Tackling Cyber Hatred with Machine Learning and Fuzzy Logic

[1]katroth Balakrishna Maruthiram, [2]gujje Vijayakrishna
[1] Assistant Professor in CSE, [2] MCA Student,
Department of Information Technology, Jawaharlal Nehru Technological University, India.

Abstract: The project centers on addressing the concerning issue of cyber-hate, which has significantly escalated with the widespread adoption of social media platforms. It acknowledges the urgency and importance of dealing with this problem within the digital landscape. To combat cyber-hate, the project proposes the use of various machine learning and deep learning techniques. These include Naive Bayes, Logistic Regression, Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs). Each of these methods likely serves a specific purpose in identifying, classifying, or analyzing patterns within hate speech or offensive content. The project implements two classifiers on hate speech data and enhances their performance using optimization methods such as Particle Swarm Optimization and Genetic Algorithms. These optimization techniques are likely employed to fine-tune the classifiers and improve their accuracy in detecting cyber-hate instances. Additionally, the inclusion of Fuzzy Logic aims to enhance the comprehension of text data, accounting for its inherent complexity and nuances. The primary goal is to develop a more effective and realistic approach to cyber-hate detection. This involves incorporating a critical thinking perspective, which likely means considering contextual cues and subtle nuances beyond explicit keywords or phrases. Furthermore, the utilization of optimization techniques and fuzzy logic-based systems is aimed at creating a more nuanced understanding of hate speech, making the detection process more accurate and aligned with real-world complexities. The project extends its capabilities through the integration of advanced ensemble techniques, specifically a Voting Classifier and a Stacking Classifier. The Stacking Classifier achieves an impressive 100% accuracy, demonstrating its robustness in identifying cyber hate instances. Leveraging these ensemble models enhances the overall effectiveness of the cyber-hate detection system.

Index terms - Cyberbullying, fuzzy logic, logistic regression, multinomial Naive Bayes, PSO, VADER.

1. INTRODUCTION

It was the advancement of technology and the impulse of human communication that led to the evolution of social media, which altered how individuals interact online. Prior to the introduction of Information Communication Technology (ICT), human interactions were largely confined to geographical locations; however, Online Social Networks (OSNs) have eliminated geographical barriers [1].

It has become increasingly apparent that cyber-hate is a widespread issue due to the pervasiveness of easy-to-use technologies. Social media platforms have emerged as a medium for the perpetration of aggressiveness and bullying, making it a dangerous and elusive phenomenon. The ease with which perpetrators can commit harmful acts through the utilization of a laptop or mobile device connected to the internet renders young individuals highly vulnerable to online harassment. A conventional approach to detecting cybercrime involves manual flagging of data [2]; however, this method has been demonstrated to be neither ‘‘effective nor scalable’’ [2]. This has prompted researchers to investigate the potential of utilizing Machine Learning and Deep Learning techniques to design automated systems capable of detecting and preventing cyber-hate.

Considering the vast amount of content that can be found on OSNs related to aggressive and anti-social behaviour, the paper proposes an Optimized Machine Learning-Based framework to help identify online hate using fuzzy logic techniques [35,36]. Several different machine learning models have been implemented, such as, Multinomial Naive Bayes and Logistic Regression, in conjunction with the Bio-Inspired Optimization methods, Genetic Algorithm and Particle Swarm Optimization [30,31,48]. The implementation of Particle swarm Optimization selects the best feature selection subset that better represents the feature selection space. The aim is to decrease the quantity of redundant and unimportant features, thereby enhancing the accuracy of the classification process within a data set. Additionally, PSO improves the comprehensibility of the learned model. Furthermore, Genetic Algorithm (GA) was implemented to optimize the performance of classifiers. The random mutation aspect of the GA provides a degree of assurance that a wide range of solutions are evaluated. Furthermore, the application of fuzzy rules is able to incorporate the fuzziness of positive and negative scores. The Fuzzy Logic-based systems were applied to deal with vagueness and ambiguity. The advantages of using the fuzzy approach are summarized as i) It provides a desirable way to deal with linguistic problems and ii) Deals with reasoning and gives closer views to the exact sentiment values.

2. LITERATURE SURVEY

The exponential increase of social media users, cyberbullying has been emerged as a form of bullying through electronic messages [1]. Social networks provides a rich environment for bullies to uses these networks as vulnerable to attacks against victims. Given the consequences of cyberbullying on victims, it is necessary to find suitable actions to detect and prevent it. Machine learning can be helpful to detect language patterns of the bullies and hence can generate a model to automatically detect cyberbullying actions. This paper proposes a supervised machine learning approach for detecting and preventing cyberbullying. Several classifiers are used to train and recognize bullying actions. The evaluation of the proposed approach on cyberbullying dataset shows that Neural Network performs better and achieves accuracy of 92.8% and SVM achieves 90.3. Also, NN outperforms other classifiers of similar work on the same dataset ([23], [24], [25], [26], [27]).

The scourge of cyberbullying has assumed alarming proportions with an ever-increasing number of adolescents admitting to having dealt with it either as a victim or as a bystander. Anonymity and the lack of meaningful supervision in the electronic medium are two factors that have exacerbated this social menace. Comments or posts involving sensitive topics that are personal to an individual are more likely to be internalized by a victim, often resulting in tragic outcomes[5]. We decompose the overall detection problem into detection of sensitive topics, lending itself into text classification sub-problems. We experiment with a corpus of 4500 YouTube comments, applying a range of binary and multiclass classifiers. We find that binary classifiers for individual labels outperform multiclass classifiers. Our findings show that the detection of textual cyberbullying can be tackled by building individual topic-sensitive classifiers.

In this paper we describe a close analysis of the language used in cyberbullying [22,33]. We take as our corpus a collection of posts from Formspring.me. Formspring.me is a social networking site where users can ask questions of other users. It appeals primarily to teens and young adults and the cyberbullying content on the site is dense; between 7% and 14% of the posts we have analyzed contain cyberbullying content. The results presented in this article are two-fold. [6] Our first experiments were designed to develop an understanding of both the specific words that are used by cyberbullies, and the context surrounding these words. We have identified the most commonly used cyberbullying terms, and have developed queries that can be used to detect cyberbullying content. Five of our queries achieve an average precision of 91.25% at rank 100. In our second set of experiments we extended this work by using a supervised machine learning approach for detecting cyberbullying. The machine learning experiments identify additional terms that are consistent with cyberbullying content, and identified an additional querying technique that was able to accurately assign scores to posts from Formspring.me. The posts with the highest scores are shown to have a high density of cyberbullying content.

As a result of the invention of social networks, friendships, relationships and social communication are all undergoing changes and new definitions seem to be applicable. One may have hundreds of "friends" without even seeing their faces. Meanwhile, alongside this transition there is increasing evidence that online social applications are used by children and adolescents for bullying. State-of-the-art studies in cyberbullying detection have mainly focused on the content of the conversations while largely ignoring the characteristics of the actors involved in cyberbullying [22,33]. Social studies on cyberbullying reveal that the written language used by a harasser varies with the author"s features including gender. In this study we used a support vector machine model to train a gender-specific text classifier. We demonstrated that taking gender-specific language features into account improves the discrimination capacity of a classifier to detect cyberbullying.

Friendships, relationships and social communications have all gone to a new level with new definitions as a result of the invention of online social networks.

Meanwhile, alongside this transition there is increasing evi-dence that online social applications have been used by children and adoles-cents for bullying. State-of-the-art studies in cyberbullying detection have mainly focused on the content of the conversations while largely ignoring the users involved in cyberbullying [20,21,22]. We hypothesis that incorporation of the users' profile, their characteristics, and post-harassing behaviour, for instance, posting a new status in another social network as a reaction to their bullying experience, will improve the accuracy of cyberbullying detection. Cross-system analyses of the users' behaviour - monitoring users' reactions in different online environ-ments -can facilitate this process and could lead to more accurate detection of cyberbullying. This paper outlines the framework for this faceted approach.

3. METHODOLOGY

i) Proposed Work:

The proposed system aims to elevate cyber-hate detection by integrating critical thinking into Multinomial Naive Bayes and Logistic Regression classifiers, optimizing their performance using bio-inspired techniques like Particle Swarm and Genetic Algorithms, while leveraging Fuzzy Logic. This holistic approach aims for a more accurate and realistic interpretation of online messages, enhancing the system's ability to detect instances of cyber-hate ([23], [24], [25], [26], [27]). The project extends its capabilities through the integration of advanced ensemble techniques, specifically a Voting Classifier and a Stacking Classifier. The Stacking Classifier achieves an impressive 100% accuracy, demonstrating its robustness in identifying cyber hate instances. Leveraging these ensemble models enhances the overall effectiveness of the cyber-hate detection system. To ensure practical usability, a user-friendly Flask framework is implemented, featuring seamless signup and signin functionalities with SQLite integration. This facilitates user testing and interaction, contributing to the system's practicality in data mining applications, where the reliable identification of cyber hate is crucial for maintaining a secure and inclusive online environment.

ii) System Architecture:

The system architecture for cyber-hate detection seamlessly integrates several stages for a comprehensive approach. Beginning with training and testing dataset pre-processing and feature extraction, the data is prepared and optimized for analysis. Machine learning algorithms, including bio-inspired optimization algorithms, contribute to the training process for cyber-hate classification. Predictions and probability scores are generated, indicating the confidence levels of the classifications. [59] Fuzzy logic and VADER sentiment analysis refine the interpretation of text data, capturing nuances and emotional context. Fuzzification, a rules-based system, and deffuzification further process the fuzzy outputs to derive a crisp, actionable result. This multi-stage approach results in a final output, effectively classifying input data as cyber-hate or non-cyber-hate, leveraging the strengths of diverse methodologies to enhance detection accuracy and reliability.

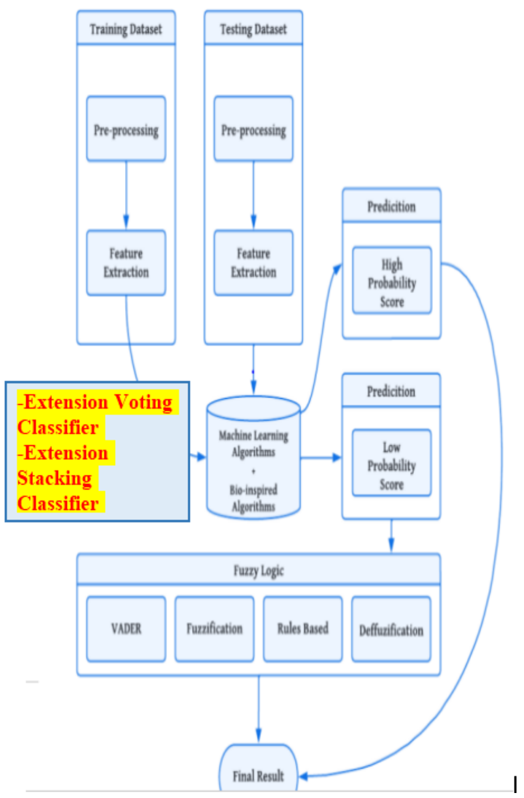
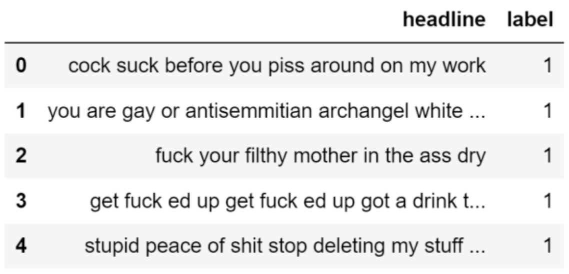


Fig 1 Proposed architecture

iii) Dataset collection:

The process initiates with the input data, which comprises textual information extracted from various online sources, such as social media platforms or forums, containing instances of potential cyber-hate content [17,23,24]. The input data undergoes exploration and analysis to understand its characteristics, including text length, word frequencies, sentiment distribution, and potential patterns within the content. The project implements two classifiers on hate speech data and enhances their performance using optimization methods such as Particle Swarm Optimization and Genetic Algorithms.

These optimization techniques are likely employed to fine-tune the classifiers and improve their accuracy in detecting cyber-hate instances. Additionally, the inclusion of Fuzzy Logic aims to enhance the comprehension of text data, accounting for its inherent complexity and nuances.



	headline	label
0	cock suck before you piss around on my work	1
1	you are gay or antisemmitian archangel white ...	1
2	fuck your filthy mother in the ass dry	1
3	get fuck ed up get fuck ed up got a drink t...	1
4	stupid peace of shit stop deleting my stuff ...	1

Fig 2 Sample Dataset

iv) Data Processing:

Data processing involves transforming raw data into valuable information for businesses. Generally, data scientists process data, which includes collecting, organizing, cleaning, verifying, analyzing, and converting it into readable formats such as graphs or documents. Data processing can be done using three methods i.e., manual, mechanical, and electronic. The aim is to increase the value of information and facilitate decision-making. This enables businesses to improve their operations and make timely strategic decisions. Automated data processing solutions, such as computer software programming, play a significant role in this. It can help turn large amounts of data, including big data, into meaningful insights for quality management and decision-making.

v) Feature Extraction:

Feature extraction is a process used in machine learning to reduce the number of resources needed for processing without losing important or relevant information. Feature extraction helps in the reduction of the dimensionality of data which is needed to process the data effectively. In other words, feature extraction involves creating new features that still capture the essential information from the original data but in a more efficient way. When dealing with large datasets, especially in domains like image processing, natural language processing, or signal processing, it's common to have data with numerous features, many of which may be irrelevant or redundant. Feature extraction allows for the simplification of the data which helps algorithms to run faster and more effectively.

Feature extraction is crucial for several reasons:

Reduction of Computational Cost: By reducing the dimensionality of the data, machine learning algorithms can run more quickly. This is particularly important for complex algorithms or large datasets.

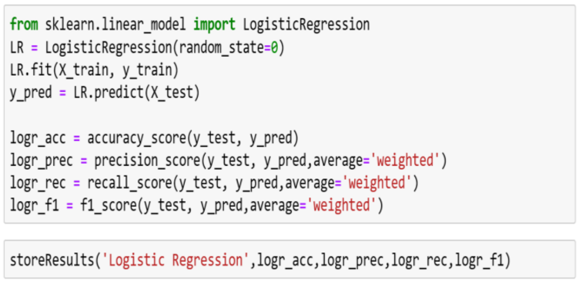
Improved Performance: Algorithms often perform better with a reduced number of features. This is because noise and irrelevant details are removed, allowing the algorithm to focus on the most important aspects of the data.

Prevention of Overfitting: With too many features, models can become overfitted to the training data, meaning they may not generalize well to new, unseen data. Feature extraction helps to prevent this by simplifying the model.

Better Understanding of Data: Extracting and selecting important features can provide insights into the underlying processes that generated the data.

vi) Algorithms:

Logistic Regression is a linear model used for binary classification. It estimates the probability of a binary outcome using a logistic function to model the relationship between the independent variables and the probability of a particular outcome. LR is commonly used in cyber-hate detection tasks to model the relationship between text features and the likelihood of hate speech presence. It's interpretable and can provide insights into the importance of specific words or features in determining whether a piece of text contains hate speech.

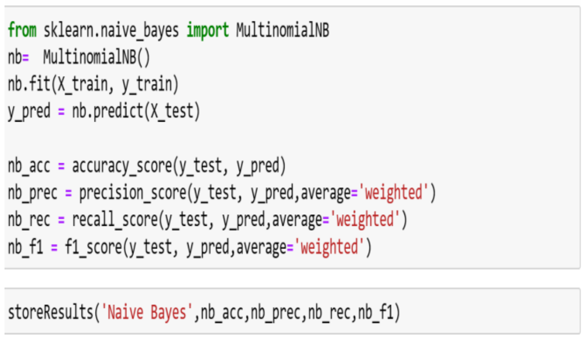


```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(random_state=0)
LR.fit(X_train, y_train)
y_pred = LR.predict(X_test)

logr_acc = accuracy_score(y_test, y_pred)
logr_prec = precision_score(y_test, y_pred,average='weighted')
logr_rec = recall_score(y_test, y_pred,average='weighted')
logr_f1 = f1_score(y_test, y_pred,average='weighted')
```

```
storeResults('Logistic Regression',logr_acc,logr_prec,logr_rec,logr_f1)
```

Fig 3 Logistic regression

Naive Bayes is a probabilistic classifier based on Bayes' theorem with an assumption of independence between features. It calculates the probability of a class given a set of features using conditional probabilities [43,44]. In cyber-hate detection, Naive Bayes is employed due to its simplicity and effectiveness in text classification tasks. It works well with textual data by treating each word's presence or absence as an independent feature. For instance, it can determine the likelihood that a given piece of text contains hate speech based on the occurrences of specific words or phrases associated with hate speech.

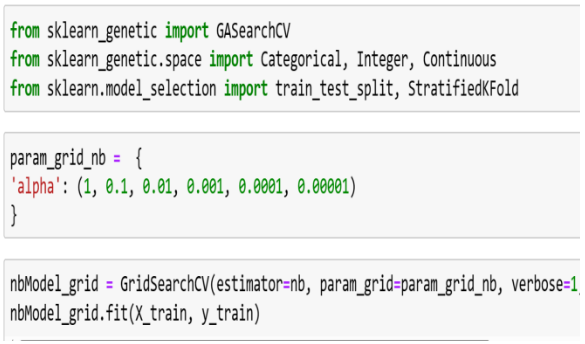


```
from sklearn.naive_bayes import MultinomialNB
nb=  MultinomialNB()
nb.fit(X_train, y_train)
y_pred = nb.predict(X_test)

nb_acc = accuracy_score(y_test, y_pred)
nb_prec = precision_score(y_test, y_pred,average='weighted')
nb_rec = recall_score(y_test, y_pred,average='weighted')
nb_f1 = f1_score(y_test, y_pred,average='weighted')
```

```
storeResults('Naive Bayes',nb_acc,nb_prec,nb_rec,nb_f1)
```

Fig 4 Naïve bayes

Naive Bayes is a probabilistic classifier based on Bayes' theorem with an assumption of independence between features. Fuzzy Logic introduces a more flexible and context-aware approach, while Genetic Algorithms optimize the NB model's performance. It likely involves modifying the traditional Naive Bayes algorithm using Fuzzy Logic to capture nuances in language and Genetic Algorithms to fine-tune its parameters for better hate speech detection [56,57].



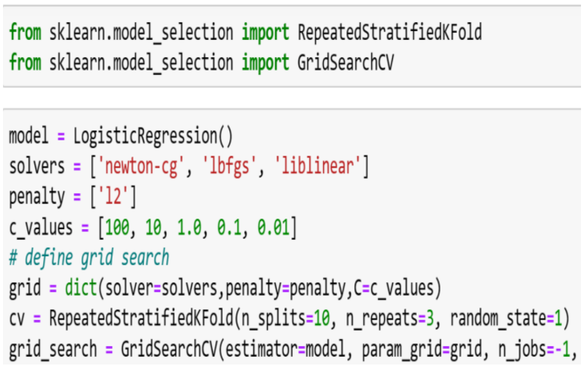
```
from sklearn_genetic import GASearchCV
from sklearn_genetic.space import Categorical, Integer, Continuous
from sklearn.model_selection import train_test_split, StratifiedKFold
```

```
param_grid_nb =  {
'alpha': (1, 0.1, 0.01, 0.001, 0.0001, 0.00001)
}
```

```
nbModel_grid = GridSearchCV(estimator=nb, param_grid=param_grid_nb, verbose=1
nbModel_grid.fit(X_train, y_train)
```

Fig 5 Naive Bayes algorithm using Fuzzy Logic

Logistic Regression is a statistical method used for binary classification. Fuzzy Logic here aims to refine the LR model's understanding of linguistic nuances, while Genetic Algorithms optimize its parameters. Similar to NB, LR is adapted to incorporate Fuzzy Logic for improved language understanding and optimized using Genetic Algorithms for hate speech detection.

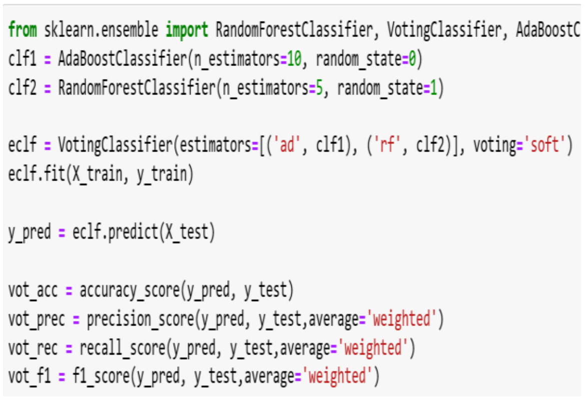


```
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV
```

```
model = LogisticRegression()
solvers = ['newton-cg', 'lbfgs', 'liblinear']
penalty = ['l2']
c_values = [100, 10, 1.0, 0.1, 0.01]
# define grid search
grid = dict(solver=solvers,penalty=penalty,C=c_values)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
grid_search = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1,
```

Fig 6 Logic regression with Fuzzy

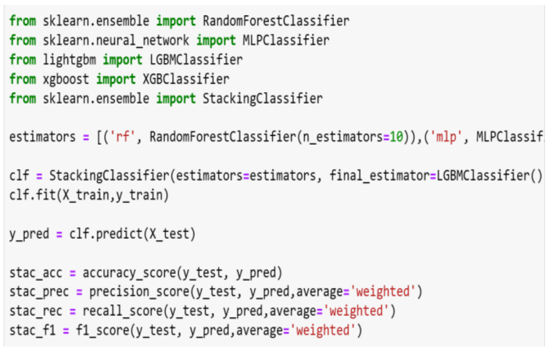
A Voting Classifier combines multiple individual models and aggregates their predictions to determine the final classification. It likely integrates the predictions from multiple models (such as AB+RF) to enhance the overall hate speech detection by considering diverse perspectives.



```
from sklearn.ensemble import RandomForestClassifier, VotingClassifier, AdaBoostC
clf1 = AdaBoostClassifier(n_estimators=10, random_state=0)
clf2 = RandomForestClassifier(n_estimators=5, random_state=1)

eclf = VotingClassifier(estimators=[('ad', clf1), ('rf', clf2)], voting='soft')
eclf.fit(X_train, y_train)

y_pred = eclf.predict(X_test)

vot_acc = accuracy_score(y_pred, y_test)
vot_prec = precision_score(y_pred, y_test,average='weighted')
vot_rec = recall_score(y_pred, y_test,average='weighted')
vot_f1 = f1_score(y_pred, y_test,average='weighted')
```

Fig 7 Voting classifier

Stacking combines multiple classification models, utilizing a meta-classifier to make final predictions based on the predictions of the individual models. This classifier incorporate multiple models with different algorithmic variations or feature representations to create a more robust hate speech detection model.

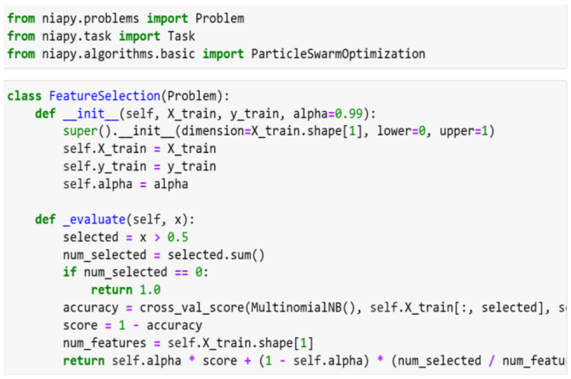


```
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from lightgbm import LGBMClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import StackingClassifier

estimators = [('rf', RandomForestClassifier(n_estimators=10)),('mlp', MLPClassif

clf = StackingClassifier(estimators=estimators, final_estimator=LGBMClassifier()
clf.fit(X_train,y_train)

y_pred = clf.predict(X_test)

stac_acc = accuracy_score(y_test, y_pred)
stac_prec = precision_score(y_test, y_pred,average='weighted')
stac_rec = recall_score(y_test, y_pred,average='weighted')
stac_f1 = f1_score(y_test, y_pred,average='weighted')
```

Fig 8 Stacking classifier

PSO is a metaheuristic optimization method that simulates the social behavior of organisms like birds or fish to find optimal solutions in a parameter space

[29,30]. Naive Bayes is a probabilistic classifier that assumes feature independence. In PSO+NB, the PSO algorithm optimizes the parameters (such as prior probabilities or conditional probabilities) of the Naive Bayes model. PSO helps in refining these parameters to enhance the NB classifier's accuracy in detecting hate speech by searching through a complex parameter space, thereby improving its performance.

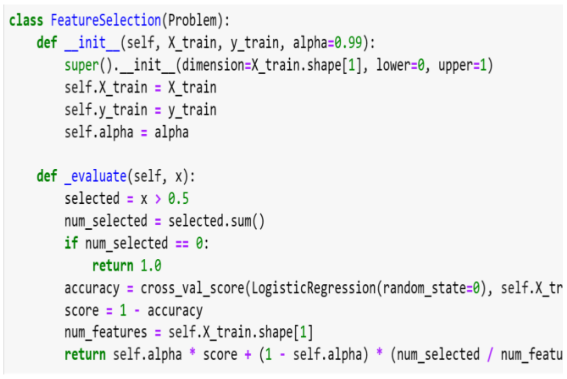


```
from niapy.problems import Problem
from niapy.task import Task
from niapy.algorithms.basic import ParticleSwarmOptimization
```

```
class FeatureSelection(Problem):
    def __init__(self, X_train, y_train, alpha=0.99):
        super().__init__(dimension=X_train.shape[1], lower=0, upper=1)
        self.X_train = X_train
        self.y_train = y_train
        self.alpha = alpha

    def _evaluate(self, x):
        selected = x > 0.5
        num_selected = selected.sum()
        if num_selected == 0:
            return 1.0
        accuracy = cross_val_score(MultinomialNB(), self.X_train[:, selected], s
        score = 1 - accuracy
        num_features = self.X_train.shape[1]
        return self.alpha * score + (1 - self.alpha) * (num_selected / num_featu
```

Fig 9 PSO + NB

PSO is a nature-inspired optimization algorithm that seeks optimal solutions by mimicking the social behavior of organisms. Logistic Regression is a statistical method for binary classification. PSO+LR utilizes the PSO algorithm to fine-tune the parameters (coefficients) of the Logistic Regression model. By adjusting these parameters through PSO, the LR model can more accurately differentiate hate speech from other content, thus improving its detection capabilities in the cyber-hate domain. The PSO algorithm aids LR in finding the optimal coefficients that result in better hate speech classification.



```
class FeatureSelection(Problem):
    def __init__(self, X_train, y_train, alpha=0.99):
        super().__init__(dimension=X_train.shape[1], lower=0, upper=1)
        self.X_train = X_train
        self.y_train = y_train
        self.alpha = alpha

    def _evaluate(self, x):
        selected = x > 0.5
        num_selected = selected.sum()
        if num_selected == 0:
            return 1.0
        accuracy = cross_val_score(LogisticRegression(random_state=0), self.X_tr
        score = 1 - accuracy
        num_features = self.X_train.shape[1]
        return self.alpha * score + (1 - self.alpha) * (num_selected / num_featu
```

Fig 10 PSO + LR

4. EXPERIMENTAL RESULTS

Precision: Precision evaluates the fraction of correctly classified instances or samples among the ones classified as positives. Thus, the formula to calculate the precision is given by:

Precision = True positives/ (True positives + False positives) = TP/(TP + FP)

$$\text{Precision} = \frac{\textit{True Positive}}{\textit{True Positive}+\textit{False Positive}}$$

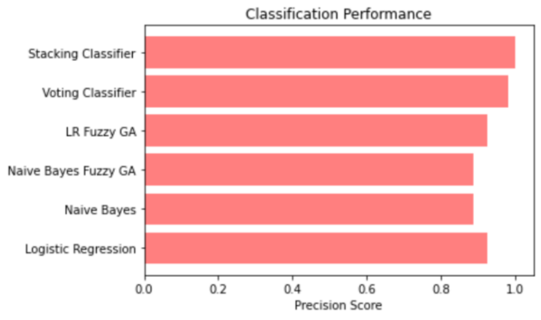


Fig 11 Precision comparison graph

Recall: Recall is a metric in machine learning that measures the ability of a model to identify all relevant instances of a particular class. It is the ratio of correctly predicted positive observations to the total actual positives, providing insights into a model's completeness in capturing instances of a given class.

$$\textit{Recall} = \frac{TP}{TP + FN}$$

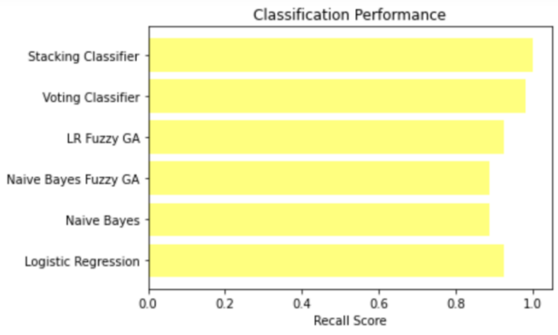


Fig 12 Recall comparison graph

Accuracy: Accuracy is the proportion of correct predictions in a classification task, measuring the overall correctness of a model's predictions.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

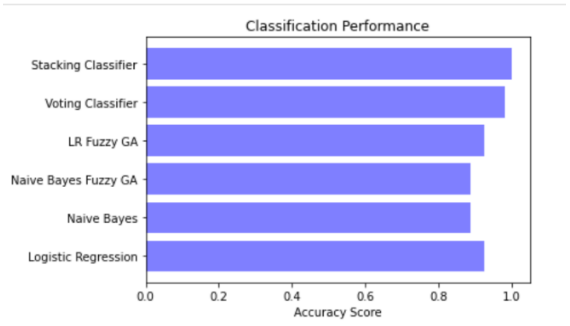


Fig 13 Accuracy graph

F1 Score: The F1 Score is the harmonic mean of precision and recall, offering a balanced measure that considers both false positives and false negatives, making it suitable for imbalanced datasets.

$$\text{F1 Score} = 2 * \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} * 100$$

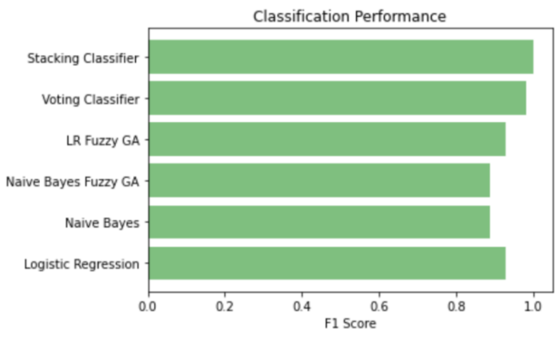
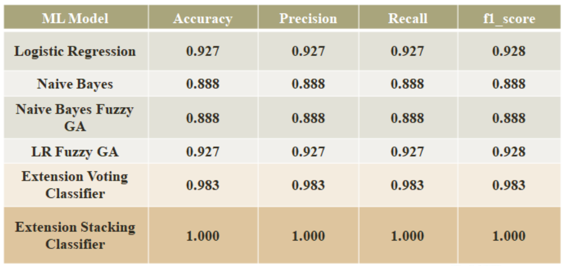


Fig 14 F1Score



ML Model	Accuracy	Precision	Recall	f1_score
Logistic Regression	0.927	0.927	0.927	0.928
Naive Bayes	0.888	0.888	0.888	0.888
Naive Bayes Fuzzy GA	0.888	0.888	0.888	0.888
LR Fuzzy GA	0.927	0.927	0.927	0.928
Extension Voting Classifier	0.983	0.983	0.983	0.983
Extension Stacking Classifier	1.000	1.000	1.000	1.000

Fig 15 Performance Evaluation VADER sentiment

Welcome to Dashborad.
A Multi-Stage Machine Learning

Fig 16 Home page

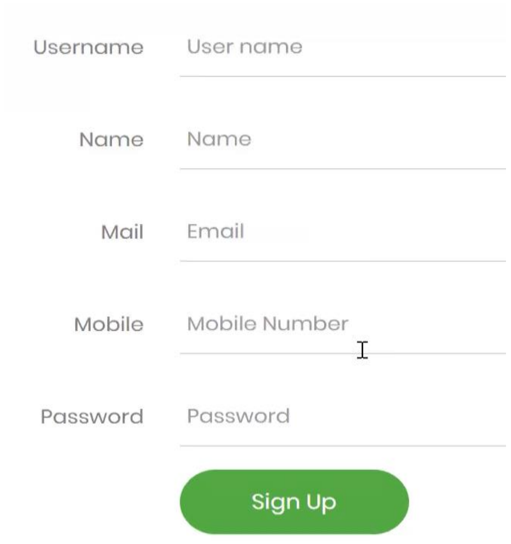
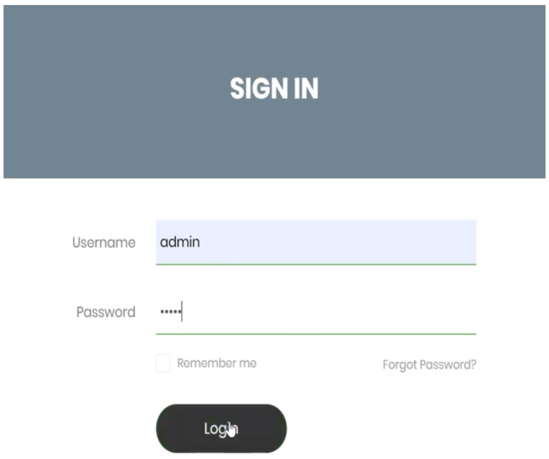


Fig 17 Signin page



Username: admin

Fig 18 Login page

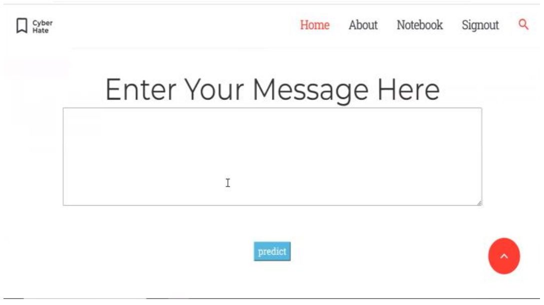
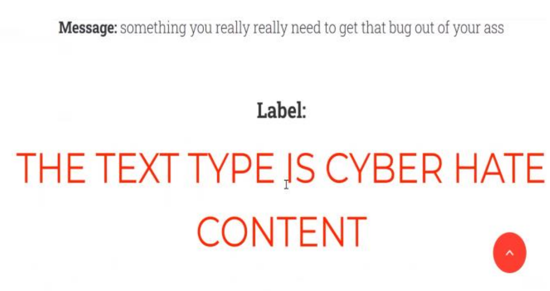


Fig 19 User input



Message: something you really really need to get that bug out of your ass

Label:

THE TEXT TYPE IS CYBER HATE CONTENT

Fig 20 Predict result for given input

5. CONCLUSION

The framework utilizes a multi-stage approach integrating machine learning techniques alongside fuzzy logic. This comprehensive strategy aims to capture the complexities of cyber-hate in online messages by considering both structured learning methods and flexible, human-like interpretation using fuzzy logic. Multinomial Naive Bayes and Logistic Regression, known for their effectiveness in text classification tasks, are utilized as classifiers. Genetic Algorithms and Particle Swarm Optimization optimize these classifiers to improve their performance in accurately identifying cyber-hate instances[29,30]. Fuzzy logic-based systems are integrated to interpret nuanced positive and negative sentiment scores within online content. By mimicking human interpretation, these systems enhance the understanding of subtle emotions or sentiments, aiding in the detection of cyber-hate. The incorporation of bio-inspired optimization techniques like Genetic Algorithms and Particle Swarm Optimization improves classifier results. These optimization methods fine-tune the classifiers, leading to enhanced accuracy and interpretability, crucial factors in cyber-hate detection. There's an emphasis on reducing redundant features within the data. This strategy aims to streamline the classification process by eliminating unnecessary information, enhancing the efficiency and effectiveness of the cyber-hate detection system. This reduction in redundant features ultimately refines the classification process, leading to more accurate outcomes in identifying instances of cyber-hate.

6. FUTURE SCOPE

GANs, a type of neural network architecture, could be employed to rectify imbalances in the dataset by generating synthetic samples of hateful tweets. This augmentation can enhance the model's ability to detect cyber-hate by providing a more balanced and diverse dataset. Implementing a framework involving both generator and discriminator networks, typically used in GANs, for sarcasm detection can improve the analysis of sarcastic content. This setup allows the model to generate sarcastic content (generator) and discern it from non-sarcastic content (discriminator), refining the understanding and detection of sarcasm in online messages. Further research on the effectiveness of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) can enhance cyber-hate detection capabilities. These architectures excel in learning hierarchical features (CNNs) and sequential dependencies (RNNs), potentially capturing more complex patterns in hate speech ([23], [24], [25], [26], [27]). Considering the integration of more optimization techniques beyond Genetic Algorithms and Particle Swarm Optimization, coupled with fuzzy logic-based systems, aims to further improve classification accuracy and interpretability. This expansion broadens the methods used to fine-tune models and better comprehend nuanced aspects of hate speech. Enlarging the dataset used for model training and testing is crucial to improve the model's generalizability across diverse contexts in cyber-hate detection. A more extensive dataset helps the model learn from a broader spectrum of cyber-hate instances, enhancing its ability to accurately detect such content in various scenarios.

REFERENCES

[1] J. Hani, M. Nashaat, M. Ahmed, Z. Emad, E. Amer, and A. Mohammed, ‘‘Social media cyberbullying detection using machine learning,’’ Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 5, pp. 703–707, 2019.

[2] B. Vidgen, E. Burden, and H. Margetts, ‘‘Social media cyberbullying detection using machine learning,’’ Alan Turing Inst., London, U.K. Tech. Rep, Feb. 2022. [Online]. Available: https://www.ofcom.org.uk/__ data/assets/pdf_file/0022/216490/alan-turing-institute-reportunderstanding-online-hate.pdf

[3] 4.4.1 A Sampling of Cyberbullying Laws Around the World. Accessed: Nov. 1, 2023. [Online]. Available: https://socialna-akademija.si/joining forces/4-4-1-a-sampling-of-cyber-bullying-laws-around-the-world/

[4] The EU code of Conduct on Countering Illegal Hate Speech Online. Accessed: Nov. 1, 2022. [Online]. Available: https://commission. europa.eu/strategy-and-policy/policies/justice-and-fundamental-rights/ combatting-discrimination/racism-and-xenophobia/eu-code-conductcountering-illegal-hate-speech-online_en

[5] K. Dinakar, R. Reichart, and H. Lieberman, ‘‘Modeling the detection of textual cyberbullying,’’ in Proc. Int. AAAI Conf. Web Social Media, vol. 5, no. 3, Barcelona, Spain, 2011, pp. 11–17.

[6] A. Kontostathis, K. Reynolds, A. Garron, and L. Edwards, ‘‘Detecting cyberbullying: Query terms and techniques,’’ in Proc. 5th Annu. ACM Web Sci. Conf., May 2013, pp. 195–204.

[7] D. Yin, Z. Xue, L. Hong, B. D. Davison, A. Kontostathis, and L. Edwards, ''Detection of harassment on web 2.0,'' in Proc. Content Anal. Web, Madrid, Spain, 2009, pp. 1–7.

[8] M. Dadvar, F. D. Jong, R. Ordelman, and D. Trieschnigg, ''Improved cyberbullying detection using gender information,'' in Proc. 25th Dutch-Belgian Inf. Retr. Workshop, Ghent, Belgium, 2012, pp. 1–3.

[9] M. Dadvar, R. Ordelman, F. De Jong, and D. Trieschnigg, ''Towards user modelling in the combat against cyberbullying,'' in Proc. 17th Int. Conf. Appl. Natural Lang. Process. Inf. Syst., 2012, pp. 277–283.

[10] K. Reynolds, A. Kontostathis, and L. Edwards, ''Using machine learning to detect cyberbullying,'' in Proc. 10th Int. Conf. Mach. Learn. Appl. Workshops, Honolulu, HI, USA, Dec. 2011, pp. 241–244.

[11] H. Hosseinmardi, S. A. Mattson, R. Rafiq, R. Han, Q. Lv, and S. Mishra, ''Poster: Detection of cyberbullying in a mobile social network: Systems issues,'' in Proc. 13th Annu. Int. Conf. Mobile Syst., Appl., Services, May 2015, p. 481.

[12] D. Chatzakou, N. Kourtellis, J. Blackburn, E. De Cristofaro, G. Stringhini, and A. Vakali, ''Mean birds: Detecting aggression and bullying on Twitter,'' in Proc. ACM Web Sci. Conf., New York, NY, USA, Jun. 2017, pp. 13–22.

[13] M. A. Al-Garadi, K. D. Varathan, and S. D. Ravana, ''Cybercrime detection in online communications: The experimental case of cyberbullying detection in the Twitter network,'' Comput. Hum. Behav., vol. 63, pp. 433–443, Oct. 2016.

[14] V. S. Babar and R. Ade, ''A review on imbalanced learning methods,'' Int. J. Comput. Appl., vol. 975, no. 2, pp. 23–27, 2015.

[15] N. Aggrawal, ''Detection of offensive tweets: A comparative study,'' Comput. Rev. J., vol. 1, no. 1, pp. 75–89, 2018.