Offensive Text Detection: Exploring Traditional Classifiers, Ensemble Models, and Kolmogorov Arnold Networks in Code-Mixed Tamil-English Text

K Jaidev, Munnangi Pranish Kumar, Jampala Sai Chandana, Charishma Chowdary and Sachin Kumar

December 16, 2024

Offensive Text Detection: Exploring Traditional Classifiers, Ensemble Models, and Kolmogorov Arnold Networks in Code-Mixed Tamil-English Text

Jaidev K, Munnangi Pranish Kumar, Jampala Sai Chandana, T Charishma Chowdary, Sachin Kumar S

Amrita School of Artificial Intelligence, Coimbatore, Amrita Vishwa Vidhyapeetham, India

Contributing authors: cb.en.u4aie21117@cb.students.amrita.edu; cb.en.u4aie21118@cb.students.amrita.edu; cb.en.u4aie21137@cb.students.amrita.edu; cb.en.u4aie21169@cb.students.amrita.edu; s_sachinkumar@cb.amrita.edu

Abstract

Offensive content has become more common in the digital era due to the growth of social media and online communication, especially in languages like Tamil. The challenges of detecting such harmful content are due to the large-scale labelled information scarcity and the intricacy of code-switching. The hybrid architecture for offensive text identification described in this paper combines the most beneficial aspects of Kolmogorov-Arnold Networks (KAN), traditional machine learning classifiers, and ensemble models. Our strategy involves preprocessing of text, several extracted features, and tuning of hyperparameters for better performance of the model. We explore many different classifier performances comprising XGBoost, AdaBoost, Gradient Boosting, K-Nearest Neighbours (KNN), Random Forest, Support Vector Machine (SVM), and Logistic Regression. Extensive trials show that our hybrid system, particularly leveraging KAN, emerges as the best model for precisely identifying objectionable material in Tamil-English datasets with mixed coding. To address the challenges of offensive content identification in multilingual and code-mixed contexts, the results demonstrate the potential benefits of integrating conventional and cutting-edge machine learning techniques.

Keywords: Offensive text detection, neural networks, Key Attention Networks, traditional classifiers, code-mixed text, Tamil-English

1 Introduction

The rise of social media and online communication in the digital age has led to an increase in the prevalence of offensive content. This tendency is especially noticeable in code-mixed languages like Tamil and English. Difficulty in code-switching and lack of large-scale labeled data make the discovery of such harmful information even more difficult to discover. In this work, we present a hybrid architecture that leverages the best features of standard machine learning classifiers and Kolmogorov Arnold Networks with ensemble models for the recognition of offensive text. It includes word preprocessing, feature extraction through Word2Vec representation, bag-of-words, Ngrams, and TF-IDF embeddings, and tuning hyperparameters for better performance. The experimental evaluation of different classifiers such as XGBoost, AdaBoost, Gradient Boosting, K-Nearest Neighbours (KNN), Random Forest, Support Vector Machine (SVM), and Logistic Regression proves the superiority of our proposed hybrid system in correctly identifying objectionable material in code-mixed Tamil-English datasets.

Such use of language is very common in multilingual societies, where speakers slip in and out of languages to convey shades of meaning, cultural allusions, and distinctive texture with ease. In Tamil-English communication, code-mixing is more than just a linguistic phenomenon; it’s a cultural and social artifact that illustrates the dynamic interplay of language, identity, and community. However, because of the peculiarity of code-mixed communication, there are specific issues regarding content filtering and incorrect identification of language. Due to syntactic, morphological, and semantic complexities, traditional language processing techniques have comparatively higher difficulty in processing code-mixed information than monolingual text. As mentioned above, there is a severe scarcity of annotated datasets and resources, which makes the process of creating error-free and efficient algorithms for the detection of offensive content in code-mixed language much more complicated.

This project intends to develop an ensemble of KAN, the Kolmogorov Arnold Network, along with conventional machine learning classification models. Codemixed text is too complex and detailed to be captured by a single methodology, which is why we require a combined method here. The purpose of this work is to overcome the drawbacks of existing approaches and improve the efficiency, accuracy, and applicability of the method of identifying obscene materials in text messages by integrating multiple approaches within the proposed framework. The text pre-processing phase of the system is to enhance the quality and normalize the nature of code-mixed text data. This consists of operations ranging from tokenization, stemming, lemmatization, and other morphological reductions sometimes used in pre-processing known as noise reduction. Further, techniques of feature engineering such as Word2Vec (Word to Vector) and Term Frequency-Inverse Document Frequency (TF-IDF) are also used to extract the contextual and semantic essence that comes along with code-mixed languages. Moreover, cross-validation is used, and high standards for hyperparameters are required to provide better results and stability in different language environments. The key contributions of this work are as follows:

1. To detect abusive content in Tamil-English code-mixed data, a hybrid framework including ensemble models, conventional classifiers, and Kolmogorov-Arnold Networks (KAN) was developed
2. To address the linguistic complexity of code-mixed text, sophisticated preprocessing and feature extraction methods such as TF-IDF and Word2Vec are used.

3. Extensive tests verifying the hybrid methodology, with findings emphasizing KAN's better interpretability and accuracy over current techniques.

This is how the rest of the paper is organized: With an emphasis on the challenges in multilingual and code-mixed environments, Section 2 reviews relevant works in offensive text identification. Section 3 explains the recommended approach, which includes the hybrid framework and feature engineering tools. In Section 4, experimental findings are presented and the comparative performance of classifiers is examined. Section 5 concludes with a discussion of the main conclusions, limitations, and potential directions for further research.

2 Literature Review

Chakravarthi et al. [1] presented an overview of the HASOC-DravidianCodeMix shared task, focusing on offensive language detection in code-mixed Tamil and Malayalam text. The task involved 16 teams in the Tamil-English track, 11 in Malayalam-English, and 14 in the Tamil-only track. These teams used a wide variety of diverse machine learning and deep learning models, out of which the best performing reported an F1-score of 0.859 for Tamil and 0.766 for Malayalam. The authors attributed performance enhancement to TF-IDF features combined with robust classifiers.

Kumar et al. in [2] have proposed a new deep learning framework based on an ensemble of CNNs and Dense Neural Networks that entails state-of-the-art performance in identifying offensive posts in the text data of Dravidian MIoT media. Biere et al. in [3] addressed hate speech detection in social media, where Logistic Regression and Random Forest classifiers were deployed to contrast the performance. Instead, they found that the TF-IDF scores outperformed the performance prompted by Sentiment Polarity scores in recognition of hate speech for tweets, hence showing the role that TF-IDF plays in hate speech detection through platforms.

Boishakhi et al. [4] researched the auto-classification of offensive language using multilingual BERT on tweet datasets, which achieved 92% accuracy, demonstrating the power of transfer learning models in finding hate speech. Similarly, Khan et al. [5] proposed a two-level classification for the detection of hate speech in Roman Urdu. In addition, Logistic Regression amongst all the supervised models demonstrated better performance than other algorithms in distinguishing hate speech over social media.

Further work by Chakravarthi et al. [6] addressed offensive language detection in Tamil, Malayalam, and Kannada and code-switching and non-native script issues. This research placed further emphasis on the need for dataset development and shared tasks which would push the research forward into offensive content detection for under-resourced languages. Their study is all the more relevant for languages in extensive use on platforms such as YouTube and Helo.

Baruah et al. [7] worked on the offensive language detection of code-mixed Dravidian languages and adapted various machine learning and deep learning methods to result in better performance. They also contributed to sentiment analysis and machine translation efforts in the Tamil and Malayalam languages, furthering the resources available for those languages.

Ravikiran et al. [8] leveraged Transformer architectures such as XLM-RoBERTa and improved offensive speech detection in multilingual datasets through the selective translation and transliteration of mixed-script texts. This work added not just new methodologies but also the validation of the effectiveness of the Transformer models across low-resource languages, hence underlining their potential to combat hate speech on social media.

3 Methodology

3.1 Data Preprocessing

Several steps were taken in the preparation phase to clean and standardize the text data. All punctuation, digits, special characters, and extra spaces were removed, and the text was all lowercase. Additional steps, such as tokenization and stop word removal, were implemented to ensure consistency and relevance because the content was code-mixed.

3.2 Feature Extraction

Multiple features were extracted to capture essential aspects of the text. The relevance of each word in the corpus was calculated using TF-IDF [9], it is a relative weight that takes into account the number of times a word appears in a document and the number of documents in which the word appears. The TF-IDF score for a term *t* in a document *d* within a corpus *D* is calculated as follows:

$$\text{TF-IDF}(t,d,D) = \text{TF}(t,d) \times \text{IDF}(t,D)$$

$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

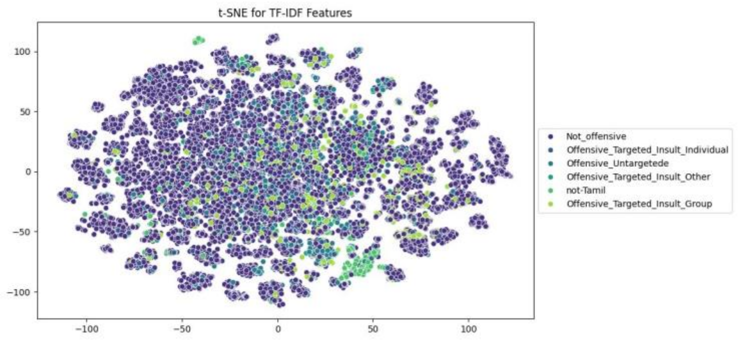

Figure 1: t-SNE of TF-IDF

is the term frequency of term *t* in document *d*, and

$$IDF(t,D) = \log\left(\frac{N}{|\{d \in D: t \in d\}|}\right)$$

is the inverse document frequency, with *N* representing the total number of documents in the corpus *D* and $|\{d \in D: t \in d\}|$ denoting the number of documents in which the term *t* appears.

The t-SNE(t-distributed Stochastic Neighbour Embedding) plots are used to visualize the high-dimensional feature space in a lower-dimensional space, making it easier to identify patterns and clusters in the data. The t-SNE plot for TF-IDF features is given in Figure 1.

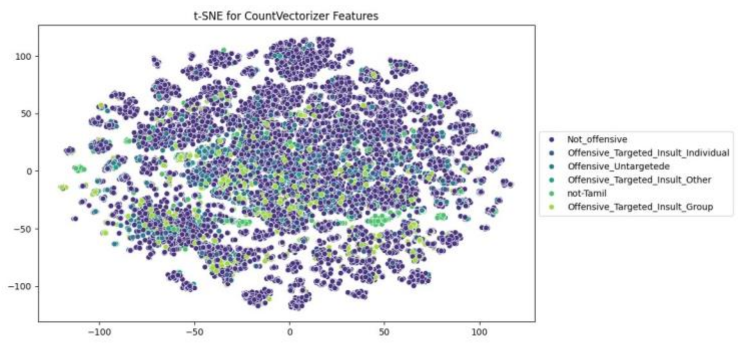

Figure 2: t-SNE of BoW

N-grams [10] are contiguous sequences of 'n' items from text, used to extract patterns and contextual information. By generating overlapping word groups, documents are analysed to uncover phrase-level patterns and dependencies. The text data is modelled using the Bag of Words (BoW) [11] approach, where the

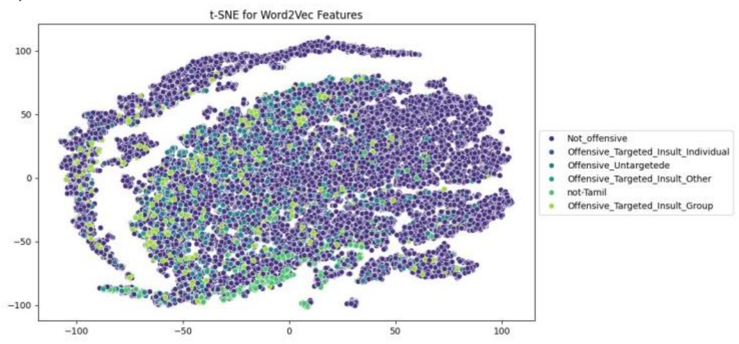

Figure 3: t-SNE of Word2Vec

data is represented in the form of sparse matrices indicating the frequency of words. In this model, every document is represented by a vector in which every component of the vector corresponds to a unique word and quantifies the occurrence of that word in the document. The t-SNE plot for BoW features is given in Figure 2.

Word2Vec [12] generates low-dimensional dense vectors that represent the syntactic connections between words.

In this model, the words are mapped to a high-dimensional space while retaining contextual relationships and meaning. The t-SNE plot for Word2Vec features is given in Figure 3.

3.3 Traditional and Ensemble methods

We have tried many standard machine-learning models for this classification. Due to its accuracy in high-dimensionality domains, the Linear Support Vector Classifier (Linear SVC) [13] was used for this task. To improve the classification performance, multiple decision trees were used along with the Random Forest Classifier [14]. It was also examined with a logistic function for a binary classification in logistic regression. The K-Nearest Neighbours (KNN) [15] algorithm was included in our work due to its simplicity and effectiveness in non-parametric classification. We also looked at the AdaBoost Classifier [16] which combines several weak classifiers by re-weighting the instances and the Gradient Boosting [17] Classifier which develops models one at a time in an attempt to rectify the errors of previous models.

3.4 Kolmogorov Arnold Networks (KAN)

Kolmogorov-Arnold Networks (KANs) [18] are a novel neural network architecture developed using the Kolmogorov-Arnold representation theorem. The complete flow diagram of the methodology is provided in Figure 4. Unlike traditional neural networks like Multi-Layer Perceptron (MLPs), KANs replace linear weights with univariate functions parameterized as splines, making activation functions learnable on edges rather than fixed on nodes. We have chosen KAN because it excels at function approximation, making it suitable for

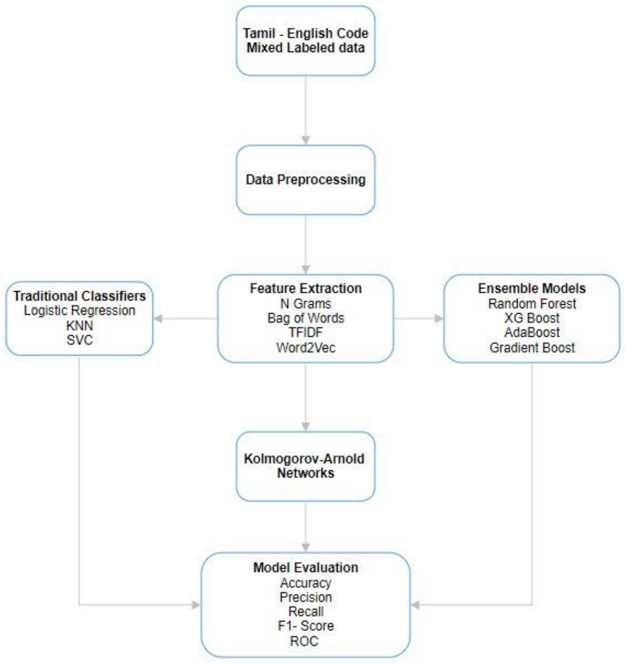
Tamil - English Code Mixed Labeled data
Data Preprocessing
Traditional Classifiers Logistic Regression KNN SVC
Feature Extraction N Grams Bag of Words TFIDF Word2Vec
Ensemble Models Random Forest XG Boost AdaBoost Gradient Boost
Kolmogorov-Arnold Networks
Model Evaluation Accuracy Precision Recall F1- Score ROC

Figure 4: Work Flow

complex NLP tasks. In this method, pre-processed text features serve as the input to KAN. Each layer of the KAN applies univariate functions to these input features, parametrized as splines. These functions replace traditional linear weights, capturing complex patterns in the data. The network combines these functions using superposition and addition to approximate the decision boundary between offensive and non-offensive text. The last layer of output classifies the text as either offensive or non-offensive. The architecture of the KAN is shown in Figure 5. It improves accuracy and readability, and as a result, KANs surpass MLPs in data fitting and solving partial differential equations. This is advantageous as doing so allows more flexibility in modelling the complexity of the data, especially in the case of code-switching as seen in Tamil-English text data.

The proposed architecture of the KAN, therefore, greatly improves performance concerning distinguishing between the two types of texts, which is crucial for solving intricate NLP problems.

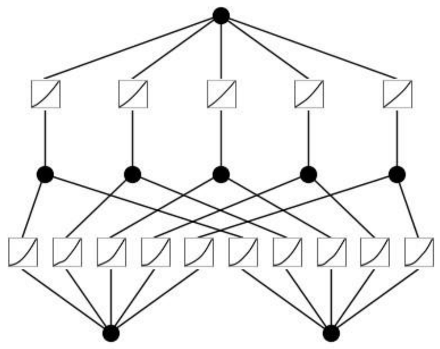

Figure 5: Architecture of KAN [18]

4 Experimentation and Results

4.1 Dataset Description

To identify objectionable language in Dravidian languages and do sentiment analysis, this study makes use of the HASOC-Dravidian Code Mixed dataset, which is publicly accessible and manually annotated [19]. With an emphasis on code-mixed communication—which is common on social networking platforms—the dataset covers Tamil, Malayalam, and Kannada. The Tamil and Tamil-English code-mixed sub-corpus of the HASOCDravidian Code Mixed dataset was the main focus of this investigation. About 44,000 comments from YouTube are included in this sample. Natural code-mixing phenomena are evident in these remarks, which capture the distinctive language patterns found in user-generated content from multilingual areas. Krippendorff's alpha, a measure of inter-annotator agreement, was strong after a group of volunteer annotators manually annotated the dataset. Every comment is categorized by the annotations into a set of predetermined classifications, including non-offensive and several offensive types.

A useful resource for developing and testing machine learning models for detecting foul language in Tamil-English code-mixed communication is the Tamil-English subset of the HASOC-Dravidian Code Mixed dataset. By concentrating on this subset, our work contributes to the growth of natural language processing research in multilingual environments by bridging the resource gap for under-resourced languages. The study "DravidianCodeMix: Sentiment Analysis and Offensive Language Identification Dataset for Dravidian Languages in Code-Mixed Text" by Chakravarthi et al. (2022) presents the HASOC-Dravidian code mixed dataset. [19]

4.2 Task Description

In this work, we consider the crucial task of detecting undesirable information in bilingual mixed-coding Tamil-English conversations. The dataset for this task is a collection of text items, further marked by specific categories indicating the kind of information. In the following experiments, employing a mixed system combining KAN, conventional machine learning classifiers, and ensemble models, the primary aim is to classify these texts into the respective categories by capturing the textual content of the items. The dataset is organized in a tabular style, where a single text input and its label are represented by each row. A thorough explanation of the information and the categories is given below:

1. **Not Offensive:**
   Text: *Screenplay vera level laaa iruku.... Waiting to watch it.*
   Translation: The movie's screenplay is very good, waiting to watch it. This entry is non-offensive and contains a congratulatory message in Tamil-English, expressing good wishes for the success of a movie for its screenplay.
2. **Offensive Untargeted:**
   Text: *Commercial kuppai bigil Ku thriller story Kaithi eh paakalam* Translation: Instead of watching trash commercial movies, thriller movies like Kaithi can be watched.
   This entry contains offensive language, but it is not targeted towards any specific individual or group. It is a general offensive statement.
3. **Not Offensive:**
   Text: *Ithan padatha matum ranjith pathan vairu erum avanku aiyoo unmiya la solitangaleeee*
   Translation: If Ranjith had watched this movie, he would have died out of jealousy.
   This entry contains offensive language directed towards a specific individual named Ranjith, indicating a personal insult.
4. **Offensive Targeted Insult Other:**
   Text: *Indha Vijay fans ku rajini mela paasam laam.... Ajith and fans mela gaandu....*
   Translation: All the Vijay fans don't like Rajini genuinely, they just hate Ajith and his fans. This entry contains offensive language targeting a specific group, namely the fans of celebrities Vijay and Ajith. It reflects animosity and derogatory remarks towards the fans of these actors.
5. **Offensive Targeted Insult Group:**
   Text: *Jathi veri pudicha naaigala suttu thalanum, intha comment keela terinjidum!*
   Translation: Caste-obsessed bi*ches should be shot down, comment down This entry contains offensive language targeted towards a specific group, with references to caste and community-based derogatory remarks.
6. **Non-Tamil:**
   Text: *Abe chutiyon sab la ke ussi date pe kyon laa rahe Ho be bhadawon*
   This entry contains offensive language but is not written in Tamil. It is an example of offensive content in a different language, indicating the importance of distinguishing between languages in code-mixed texts.

The goal would be to develop a classification model that will categorize each incoming text into one of these pre-defined classes with a good level of accuracy. This would mean dealing with the subtlety of code-switching, as well as the contextual undertones of the text. Emphasis will be given to formulating an effective and credible procedure for objectionable information detection in code-mixed Tamil-English communication and serving online platform safety and moderation.

4.3 Experimental Setup

An experimental setting was developed to compare how well different models performed under various circumstances. We conducted two main experiments for the KAN model: training on the entire dataset and training on smaller, randomly chosen samples. In the latter, the procedure was carried out 100 times after 2000 data points were chosen at random. This method was selected to test the stability of the built model and its efficacy with a restricted amount of material, but it closely mimics real-world conditions where full dataset training may not be feasible. In particular, hyperparameter adjustment improved the performance of traditional machine learning models [20]. Grid search and cross-validation techniques were used to identify the hyperparameters that were appropriate for each model. To avoid confusion resulting from different categorizations of the models, a structure was devised to allow for an accurate comparison between several classifiers. This procedure ensured that each model was differentiated and assessed to the best of its capability.

4.4 Results

An analysis of many classifiers to detect offensive content in Tamil-English code-mixed text provides insightful information about the classifiers' performance on a variety of criteria. With an accuracy of 78.20%, Linear SVC distinguishes itself from other traditional machine learning models and shows its resilience in differentiating between offensive and non-offensive language. A balanced precision-recall trade-off is necessary to reduce false positives and false negatives in situations where misclassification could have major consequences.

Ensemble methods such as Random Forest and XGBoost also show promising results. XGBoost demonstrates its ability to effectively optimize model parameters and capture intricate feature interactions with a high F1 score and an accuracy of 76.55%. However, Random Forest has a lower F1 score with a good accuracy of 74.89%, suggesting that hyperparameter tweaking is required to balance the hazards of overfitting with the likelihood of overfitting. With a 75.72% accuracy rate, logistic regression is a solid starting point. It does a decent job of balancing precision and recall, but more advanced modelling techniques or improved feature architecture could help it perform even better [21]. The performance metrics for each classifier are given in Table 1.

While the overall accuracy may be quite decent, their lower precisions and F1 scores do show in comparisons that the tendency for false positives in models such as KNN and AdaBoost is much higher. This underlines the fact that apart from overall accuracy, precision, recall, and F1 scores need to be checked in cases where misclassifications need to be minimized.

Table 1: Evaluation Metrics for Different Classifiers

Classifier	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.7572	0.6954	0.7572	0.7028
RandomForest	0.7489	0.7219	0.7489	0.6525
LinearSVC	0.7820	0.7509	0.7820	0.7502
KNeighbors	0.7324	0.6476	0.7324	0.6726
AdaBoost	0.7472	0.6542	0.7472	0.6629
Gradient Boost	0.7558	0.6946	0.7558	0.6880
XGBoost	0.7655	0.7137	0.7655	0.7057

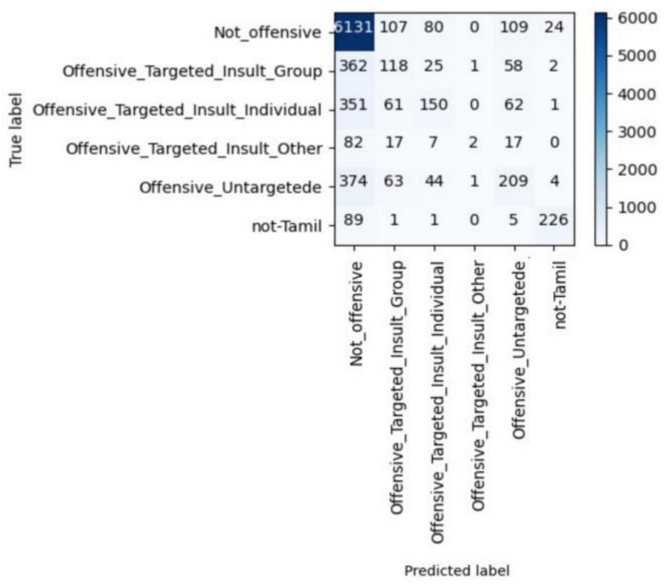

Figure 6: Confusion matrix of Linear SVC

Figure 6 presents the confusion matrix for Linear SVC. The confusion matrix, in turn, provides detailed information on the classifier's performance in terms of true positives, true negatives, false positives, and false negatives. An illustration of the true positive rate versus the false positive rate for every classifier is shown by the Receiver Operating Characteristic curve (ROC). The performance at different thresholds is displayed by each model's ROC curve, which is displayed in Figure 7. Higher values of AUC suggest better performance: the higher the area under the curve, the better the model's class separation. AUC values of 0.84 and 0.82 indicate strong performance. As for the outliers, the maximum AUC ROC statistics of 0.82 for both Logistic Regression and Linear

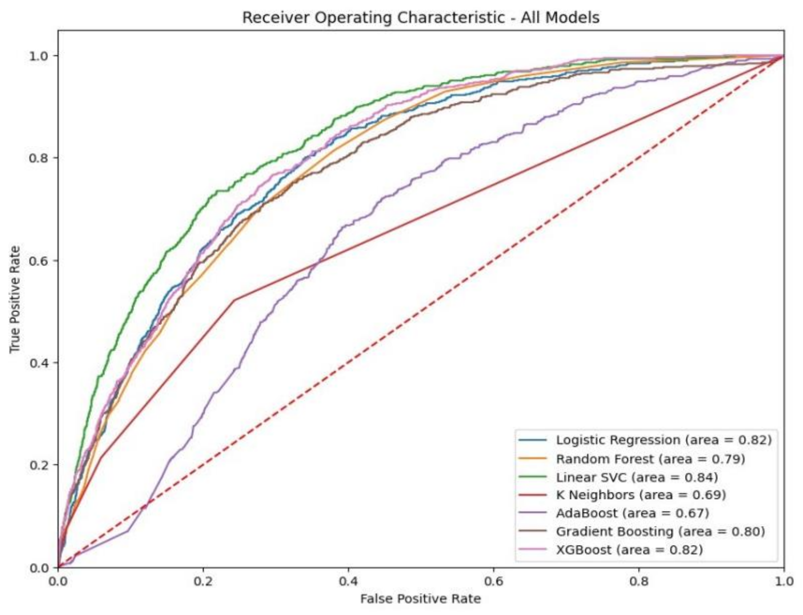

Figure 7: ROC graph of all classifiers

SVC demonstrates the highest discriminating ability. It is also established that, due to achieving similar levels of AUC, ensemble techniques like XGBoost and Gradient Boosting are capable of solving difficult classification problems. Indeed, the KAN model provides a plausible approach. This paper achieved a training accuracy of 72 percent in classifying data. It performs reasonably well when trained on the complete dataset, with a training accuracy of 42 percent and a test accuracy of 72.22 percent. However, when trained on smaller, randomly selected samples, it performs notably better. For 2000 randomly generated data points, the KAN model, using 100 iterations, showed an approximate training accuracy of 82.48 percent and an average testing accuracy of 81.38 percent. The performance metrics for the KAN model under different training methods are given in Table 2.

Table 2: KAN Model Performance by Different Training Methods

Training Method	Accuracy	Precision	Recall	F1 Score
By Training the whole data	0.7242	0.7198	0.7223	0.6860
By randomly choosing 2000 data (100 iterations)	0.8138	0.7942	0.8025	0.8083

This clearly shows that by applying the given model, there is a possibility of efficiently using the available resources, making it very suitable for situations with many constraints. Thus, it has been concluded that, with almost no difference in accuracy rates, the KAN model is a very effective replacement for Linear SVC, which is itself highly efficient among traditional classifiers. This is especially true if these elements are supported by the right training and development initiatives.

This demonstrates how the KAN model can be used and presents a realistic approach for identifying inappropriate content in Tamil-English text containing mixed code.

5 Conclusion

This work has presented the results of applying KAN and other traditional ML classifiers for obscene language identification in code-mixed Tamil-English data. Along with advanced preprocessing and feature extraction methods, these approaches have become a good basis for addressing the specifics of text with mixed codes. Several lines of further development could be pointed out. These include deeper CNNs and RNNs that can learn extended context dependencies in text. Online detection solutions, optimization for faster inference times, and integration of the proposed approach with widely used platforms become significant next steps. More research is also recommended with other code-mixed languages and domain-specific data, such as news articles and social media posts. Their use is recommended to be conditioned upon their interpretability and solving problematic aspects related to ethical norms, such as bias. The study ends with a clear and strong foundation for objectionable text identification in code-mixed Tamil-English data, leaving many avenues for deeper research and experimentation along these lines—some of which may find applications in many areas.

References

1. Bharathi Raja Chakravarthi, Prasanna Kumar Kumaresan, Ratnasingam Sakuntharaj, Anand Kumar Madasamy, Sajeetha Thavareesan, Bhavukam Premjith, K R Sreelakshmi, Subalalitha Chinnaudayar Navaneethakrishnan, John, Patrick McCrae, and Thomas Mandl. Overview of the hasoc-dravidiancodemix shared task on offensive language detection in Tamil and Malayalam. In *Fire*, 2021.
2. Abhinav Kumar, Sunil Saumya, and Ashish Singh. Detecting Dravidian offensive posts in miot: A hybrid deep learning framework. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 2023.
3. Shanita Biere, Sandjai Bhulai, and Master Business Analytics. Hate speech detection using natural language processing techniques. *The master business analytics department of mathematics faculty of science*, 2018.
4. Fariha Tahosin Boishakhi, Ponkoj Chandra Shill, and Md Golam Rabiul Alam. Multi-modal hate speech detection using machine learning. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 4496–4499. IEEE, 2021.
5. Moin Khan, Khurram Shahzad, and Kamran Malik. Hate speech detection in Roman Urdu. *arXiv preprint arXiv:2108.02830*, 2021.
6. Bharathi Raja Chakravarthi, Ruba Priyadharshini, Vigneshwaran Muralidaran, Navya Jose, Shardul Suryawanshi, Elizabeth Sherly, and John P McCrae. Dravidiancodemix: Sentiment analysis and offensive language identification dataset for Dravidian languages in code-mixed text. *arXiv preprint arXiv:2106.09460*, 2021.

7. Arup Baruah, Kaushik Amar Das, Ferdous Ahmed Barbhuiya, and Kuntal Dey. Iiitg-adbu@ hasoc-dravidian-codemix-fire2020: Offensive content detection in codemixed dravidian text. *arXiv preprint arXiv:2107.14336*, 2021.
8. Manikandan Ravikiran, Ratnavel Rajalakshmi, Bharathi Raja Chakravarthi, Sajeetha Thavareesan, et al. Findings of the first shared task on offensive span identification from code mixed kannada-english comments. In *Proceedings of the Fourth Workshop on Speech, Vision, and Language Technologies for Dravidian Languages*, pages 43–48, 2024.
9. Ishan Bansal and Mehak Sood. Improvement of accuracy for hate speech detection using modified feature extraction. *International Journal for Multidisciplinary Research (IJFMR)*, 5(5):1–9, 2023.
10. Deepa Gupta, K. Vani, and Charan Kamal Singh. Using natural language processing techniques and fuzzy-semantic similarity for automatic external plagiarism detection. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2694–2699, 2014.
11. Wisam A. Qader, Musa M. Ameen, and Bilal I. Ahmed. An overview of bag of words; importance, implementation, applications, and challenges. In *2019 International Engineering Conference (IEC)*, pages 200–204, 2019.
12. Eric L Goodman, Chase Zimmerman, and Corey Hudson. Packet2vec: Utilizing word2vec for feature extraction in packet data. *arXiv e-prints*, pages arXiv–2004, 2020.
13. G Fung and OL Mangasarian. Multicategory proximal support vector machine classifiers (technical report 01-06). *Data Mining Institute*, 2001.
14. Gilles Louppe. Understanding random forests: From theory to practice. *arXiv e-prints*, pages arXiv–1407, 2014.
15. Padraig Cunningham and Sarah Delany. K-nearest neighbor classifiers. *Mult Classif Syst*, 54, 04 2007.
16. Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society for Artificial Intelligence*, 14(771-780):1612, 1999.
17. Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.
18. Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljaˇci´c, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.
19. Bharathi Raja Chakravarthi, Ruba Priyadharshini, Vigneshwaran Muralidaran, Navya Jose, Shardul Suryawanshi, Elizabeth Sherly, and John P McCrae. Dravidiancodemix: Sentiment analysis and offensive language identification dataset for Dravidian languages in code-mixed text. *Language Resources and Evaluation*, 56(3):765–806, 2022.
20. M Geerthana Anusha, S Sachin Kumar, and KP Soman. Deep learning-based topic categorization of Tamil social media text content. In *Advanced Machine Intelligence and Signal Processing*, pages 829–844. Springer, 2022.
21. M Visweswaran, Jayanth Mohan, S Sachin Kumar, and KP Soman. Synergistic detection of multimodal fake news leveraging text can and vision transformer. *Procedia Computer Science*, 235:142–151, 2024.