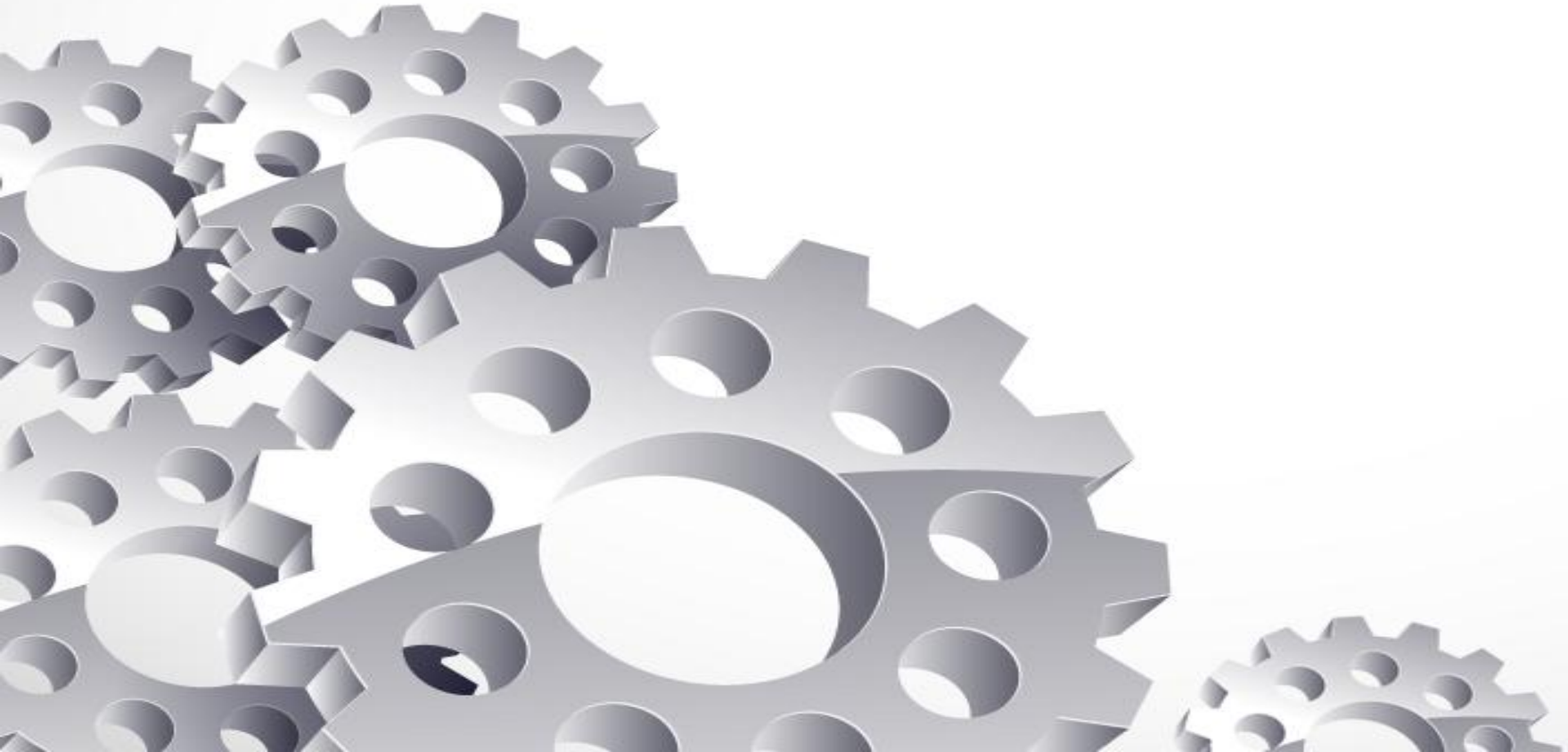
Hate Speech Detection in Movie Reviews



Group 1:
B.Veda Prabhas
Daniel Suresh
Peram Bharathi

Problem Statement:

- Online movie reviews affect the reputations of films, yet widespread hate speech on digital platforms can harm actors, directors, studios, and platforms. Hate speech not only discourages participation but also poses moral and legal questions, affecting user confidence. It's critical to identify and eliminate hate speech from reviews in order to protect consumer satisfaction, maintain legal compliance and preserve reputations.

Proposed system:

- This project aims detects hate speech in movie reviews using advanced machine learning models.
- Accurately identifying offensive content enhances user experience, protects the platform's reputation, and ensures compliance with legal standards.
- By leveraging a high-quality dataset with labeled movie reviews, the models are trained to recognize and filter out hate speech effectively.
- The project ensures precise detection of hate speech, making online movie reviews safer and more welcoming for all users.

Dataset selected

We used the Davidson dataset, which includes over 24,000 annotated tweets classified as hate speech, offensive language, or neither, for our hate speech detection project. This well-balanced and carefully labeled dataset helps us develop robust algorithms to accurately identify offensive material in movie reviews.

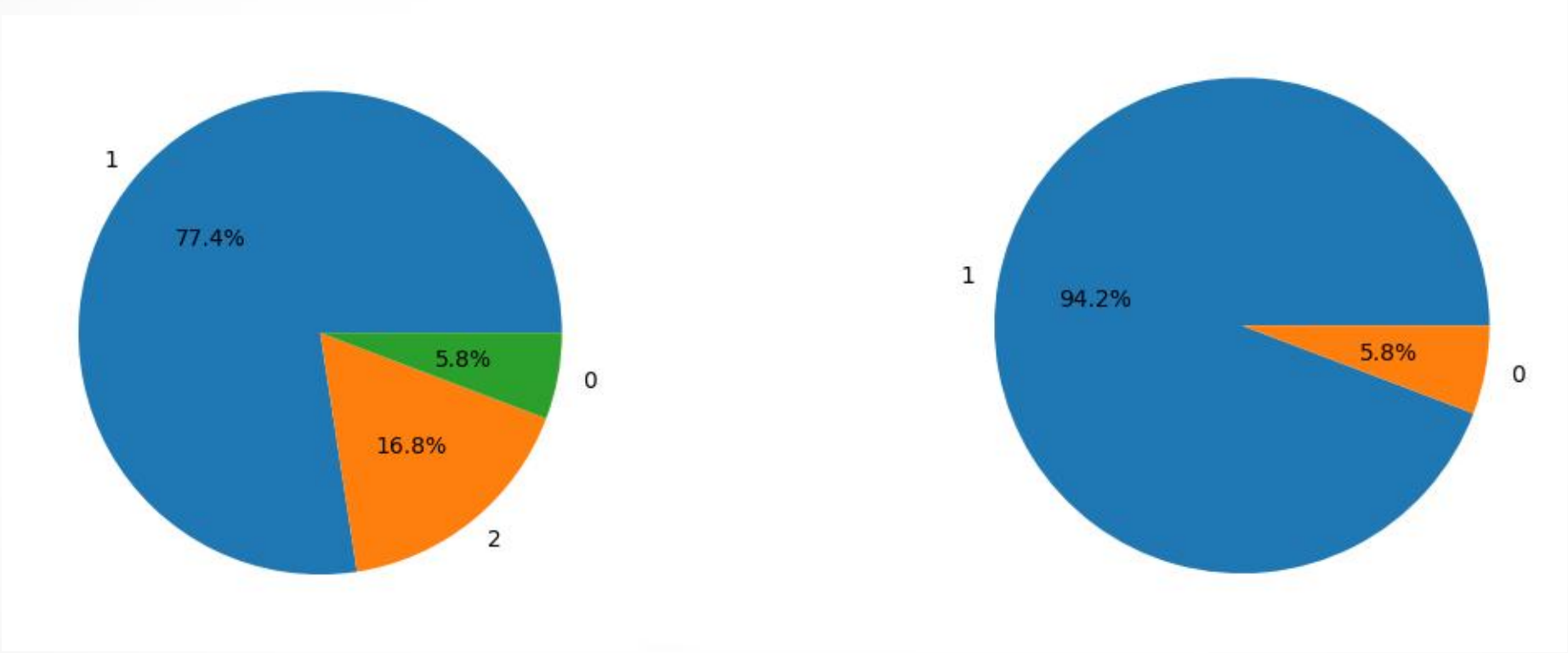
Examples:

- **Tweet:** "@RTNBA: Drakes new shoes... dudes a fag"
- **Class 0:** - Non-offensive and no hate speech
- **Tweet: "**& you might not get ya bitch back & thats that"
- **Class 1:** - Offensive
- **Tweet:** ""@OSAY_it_aint_so: “@IgnoreAllLaws: Fosters home for imaginary trash&L"#8221; WHOA CHIL”
- **Class 2:** - Hate speech

Data Visualization:-

- The Davidson dataset categorizes tweets into three classes: non-offensive, offensive, and hate speech.
- Ensures models are trained on a diverse and representative sample.
- Helps distinguish between harmless content and offensive or harmful language.
- We have mapped offensive to hate speech to further improve the performance.
- A pie chart shows the dataset's balance, guiding model training and evaluation strategies effectively.

Data visualization



Before mapping

After mapping

Data Preprocessing:

To ensure our dataset is well-prepared for training and to enhance the quality and relevance of the text data, we undertook essential data cleaning and preprocessing steps. These measures systematically eliminate noise and standardize the dataset for analysis and model training:

Data Preprocessing Steps

1. Loaded and inspected dataset for structure and size.
2. Removed usernames, numeric sequences, and URLs.
3. Lowercased all text for consistency.
4. Removed punctuation for simplification.
5. Standardized abbreviations and slang.
6. Removed stopwords and lemmatized words for clarity.

Tokenization and Embedding Techniques:-

- In our project, we employed TF-IDF (Term Frequency-Inverse Document Frequency) and tokenization techniques for text processing.
- TF-IDF highlights important terms by weighing their frequency in documents against their rarity across the corpus, reducing noise and enhancing text classification accuracy.
- Tokenization breaks text into tokens, aiding in numerical representation and embedding, which helps deep learning models understand the semantic context.
- Together, these methods enable our models to extract nuanced patterns and semantic meanings, improving prediction accuracy. We used a word tokenizer to segment text into individual words, enhancing our hate speech detection model's effectiveness.

Modeling:-

- Our approach to hate speech detection integrates a variety of machine learning and deep learning models selected for their unique strengths in analyzing textual data, spanning from traditional classifiers to advanced architectures.
- The tested models include a variety of machine learning and deep learning architectures tailored for hate speech detection.

Machine Learning Models:

- Random Forest: Robust for high-dimensional data, minimal preprocessing, effective hate speech categorization.
- Support Vector Machines (SVMs): Efficient in delineating complex class boundaries via kernel functions, adept at identifying subtle hate speech patterns.
- Naive Bayes: Simple, quick, handles sparse data well, offers foundational insights into hate speech detection.
- Logistic Regression: Interpretable, reveals feature importance, useful for probabilistic outputs in hate speech classification.
- Decision Tree: Intuitive, captures non-linear relationships, identifies intricate linguistic patterns indicative of hate speech.

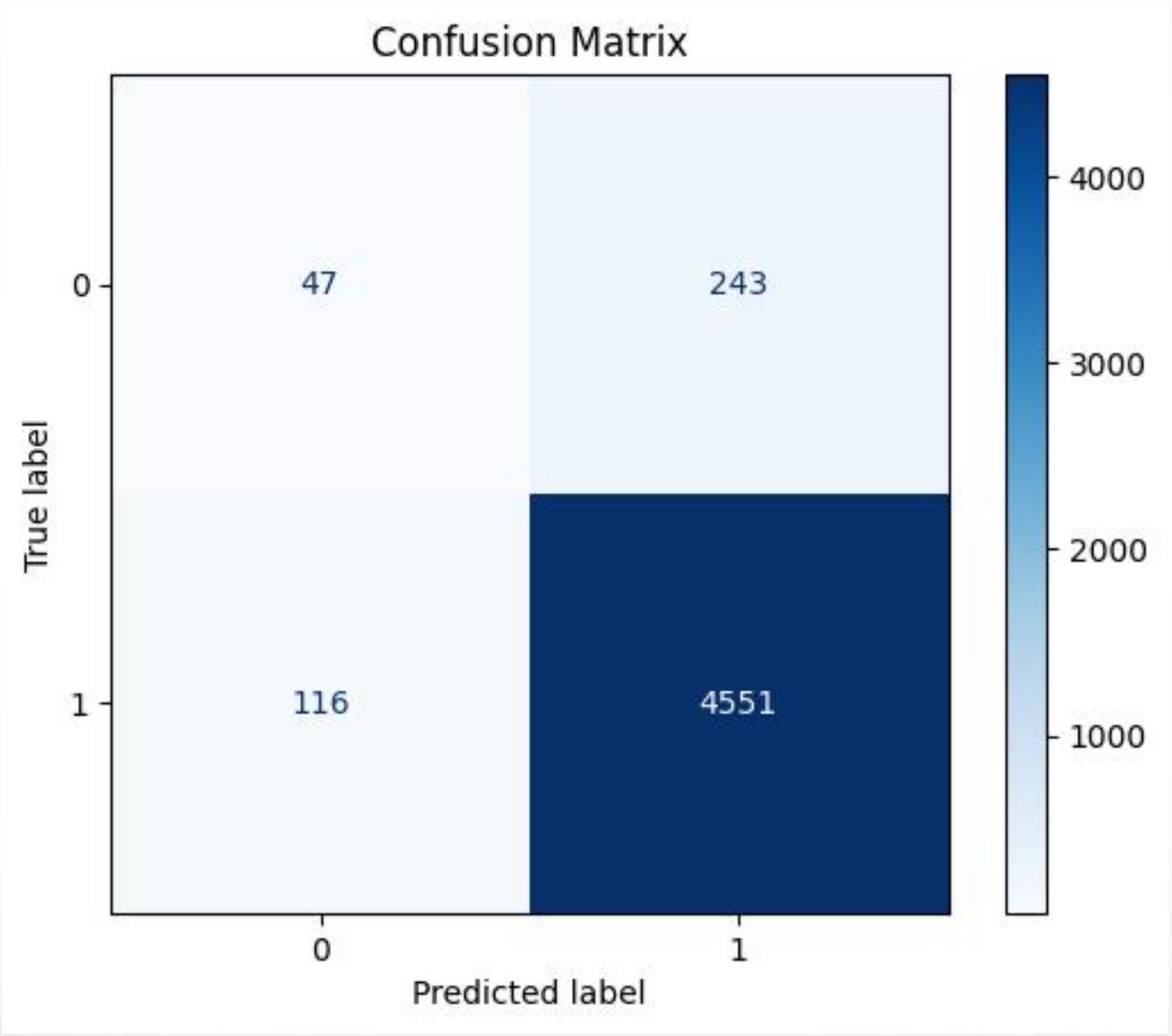
Deep Learning Models:

- LSTM (Long Short-Term Memory): Captures temporal dynamics, adept at identifying nuanced linguistic patterns in hate speech detection.
- Bidirectional LSTM: Enhances LSTM by considering context from both past and future, improves hate speech classification accuracy.
- CNNs (Convolutional Neural Networks): Extracts hierarchical features from text, identifies local and global hate speech patterns effectively.

Evaluation Metrics:

- We have used the F1 score as an evaluation metric due to its ability to balance precision and recall effectively, ensuring comprehensive evaluation of model performance.
- Precision measures the accuracy of positive predictions, while recall measures the completeness of positive predictions.
- The bidirectional LSTM model achieved an F1 score of 0.9239, highlighting its robustness in capturing contextual dependencies for accurate identification of hate speech.

Confusion Matrix



Classification report

```
Weighted F1 Score: 0.9239822701671365
Classification Report:
              precision    recall  f1-score   support

           0       0.42      0.14      0.22       290
           1       0.95      0.99      0.97      4667

    accuracy                           0.94      4957
   macro avg       0.69      0.57      0.59      4957
weighted avg       0.92      0.94      0.92      4957
```

Model: "sequential_9"

Layer (type)	Output Shape	Param #
embedding_8 (Embedding)	(None, 100, 100)	1,000,000
spatial_dropout1d_8 (SpatialDropout1D)	(None, 100, 100)	0
bidirectional_8 (Bidirectional)	(None, 128)	84,480
dense_8 (Dense)	(None, 2)	258

THANK YOU