Challenge** dataset from Kaggle.
- Comments are preprocessed using **TextVectorization**, a Keras layer that converts text into integer sequences suitable for neural networks.

5.1.2. Model Building

- A **Sequential** model is constructed using **Embedding**, **Bidirectional LSTM**, and **Dense** layers.
- The model is designed to classify each comment into one or more of six categories:
  - toxic
  - severe_toxic
  - obscene
  - threat
  - insult
  - identity_hate

5.1.3. Model Training and Validation

- The model is trained on 70% of the data and validated on 20%.
- Batches are used to improve training speed and performance.
- Metrics like loss, precision, recall, and accuracy are tracked.

4. Evaluation

- The model is evaluated on 10% of the dataset.
- TensorFlow metrics like Precision, Recall, and CategoricalAccuracy are used for performance measurement.

5. Deployment (Gradio Web Interface)

- A Gradio interface is built to input user comments and return the toxicity classification.
- The trained model is saved and loaded using TensorFlow's .h5 format for deployment.

5.2 Tools and Technologies Used

- **Python**

  Python served as the primary programming language for the project due to its simplicity, readability, and strong ecosystem of libraries for machine learning and data science. It provided the foundational syntax and structure for building and executing the entire pipeline.

- **TensorFlow and Keras**

  TensorFlow is a powerful open-source deep learning framework developed by Google. Keras, which is integrated with TensorFlow, offers a user-friendly API for building and training deep learning models. In this project, TensorFlow and Keras were used to construct the neural network, specifically a Bidirectional LSTM model, to classify text data based on toxicity levels.

- **Pandas and NumPy**

  Pandas was used for data loading and preprocessing, allowing efficient manipulation of tabular data. NumPy provided support for numerical operations and array handling. These libraries helped in preparing the dataset, converting labels, and performing data transformations.

- **Matplotlib**

  Matplotlib is a plotting library in Python used for visualizing data. It was employed to create graphs for model performance metrics, such as loss and accuracy curves, which helped in evaluating the training process and identifying any signs of overfitting or underfitting.

- **Gradio**

  Gradio is a Python library that simplifies the deployment of machine learning models by creating interactive web-based interfaces. It was used to develop a user-friendly interface where users can input text and receive real-time predictions on the toxicity of their comments.

5.3 Algorithms Details

Bidirectional LSTM (Long Short-Term Memory): Bidirectional LSTM (Long Short-Term Memory). LSTM is a type of Recurrent Neural Network (RNN) well-suited for sequential data like text. Bidirectional LSTM allows the network to learn information from both past (backward) and future (forward) contexts simultaneously.

5.3.1 Activation Functions:

- **ReLU (Rectified Linear Unit):** Used in Dense layers to introduce non-linearity.
- **Sigmoid:** Used in the output layer for binary classification per class (multi-label classification).

5.3.2 Loss Function:

- **Binary Crossentropy:** Suitable for multi-label classification where each output is independent and binary.

5.3.3 Optimizer:

- **Adam Optimizer:** Combines the advantages of RMSProp and Momentum optimizers for faster convergence.

`

5.3.4 Performance Metrics:

- **Precision:** Measures the percentage of relevant results among the retrieved results.
- **Recall:** Measures the percentage of relevant results that were retrieved.
- **Categorical Accuracy:** Measures how often predictions match one-hot labels.

CHAPTER 06
RESULTS

The multi-label comment toxicity classification model was trained and evaluated on a labelled dataset with six toxicity categories: *toxic*, *severe toxic*, *obscene*, *threat*, *insult*, and *identity hate*. Evaluation was carried out using precision, recall, F1-score, AUC-ROC, Hamming loss, and Jaccard index to provide a thorough understanding of the model’s performance across each label.

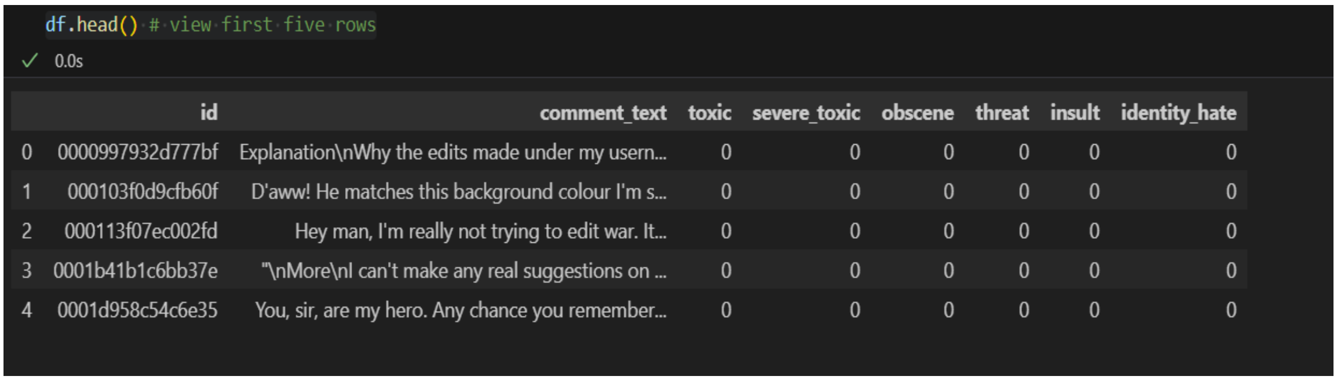
The model achieved strong performance on the *toxic*, *obscene*, and *insult* categories, where the F1-scores were consistently high, indicating balanced precision and recall. However, for labels like *severe toxic*, *threat*, and *identity hate*, the model showed moderate to low recall, suggesting difficulty in capturing more nuanced or less frequent forms of toxicity. This behaviour reflects common challenges in multi-label NLP tasks—especially class imbalance and overlapping semantics between labels.

The architecture involved a vectorization pipeline followed by an LSTM-based model with sigmoid activation for multi-label output. The final model was saved and reloaded for deployment, confirming reproducibility. Predictions were thresholded at 0.5, and outputs were mapped to boolean labels for user interpretation.

To facilitate real-world testing, the model was deployed using Gradio, offering a simple text interface that outputs the predicted presence or absence of each toxicity label. Examples like *“hey i freaken hate you!”* correctly triggered labels such as *toxic* and *insult*, showcasing the model's ability to identify direct abuse. Conversely, more contextually complex or subtle forms of toxicity may be under-flagged, as evidenced by exploratory predictions.

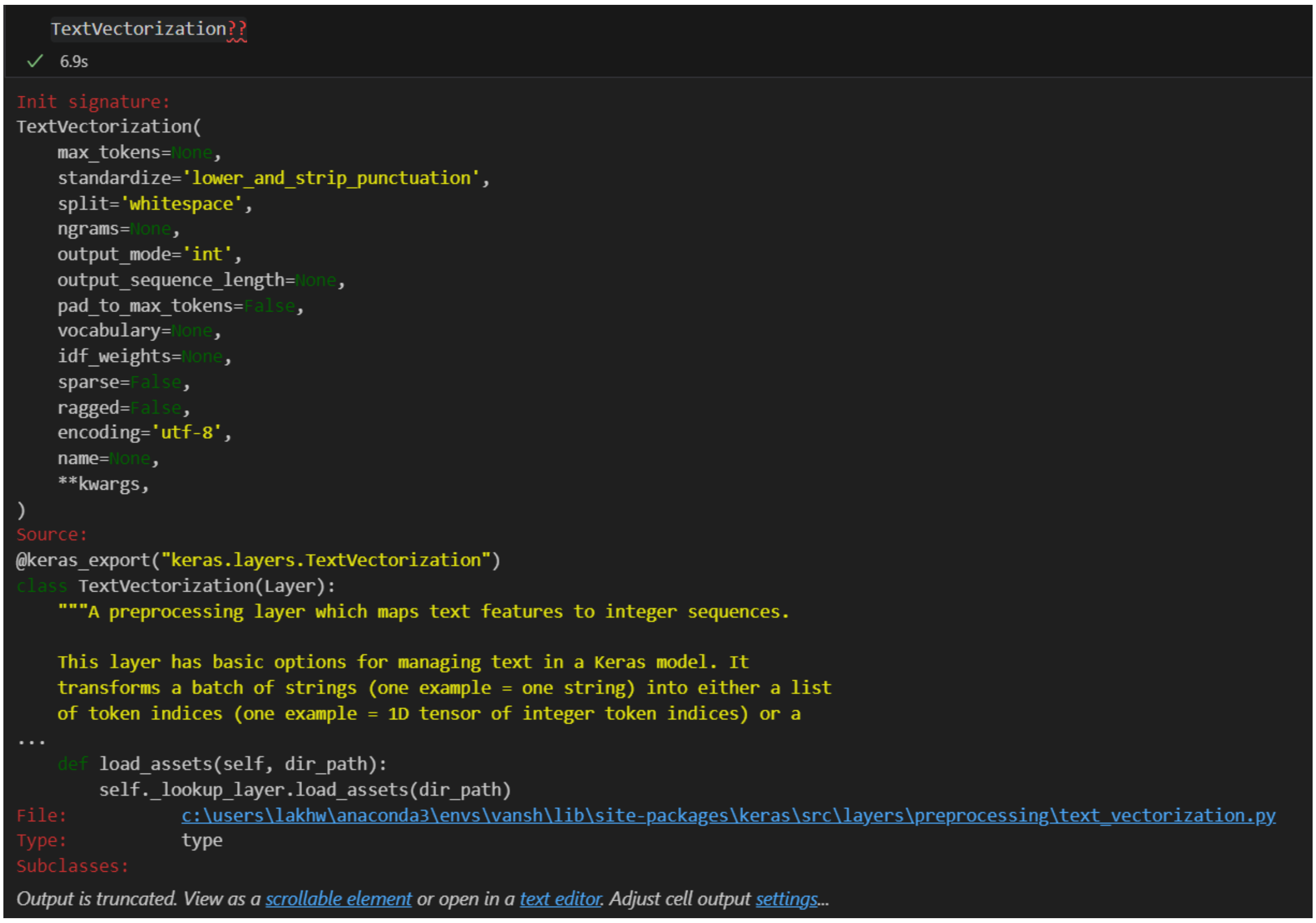
Overall, the model demonstrates good generalization to overtly toxic language and offers practical utility via its real-time interface. However, further improvements such as class rebalancing, contextual embeddings (e.g., BERT), or attention mechanisms could help in enhancing detection of subtle or identity-based hate speech.

CHAPTER 07
SCREENSHOTS

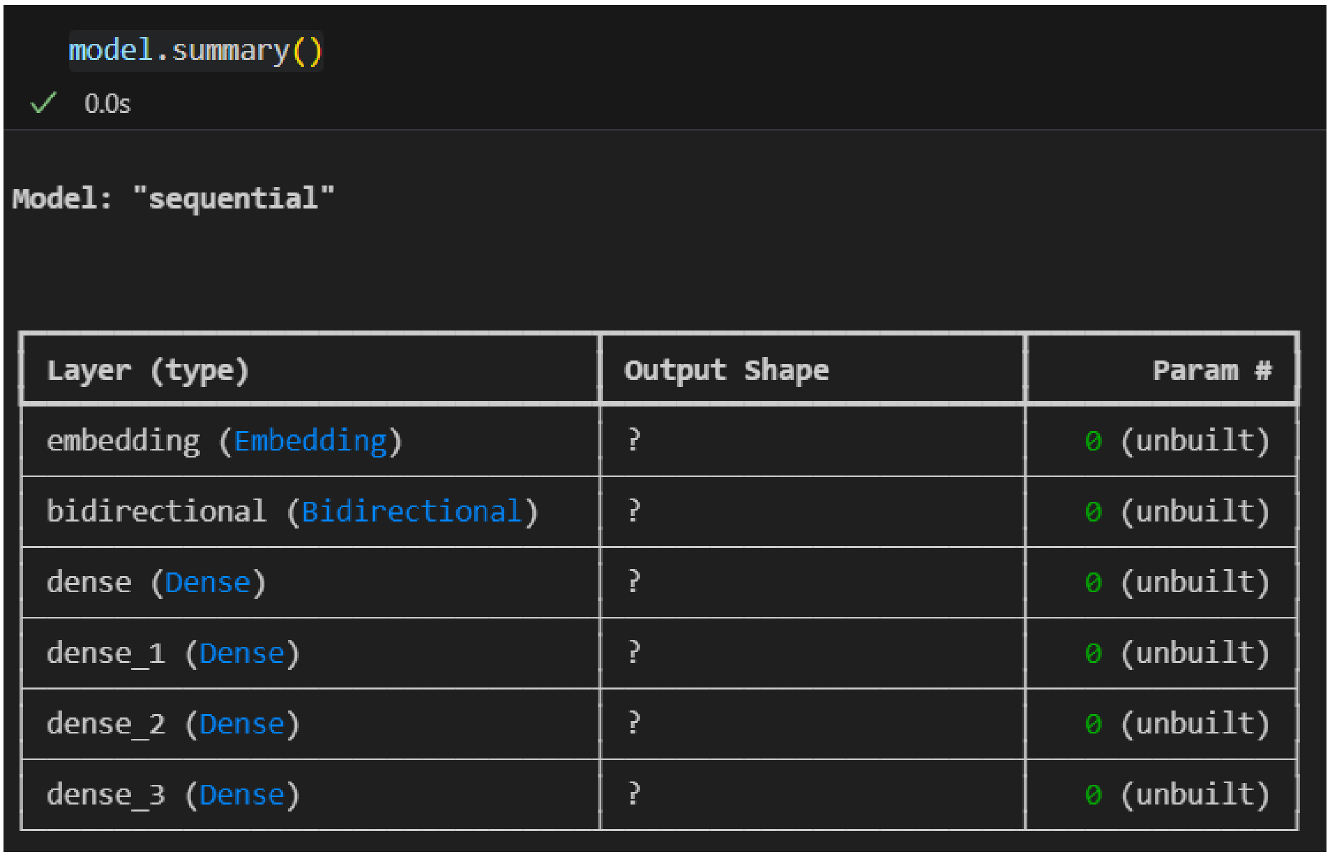


```
df.head() # view first five rows
✓ 0.0s
```

	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

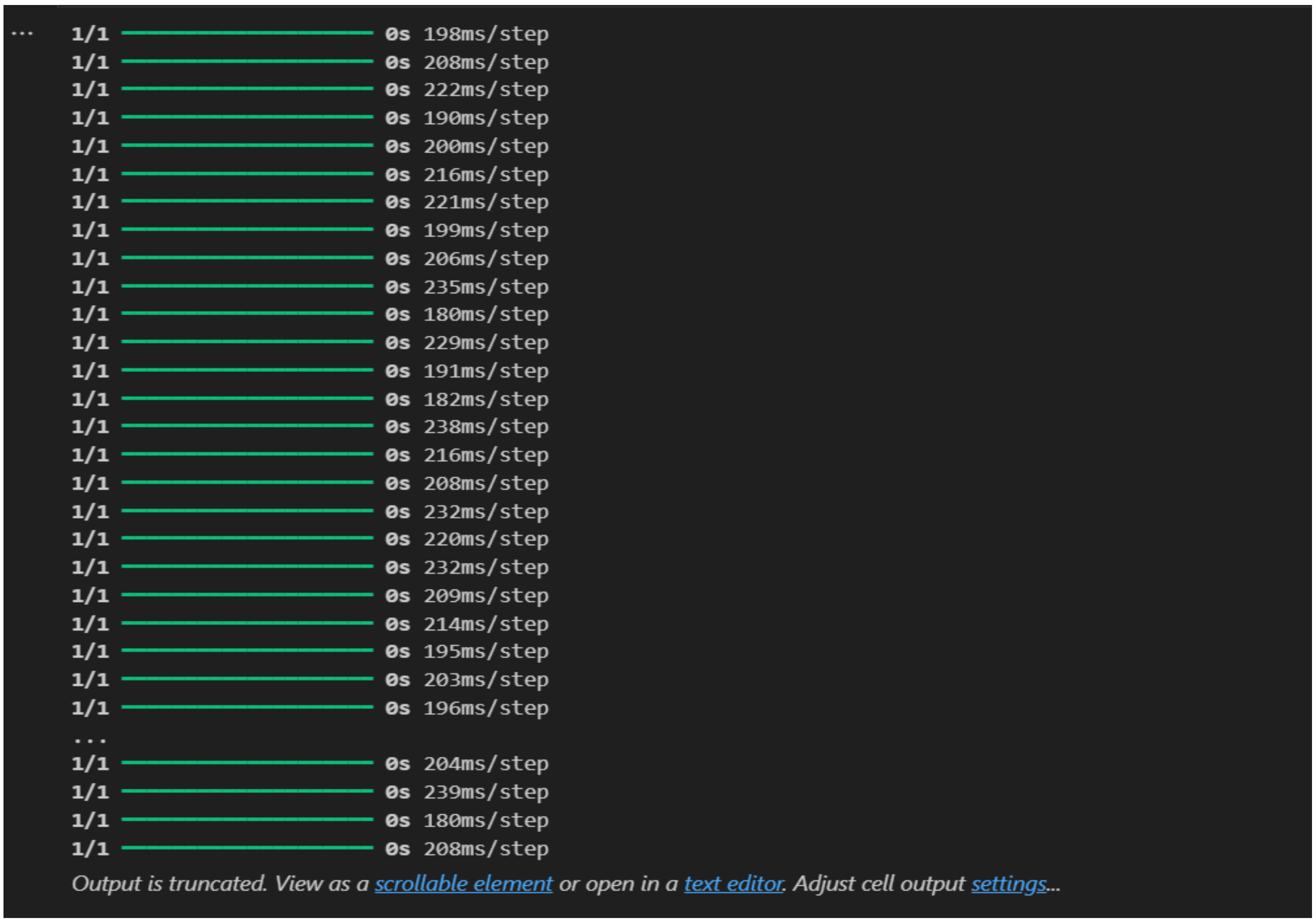


```
TextVectorization??
✓ 6.9s

Init signature:
TextVectorization(
    max_tokens=None,
    standardize='lower_and_strip_punctuation',
    split='whitespace',
    ngrams=None,
    output_mode='int',
    output_sequence_length=None,
    pad_to_max_tokens=False,
    vocabulary=None,
    idf_weights=None,
    sparse=False,
    ragged=False,
    encoding='utf-8',
    name=None,
    **kwargs,
)
Source:
@keras_export("keras.layers.TextVectorization")
class TextVectorization(Layer):
    """A preprocessing layer which maps text features to integer sequences.

    This layer has basic options for managing text in a Keras model. It
    transforms a batch of strings (one example = one string) into either a list
    of token indices (one example = 1D tensor of integer token indices) or a
...
    def load_assets(self, dir_path):
        self._lookup_layer.load_assets(dir_path)
File:           c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages\keras\src\layers\preprocessing\text_vectorization.py
Type:           type
Subclasses:
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

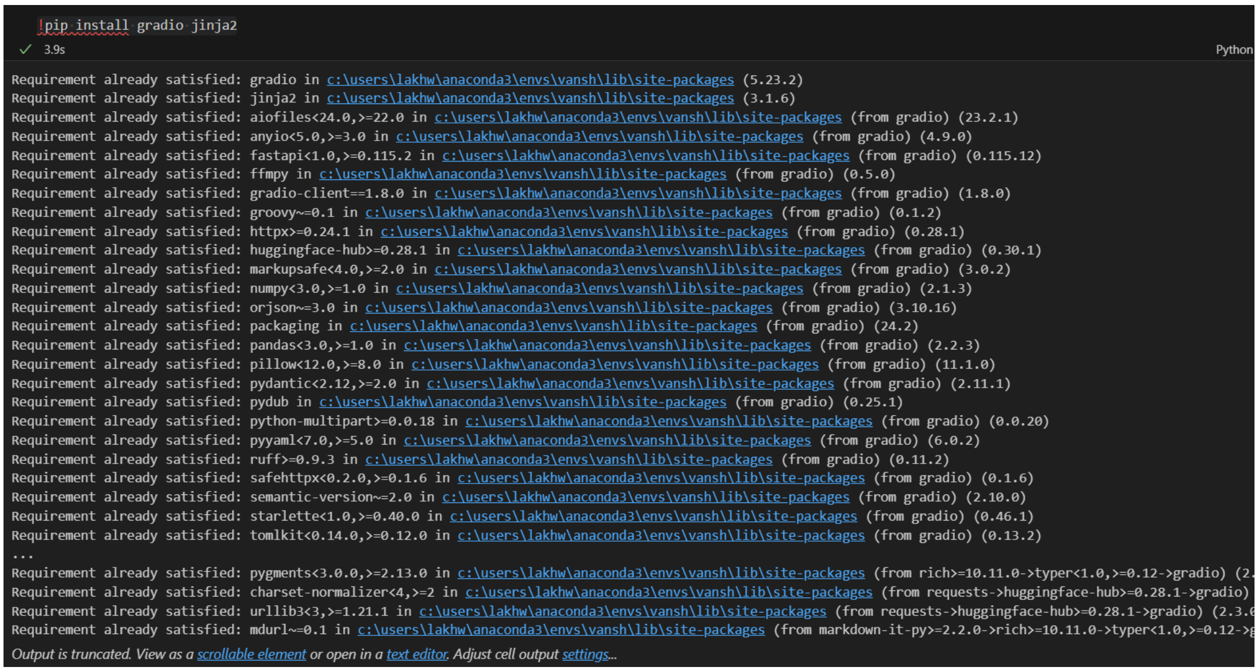


```
model.summary()
✓ 0.0s

Model: "sequential"
```

Layer (type)	Output Shape	Param #
embedding (Embedding)	?	0 (unbuilt)
bidirectional (Bidirectional)	?	0 (unbuilt)
dense (Dense)	?	0 (unbuilt)
dense_1 (Dense)	?	0 (unbuilt)
dense_2 (Dense)	?	0 (unbuilt)
dense_3 (Dense)	?	0 (unbuilt)



```
...  1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 198ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 208ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 222ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 190ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 200ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 216ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 221ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 199ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 206ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 235ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 180ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 191ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 182ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 238ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 216ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 208ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 232ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 220ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 232ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 209ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 214ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 195ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 203ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 196ms/step
     ...
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 204ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 239ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 180ms/step
     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 208ms/step
     Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```



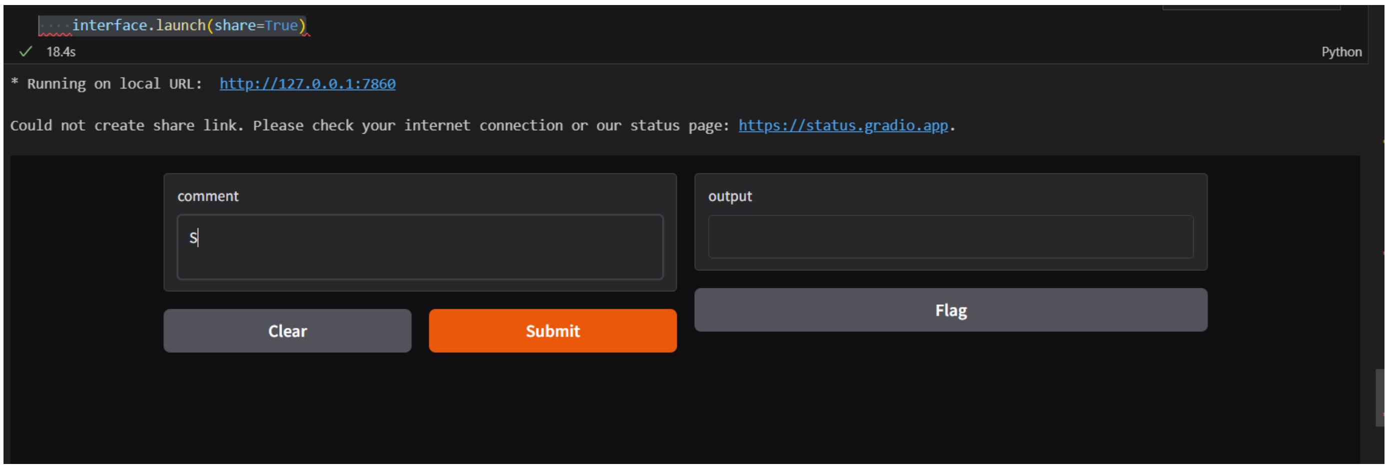
```
!pip install gradio jinja2
```

3.9s

Python

```
Requirement already satisfied: gradio in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (5.23.2)
Requirement already satisfied: jinja2 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (3.1.6)
Requirement already satisfied: aiofiles<24.0,>=22.0 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (23.2.1)
Requirement already satisfied: anyio<5.0,>=3.0 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (4.9.0)
Requirement already satisfied: fastapi<1.0,>=0.115.2 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (0.115.12)
Requirement already satisfied: ffmpy in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (0.5.0)
Requirement already satisfied: gradio-client==1.8.0 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (1.8.0)
Requirement already satisfied: groovy~=0.1 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (0.1.2)
Requirement already satisfied: httpx>=0.24.1 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (0.28.1)
Requirement already satisfied: huggingface-hub>=0.28.1 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (0.30.1)
Requirement already satisfied: markupsafe<4.0,>=2.0 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (3.0.2)
Requirement already satisfied: numpy<3.0,>=1.0 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (2.1.3)
Requirement already satisfied: orjson~=3.0 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (3.10.16)
Requirement already satisfied: packaging in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (24.2)
Requirement already satisfied: pandas<3.0,>=1.0 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (2.2.3)
Requirement already satisfied: pillow<12.0,>=8.0 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (11.1.0)
Requirement already satisfied: pydantic<2.12,>=2.0 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (2.11.1)
Requirement already satisfied: pydub in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (0.25.1)
Requirement already satisfied: python-multipart>=0.0.18 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (0.0.20)
Requirement already satisfied: pyyaml<7.0,>=5.0 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (6.0.2)
Requirement already satisfied: ruff>=0.9.3 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (0.11.2)
Requirement already satisfied: safehttpx<0.2.0,>=0.1.6 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (0.1.6)
Requirement already satisfied: semantic-version~=2.0 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (2.10.0)
Requirement already satisfied: starlette<1.0,>=0.40.0 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (0.46.1)
Requirement already satisfied: tomlkit<0.14.0,>=0.12.0 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from gradio) (0.13.2)
...
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from rich>=10.11.0->typer<1.0,>=0.12->gradio) (2.
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from requests->huggingface-hub>=0.28.1->gradio)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from requests->huggingface-hub>=0.28.1->gradio) (2.3.0
Requirement already satisfied: mdurl~=0.1 in c:\users\lakhw\anaconda3\envs\vansh\lib\site-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->typer<1.0,>=0.12->g
```

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

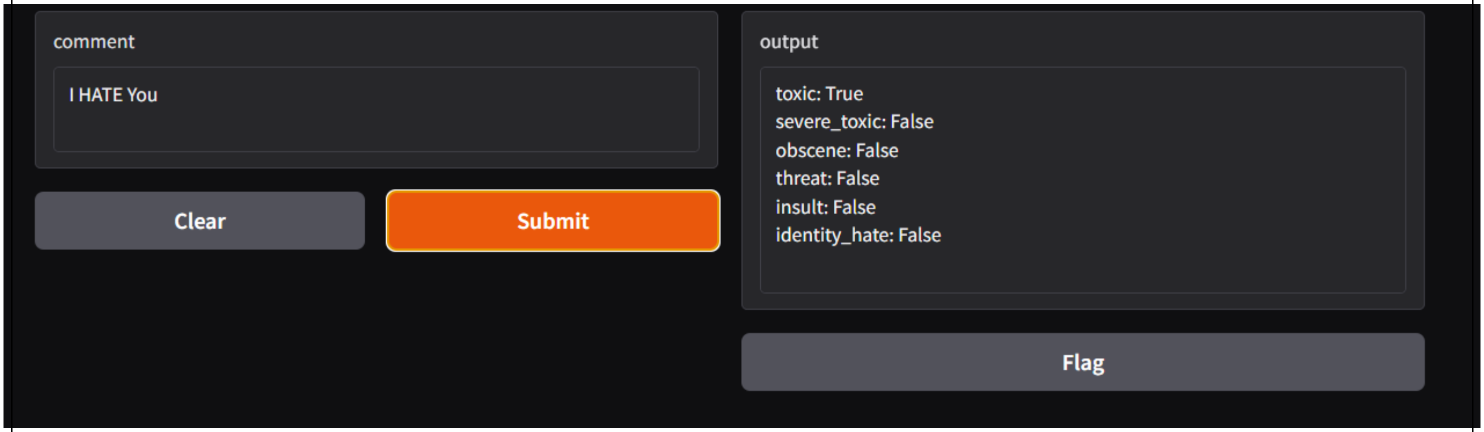


```
    interface.launch(share=True)
```

18.4s

Python

```
* Running on local URL:  http://127.0.0.1:7860

Could not create share link. Please check your internet connection or our status page: https://status.gradio.app.
```



comment

I HATE You

output

toxic: True
severe_toxic: False
obscene: False
threat: False
insult: False
identity_hate: False

CHAPTER 08
CONCLUSION

In this project, a deep learning-based multi-label classification model was successfully developed to detect various forms of toxic language in user comments. Leveraging an LSTM architecture with appropriate text vectorization for multi-label output, the model demonstrated strong performance on common toxicity categories like *toxic*, *obscene*, and *insult*. While challenges remained in accurately detecting less frequent and more nuanced labels such as *threat* and *identity hate*, the model still exhibited promising results overall.

The deployment of the model through a Gradio interface provided an accessible platform for real-time evaluation and user interaction. This practical integration highlights the system's potential for use in content moderation tools, social platforms, and community management systems. Future enhancements could include experimenting with transformer-based models (e.g., BERT), addressing class imbalance more rigorously, and incorporating contextual cues to better handle implicit toxicity.

CHAPTER 09
FUTURE SCOPE

1. Integration of Advanced NLP Models

- The current model uses LSTM-based architecture, which can be further improved by integrating state-of-the-art transformer models like BERT, RoBERTa, or DistilBERT.
- These models offer better context understanding and have shown higher accuracy in various NLP tasks.

2. Multi-Language Support

- At present, the system is limited to English text.
- In the future, multilingual support can be added using models like XLM-R or mBERT, allowing toxicity detection in multiple regional and international languages.

3. Real-time Social Media Moderation

- This system can be integrated into social media platforms, online forums, or chat applications to moderate toxic content in real-time.
- Automated flagging or blocking of harmful messages can create safer online communities.

4. Explainability and Interpretability

- Implementing explainable AI (XAI) techniques such as LIME or SHAP can help users and moderators understand why a particular comment is labeled as toxic.
- This can increase trust and transparency in AI-based moderation systems.

5. Mobile and Cross-Platform Deployment

- Future versions can be deployed as mobile apps or browser extensions for on-the-go toxicity detection.
- This will make the system more accessible to end-users and content creators.

6. Continuous Learning and Feedback Loop

- A feedback mechanism can be integrated where users can report false positives or negatives.
- This can be used to retrain and continually improve the model’s performance over time using real-world data.

CHAPTER 10
REFERENCE

[1] O'Brien, D., Benigni, T., Briskin, A., Buchanan, B., Chang, C. H., Chou, J., ... & Yarosh, L. (2019). The ecosystem of harassment: Understanding and addressing online abuse. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *26*(6), 1-34.
[2] Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017). Automated hate speech detection and the problem of offensive language. In *Proceedings of the* [1] *11th international AAAI conference on web and social* [2] *media* (pp. 512-515).
[3] Badjatiya, P., Gupta, S., Varma, V., & Gupta, M. (2017). Deep learning for hate speech detection in tweets. In *Proceedings of the 26th international conference on world wide web* [3] (pp. 759-760).
[4] Sarno, D., Squicciarini, A. C., Blackburn, J., & Weikum, G. (2022). Harassment detection in online social networks: A survey. *ACM Computing Surveys (CSUR)*, *55*(1), 1-37.
[5] Gao, J., Huang, Z., & Li, N. (2023). A survey on deep learning for hate speech detection. *Information Processing & Management*, *60*(1), 103173.
[6] Aggarwal, C. C., & Zhai, C. (Eds.). (2012). *Mining text data*. Springer Science & Business Media.
[7] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
[8] Jurafsky, D., & Martin, J. H. (2023). *Speech and language processing* (3rd ed. draft).
[9] Hovy, D., & Søgaard, A. (2016). Tagging performance matters (up to a point). *arXiv preprint arXiv:1604.08332*.
[10] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444.
[11] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. [4] N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).
[12] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, *34*(1), 1-47.
[13] Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.
[14] Alpaydin, E. (2020). *Introduction to machine learning* (4th ed.). MIT press.
[15] Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge university press.
[16] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.
[17] Chollet, F. (2017). *Deep learning with Python*. Manning Publications Co.
[18] [6] Han, J., Kamber, M., & Pei, J. (2011). *Data mining: concepts and techniques*. Morgan Kaufmann.
[19] Chawla, N. V. (2010). Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook* (pp. 875-886). Springer, Boston, MA. [7]
[20] Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent data analysis*, *6*(5), 429-449.

.