International Journal of Engineering Trends and Technology
ISSN: 2231-5381/ https://doi.org/10.14445/22315381/IJETT-V74I8P111

Volume 74 Issue 8, 165-181, August 2026

Original Article

Performance Improvement using Fuzzy String Matching Based Features for Sentiment Analysis on Social Media Data

Amit Kumar Jadiya[1], Ramesh Thakur[2], Archana Thakur[3]

[1]*Institute of Engineering & Technology, DAVV Indore, India.*
[2]*International Institute of Professional Studies, DAVV, Indore, India.*
[3]*Shool of CS & IT, DAVV, Indore, India.*

[1]*Corresponding Author : amitjadiya@gmail.com*

Received: 26 December 2024 Revised: 05 July 2025 Accepted: 11 July 2026 Published: 29 August 2026

Abstract - *This paper presents Nested Social Sentiments Classification (NeSS-Class), a sentiment analysis framework that incorporates both primary posts and their nested comments from social media platforms. Unlike traditional approaches, NeSS-Class introduces derived features based on fuzzy string matching to capture subtle textual similarities and to address challenges arising from complex user interaction behaviors in nested discussions. Data was gathered from multiple social media platforms, carefully preprocessed it, and divided into training, validation, and testing sets in an 80-10-10 ratio. Feature stability was evaluated using univariate analysis, and baseline machine learning models were employed for performance assessment. Experimental results demonstrate that Logistic Regression integrated with NeSS-Class significantly improves classification performance, achieving a log-loss value of 0.6553, compared to 0.9099 obtained without the proposed feature set. These results confirm the effectiveness of fuzzy string matching as a feature engineering strategy for enhancing sentiment analysis in noisy, multi-layered social media data.*

Keywords - *Sentiment analysis, Social media data, Natural Language Processing (NLP), Machine learning, Fuzzy string-matching features.*

1. Introduction

Nowadays, social media websites are not only used for chatting with friends and sharing pictures; they are also used as the most significant marketing tools. Facebook, Twitter, YouTube, and others have emerged as standard tools for disseminating ideas and opinions to a large audience [1, 2]. These include many news articles, product promotion advertisements, discussions about ongoing political issues, new trends and technologies, and many more. To gain valuable insights from social media data, extensive research has focused on sentiment analysis using lexicon-based, machine learning, and hybrid approaches.

Existing sentiment analysis methods primarily perform flat classification by assigning a single polarity (positive, negative, or neutral) to social media text. However, social media content often contains complex and layered opinions expressed through informal language, contextual dependencies and, and emojis [4]. These characteristics make it difficult for conventional ML and deep learning models to accurately capture the underlying sentiment structure. Moreover, current approaches rarely consider hierarchical or nested relationships among sentiments within the same text. Therefore, there is a need for a robust and scalable framework capable of identifying and modeling multi-level sentiment patterns in social media data. The proposed NeSS-Class (Nested Social Sentiments Classification) model aims to address this challenge by introducing a structured classification approach that can effectively capture nested sentiment relationships and improve sentiment analysis performance.

Lexicon-based methods are divided into two categories: corpus-based and dictionary-based. Whereas corpus-based sentiment analysis relies on statistical analysis of content in the document, other depends on pre-existing dictionaries. The machine learning approach uses text messages to train models to automatically recognize sentiment. The hybrid approach is another one where lexicon-based techniques are getting integrated with ML methods [4, 5].

In the Machine learning approach, ML models such as K-Nearest Neighbour (KNN), Support Vector Machines (SVMs), Naive Bayes (NB), Logistic Regression (LR), Linear

Regression, and Random Forests are used across different areas for sentiment analysis. Deep learning models have also performed well in sentiment analysis in recent years. Extensive research has been conducted on Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTMs), and their various combinations [6, 7].

A recent deep learning-based model for consumer sentiment analysis uses a hybrid feature extraction approach. The feature extraction process has two sections. The first is Review-Related Features (RRF), which employs several techniques to improve the accuracy of emotional representations, including n-gram features, emoticon polarity, and term frequency-inverse document frequency.

The second section, Aspect-Related Features (ARF), assigns polarities after extracting aspect terms based on co-occurrence frequencies. The Hybrid Feature Vector (HFV) is generated by merging RRF and ARF, an efficient approach that provides high accuracy [8].

The sentiment analysis method considers only subject-specific text comments. However, nested comments are also critical in social media data, and some approaches need to be studied to improve sentiment analysis accuracy by leveraging them.

This study proposes an efficient approach that introduces a new set of features based on fuzzy string-matching techniques. In this work, the term subject denotes the primary post or main comment that initiates the discussion, while the subsequent responses provided by users in the form of comments, reviews, or opinions are considered as nested comments associated with that subject.

As mentioned, the impact on polarity due to nested comments is a significant challenge. For Example, as shown in Figure 1, the sentiment classification for comment 1.1 in the dataset is “positive.” But when we see its subject—“Apple urges users to update devices following security threats", also, this comment should be classified as a Negative sentiment.

Another example for comment 2.1 in Figure 1 - “Amazing Nick, best analysis on YouTube. Keep up the good work” is positive in the training dataset, which is correct. But for which subject is this comment? This review is for the subject “Is the BITCOIN crash almost over or is it just getting started? Crypto News Today” and subjective details are also needed before assigning any polarity.

The following is a summary of this study's contribution:

1) This study proposes NeSS-Class, a structured framework designed to improve sentiment analysis by capturing relationships between primary posts and their associated nested comments in social media discussions.
2) A novel set of five fuzzy-based string-matching features is introduced to better capture textual similarity and contextual relationships between subjects and nested comments.
3) A univariate statistical analysis is performed to evaluate feature stability and identify the most informative features for sentiment classification.
4) The proposed framework enhances classification effectiveness by incorporating engineered features and structured sentiment relationships within the dataset.
5) The effectiveness of the proposed method is validated by comparing it with baseline models using multiple evaluation metrics, including log loss, confusion matrix, precision, and recall.

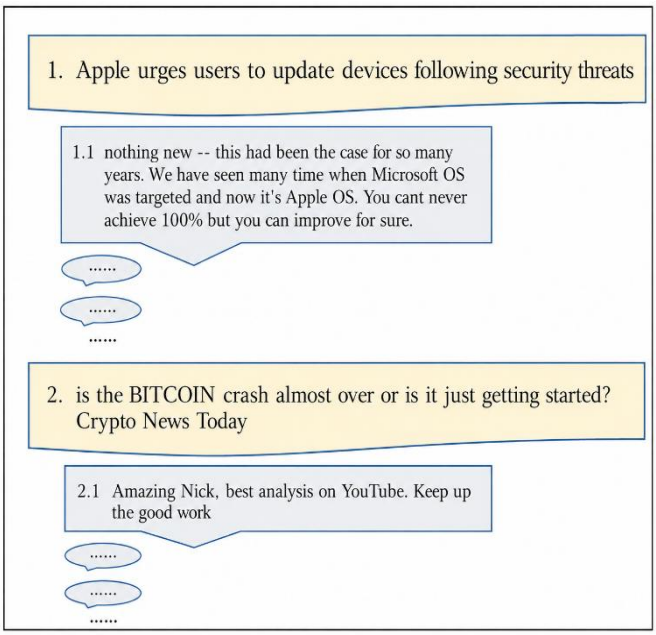

Fig. 1 Example of nested comments

After pre-processing and integrating social media data, this research aims to classify sentiments into three classes: positive, negative, and neutral, where nested comments are also considered for sentiment analysis. We proposed new features for sentiment classification using fuzzy string-matching techniques, achieving better accuracy than baseline models. We have compared results with baseline models and demonstrated effectiveness using performance metrics such as log loss, confusion matrices, precision, and recall.

The paper structure is as follows: Section 2 contains the literature studies of sentiment analysis approaches for social media data. In Section 3, the technologies and tools used for the proposed framework are explained. The proposed NeSS-Class algorithm and theoretical framework are described in Section 4. The experiment, evaluation and outcomes are presented in Section 5 and 6. In Section 7, conclusions are presented.

2. Literature Review

Recent research in sentiment analysis has enhanced the capability of identifying opinions and emotional expressions from social media content through the use of machine learning and deep learning techniques. Earlier studies mainly concentrated on document-level and sentence-level sentiment classification using lexicon-based approaches and traditional machine learning models such as Naïve Bayes, Support Vector Machines (SVM), Logistic Regression, and Random Forests.

Although these approaches produced satisfactory results for sentiment polarity prediction, they generally analyzed text independently and were unable to effectively represent contextual relationships and conversational interactions available in social media discussions.

Researchers have also investigated various feature extraction and text representation methods to improve sentiment classification performance. Popular techniques include Bag-of-Words (BoW), TF-IDF, Word2Vec, One-Hot Encoding, and Response Coding for converting textual information into numerical form suitable for machine learning algorithms. Despite their effectiveness, these methods have certain limitations, including sparse feature representation, inadequate contextual understanding, and limited capability to capture relationships between original posts and their corresponding nested comments or replies.

Also, in the existing studies, most sentiment analysis approaches are developed using either single-source datasets or limited heterogeneous data. However, social media data is generated from multiple platforms such as Facebook, Twitter, YouTube, and online forums, where each source contains different data structures, formats, attributes, and quality-related issues. Integrating such multi-source social media data is a challenging task due to problems such as schema heterogeneity, noisy content, missing values, inconsistent feature representation, and varying data scales.

This section briefly reviews the existing work related to preprocessing and sentiment classification approaches for social media and Social Big Data analytics. For the pre-processing part, an effective method, Polymorphic Social Big Data Preprocessor, is used, which addresses issues such as noise, inconsistency, missing data, and irrelevant data. As demonstrated through a case study, the suggested approach yields output data, the ideal source for the next level of data processing, such as feature extraction and use in machine learning algorithms.

The Social Big Data preprocessor framework takes input as a set of datasets from multiple sources and produce preprocessed, integrated output. Machine learning approaches can use this dataset for further steps in sentiment classification.

Lombardo et al. [3] used a Facebook-provided dataset of 50,000 manually annotated posts and comments from 2009 to 2017 for Italian patients with hidradenitis suppurativa. They have used social network sentiment analysis tools to create Facebook support groups for these patients. Over the years of observation, they examined how different social network elements, such as friendships and group interactions, impacted their mood.

After organizing the reviews hierarchically, the sentiments were categorized as surprise, joy, love, fear, sadness, and rage. Furthermore, this study created a communication network that examined connections between review authors and respondents, and a social network that focused on users and their friendships.

Smiles and emotions were replaced with suitable terms for the sentiment analysis. For the classifier, posts with positive labels were grouped into categories such as love, joy, and surprise, whereas posts with negative labels were grouped into categories such as fear, anger, and sadness. The Naive Bayes Multinomial technique was used to train the classifiers, and their accuracy was 0.490.

When COVID-19 cases increased, Chandra et al. [9] employed deep learning-based language models, including Long Short-Term Memory (LSTM) recurrent neural networks, for sentiment analysis of tweets related to novel cases in India. They employed LSTM and bidirectional LSTM models with Global Vector (GloVe) word representations to construct a language model.

They presented a multi-label sentiment categorization framework that enables the expression of several sentiments simultaneously. They used the COVID-19 sentiment dataset, comprising 10,000 tweets collected worldwide and manually classified by 50 professionals, to train LSTM models. Their proposed framework consists of four primary parts: 1.) extracting tweets; 2.) pre-processing tweets; 3.) creating and training models with LSTM, Bi-directional LSTM, and BERT; and 4.) Making predictions using particular COVID-19 data.

They assessed the model's capacity to manage multi-label classification problems using the BCE Loss, Hamming Loss, Jaccard Score, and LRAP Score metrics. The accuracy with BERT is F1 micro = 0.587 and F1 macro = 0.530.

A novel deep neural network model for sentiment analysis based on grid search and Long Short-Term Memory (LSTM) Convolutional Neural Networks (CNN) was proposed by Priyadarshini et al., [10]. The baseline algorithms studied are CNN–LSTM, LSTM–CNN, K-nearest neighbor, convolutional neural networks, and neural networks. These algorithms have been assessed using accuracy, precision, and F-1 score on several datasets, including the IMDB dataset,

which contains 50,000 movie reviews, and the Amazon product review dataset, which contains approximately 29 lacs product reviews.

Their proposed study uses a unique LSTM–CNN–grid-search-based deep neural network to analyze text sentiment. Grid search and the fully connected neural network are combined with the LSTM–CNN model. The grid search aims to identify the best hyperparameters for more precise sentiment polarity classification. Both datasets' results demonstrate perfect accuracy.

Ariel et al., [5] compared several research issues pertaining to the data sets, text languages, analysis techniques, and assessment criteria for sentiment analysis of social media data. They have created a taxonomy of sentiment analysis in social media that includes information about the dataset, source, volume, domain, sentiment analysis methodology, algorithms used or suggested, and corresponding outcomes. This study is a great way to learn about all the work done on sentiment analysis in social media.

Regarding the work done in by using Fuzzy string-matching techniques, Pietras [11] proposed a fuzzy logic–based sentiment analysis approach using fuzzy word matching and a modified Levenshtein distance algorithm to handle variations in textual data.

The study used fuzzy sentiment classification for sentence-level polarity prediction and demonstrated that fuzzy matching improves sentiment estimation in noisy text environments. However, the work was limited to sentence-level analysis and did not consider nested conversational structures, multi-platform social media data, or large-scale Social Big Data processing.

We have studied further research by Talaat et al., [1], Ibanez et al., [12], and others, which use different approaches, domains, and algorithms. Most of them do not discuss deriving insights from nested reviews or comments, where a relationship can be established with the respective main comment/review to improve sentiment analysis efficiency.

Also studied recent work on how ML models are used in other areas, such as Teng et al., [21], who proposed PSO_KFCM, which combines Particle Swarm Optimization (PSO) with hybrid optimization and the kernel fuzzy C-means clustering technique to cluster similar recommendation data into a single class.

3. Techniques and Tools used in the Proposed NeSS-Class Framework

Below are details about the techniques and tools used in the proposed method. These include Univariate analysis, Feature selection, and Fuzzy string-matching techniques.

3.1. Univariate Analysis

One crucial element in determining the significance of the features is univariate analysis, a data visualization method that allows us to examine one variable at a time. Univariate analysis helps examine the distribution of variables within the data, enabling further research.

3.2. Fuzzy String Matching-Based Features

For Machine learning models, extracted features must be converted into a mathematical form. To address the crucial problem of retrieving mathematical text from data, we have examined three distinct algorithms: fuzzy-wuzzy, Levenshtein Distance, and sequence matcher.

Four distinct Fuzzy-Wuzzy variations are identified as relevant to this study, and their effectiveness in recovering mathematical texts is evaluated through time-series exploration, sensitivity analysis, and efficiency measures. The four variations are the Simple Ratio, Partial Ratio, Token Sort Ratio, and Token Set Ratio.

The degree to which two strings resemble one another is called string similarity, and the process of comparing two strings and calculating a measure of their match is called string matching. For instance, "apple" and "apples" are similar because they differ by only one letter. However, the terms "apple" and "banana" are not very similar due to the very different letters.

Fuzzy string-matching methods help find matches that are close enough, even if they have some differences. This can be helpful for several purposes, including comparing names, correcting spelling errors, and finding related words [13, 14].

Subject and nested comments may or may not have similarities. If the subject is biased towards positive or negative in nature, then respective nested comments may also have high similarity between the subject and nested comments. String-matching approaches help calculate such similarity, thereby improving the accuracy of sentiment classification problems.

We have derived a new set of features using the fuzzy algorithms mentioned below.

1) Simple Ratio: The simple ratio algorithm determines the similarity ratio by comparing how many characters there are in both strings overall with the number of matching characters. Word order and partial matches are not taken into account.
2) Partial Ratio: The partial ratio algorithm finds the best matching substring inside the longer text by considering partial matches.
3) Token Sort Ratio: After tokenizing both strings into separate words and sorting them alphabetically, the token

sort ratio algorithm determines the similarity ratio. It disregards the order of the words.

4) Token Set Ratio: After tokenizing the strings into distinct words and removing duplicates, the token set ratio algorithm determines the similarity ratio. Additionally, it also disregards word order.

3.3. Feature Selection

Massive samples and features in the big data era have significantly increased the processing burden and introduced the "curse of dimensionality", a term that describes several phenomena that occur when evaluating data in high-dimensional space.

It makes data sparser, and training models requires many samples, making data processing much more challenging. One of the best methods for addressing the aforementioned issue and reducing dimensionality is feature selection.

It has been widely used as an efficient pre-processing tool in applications like data mining, pattern recognition, and machine learning [15, 16].

3.4. Feature Engineering

Numerous novel features are being added to social media platforms these days, leading to an increase in the number of features. The complexity of the problem can be seen in the vast number of features (attributes) used to classify reviews. Good features with better predictive power are essential for solving complicated classification problems.

A critical phase in the feature engineering process is currently not well supported, even though several dimensionality reduction approaches, such as feature selection, are available to aid in many parts of learning classification problems.

A significant feature set must be created to categorize objects and make highly accurate predictions [17, 18].

3.5. One Hot Engineering

We have used one-hot encoding for text data and categorical features because it is more efficient than alternatives, such as response coding.

Since one-hot encoding can more accurately depict the data and its relationships, it is more expressive than integer encoding. It can more accurately indicate whether a category exists.

The one-hot encoded representation of a sample that falls into the "Red" and "Green" categories, for instance, would be [1, 0, 1] (with a 0 in the second column and a 1 in the first and third columns). The sample falls into the "Red" and "Green" categories but not the "Blue" category, as shown. On the other hand, the identical sample's integer encoded representation would be [1, 2], which offers no insight into whether the "Blue" category is present.

4. The Proposed Nested Social Sentiments Classification (NeSS-Class) Framework

In this paper, the NeSS-Class (Nested Social Sentiments classification) framework is proposed, which generates new features from the subject and its respective nested comments available on social media.

We have shown that, with derived features for sentiment classification, the Logistic regression model achieves the highest accuracy among baseline models such as Naive Bayes, SVM, and Random Forest.

4.1. NeSS-Class Framework

Figure 2 shows the design and functionality of the NeSS-Class framework, which has three main modules: (A) Data collection and Integration, (B) Feature Engineering, and (C) Classification. Algorithm 1 displays the pseudo-code for the suggested work strategy.

4.1.1. Data Preparation: Data collection and Integration

For our study, the final dataset [19] comprises multiple social media sources with varying schemas. After preprocessing (missing value imputation, noise removal, etc.) at the source level, data aggregation is needed to create a single data source for further processing. For source-level dataset pre-processing and data integration to create a single dataset (PD), the Polymorphic Social Big Data Preprocessor (SBD) approach is used.

As per Equation 1, S is a set of n sources of social media text datasets with different schemas. Each source S has a list of features {F1, F2 … Fm} with different attributes and scale values that must be converted into the same scale using the Min-max or Z-score normalization approach.

The SBD preprocessor produces the PD (Preprocessed Dataset), a merged set of integrated and preprocessed data per Equation 2. Dataset PD is ready for the next step, feature engineering.

$$S = \{S_1, S_2, \ldots, S_n\} \quad (1)$$

$$PD = PD_1 \cup PD_2 \cup \ldots\ldots \cup PD \quad (2)$$

Intermediate steps we have used for pre-processing include removing noise, irrelevant data, and characters; converting all data to a single standard encoding format; removing expressions; splitting attached words; handling missing values; data normalization; data transformation; and data integration.

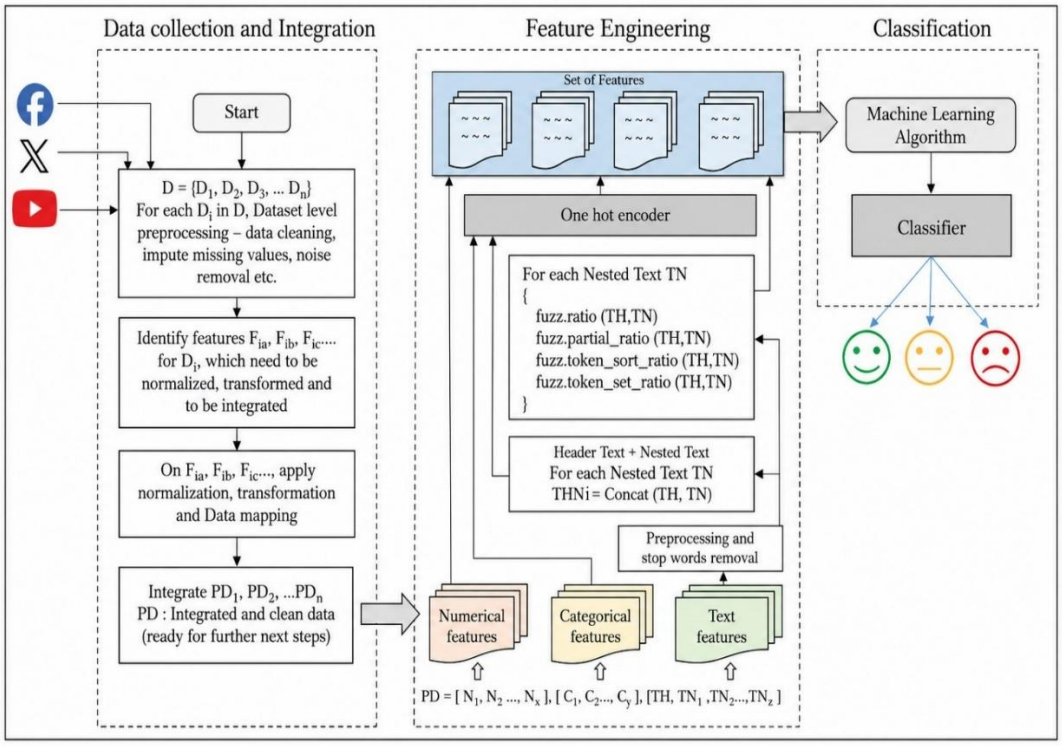

Fig. 2 NeSS-Class framework

4.1.2. Feature Engineering

The input to this step is the preprocessed and integrated dataset (PD), prepared in the previous step. The list of features in PD needs to be converted to a mathematical format before being used in machine learning classification models. PD is divided into list of numerical features [N1,N2,….,Nx], categorical features [C1,C2,…….,Cy] and textual features [TS1,TS2….TSm,,TN1, TN2…..,TNn].

$$PD = [N_1, N_2 \dots, N_x], [C_1, C_2 \dots, C_y],$$

$$[TS_1, TS_2, \dots TS_m, TN_1, TN_2 \dots, TN_n] \quad (3)$$

$$PD = \sum_{p=1}^{x} N_p, \sum_{q=1}^{y} C_q, \sum_{i=1}^{m} \sum_{j=1}^{n} TS_i\ TN_j \quad (4)$$

Where,

PD = Preprocessed integrated dataset

TS = Header or main comment/subject = $\sum_{i=1}^{m} TS_i$

TN = nested text/comment = $\sum_{j=1}^{n} TN_j$

N = set of numerical features = $\sum_{p=1}^{x} N_p$

C = set of categorical features = $\sum_{q=1}^{y} C_q$

The set of numerical features [N] can be directly moved to the final set of features, but a set of categorical features [C] and a set of textual features [TS1, TS2,…, TSm, TN1, TN2,…, TNn] must be converted into numerical vectors using one-hot encoding. In the dataset, the main and respective nested comments are part of the same ongoing discussion, so there may be some string similarities between them.

A set of new features is introduced using string similarity to improve sentiment analysis accuracy. Fuzzy string-matching algorithms are the best approach for generating string-similarity features. Using this, four new features are generated: Simple ratio, Partial ratio, Token Sort ratio, and Token set ratio.

As mentioned, the structure of comments and reviews in social media text data is like a main subject, with its reviews in the form of nested comments. For our sentiment analysis problem, only the nested comment part cannot identify the correct opinion.

For correct evaluation, the subject part should also be considered. So, in addition to string similarity features, we have proposed one more feature: the Concatenation of the subject and nested comment TSN = concat (TS, TN). At this stage of the NeSS-Class framework, the final set of features is prepared, which can be supplied to the next stage, ‘classification’.

Algorithm 1: Pseudo code for NeSS-Class Framework

Input: PDtrain = Training dataset (PDtrain_X) having features Numerical (N), Categorical (C), Text Subject (TS) comment, Text Nested (TN) comments and sentiment class (PDtrain_Y) values positive, negative, neutral.
PDtest = Testing dataset having features Numerical, Categorical, Subject Text, Nested Text, and sentiment class values positive, negative, neutral.
Output: log loss, confusion, precision, and recall matrix.

Method:
Start
step 1. $PDtrain = \sum_{p=1}^{x} N_p \ , \ \sum_{q=1}^{y} C_q \ , \ \sum_{i=1}^{m} \ \ \sum_{j=1}^{n} TS_i \ TN_j$
step 2. for each i = 1 to m
2.1 for each j = 1 to n
2.1.1 pre-processing and stop words removal (TS_i , TN_j)
2.1.2 $(subject + Nested)_{ij}$ ← concat (TS_i , TN_j)
2.1.3 $(simple\_ratio)_{ij}$ ← simple_ratio (TS_i , TN_j)
2.1.4 $(partial\_ratio)_{ij}$ ← partial_ratio (TS_i , TN_j)
2.1.5 $(token\_sort\_ratio)_{ij}$ ← token_sort_ratio (TS_i , TN_j)
2.1.6 $(token\_set\_ratio)_{ij}$ ← token_set_ratio (TS_i , TN_j)
2.2 end loop
step 3. end loop
step 4. Derived_features ← [subject + Nested], [simple _ratio], [partial_ratio], [token_sort_ratio], [token_set_ratio]
step 5. Text_sub_nested_One_hot ← One-hot encoder (subject + Nested)ij
step 6. Categorical_one_hot ← One_hot_encoder ($\sum_{q=1}^{y} C_q$)
step 7. Final_Feature_Vector ← Numerical features($\sum_{p=1}^{x} N_p$), Derived string matching features, Categorical_one_hot, Text_sub_nested_One_hot
step 8. Fit a model, test the model using PDtest, and deploy the final classifier: Logistic Regression ← Final Feature Vector, Hyperparameter tuning.
step 9. Return evaluation metric: Log loss, Confusion, Precision, and Recall matrix.
End

4.1.3. Classification

The final set of features includes numerical features, one-hot encoding of categorical features, one-hot encoding of the concatenated main and nested comments, and four generated features developed using string-matching techniques. With these prepared features, sentiment classification can be performed as positive, negative, or neutral using appropriate machine-learning algorithms. The result discussed in a later section shows that Logistic regression with new features is more efficient than the others.

5. Experiments and Evaluation

5.1. Dataset

The proposed NeSS-Class model is applied to Social media data prepared from Facebook and YouTube text comments [19]. The dataset had 1880 main subjective text, and for each subject, ten nested reviews were taken for our study. So, a total of 18800 reviews covering 1880 main subjective texts.

The dataset covers data across various areas like Advertisements, Indian Government plans, Political discussions, Share marketing comments, Games, etc. As data is gathered from multiple sources, initial cleaning and pre-processing are done, and then data is aggregated to create a single data source for further analysis. 80% of this data is used for training, 10% for cross-validation, and 10% for testing. The data set has features 'Title', 'Keyword', 'number of likes', and 'comments', including nested comments.

5.2. Evaluation Metrics

We have used log loss as an evaluation metric because it is the best metric for binary and multiclass classification models and for comparing them. Cross-entropy loss and logarithmic loss are other names for log loss. The difference between expected probability and actual values determines how well the model performs. A lower log loss indicates better model performance, while higher values indicate inaccurate predictions or classification.

The negative average of the log of the corrected predicted probability for each event is the mathematical definition of log loss.

$$\log loss = -\frac{1}{N}\sum_{i=1}^{N} y_i . \log\big(p(y_i)\big) + (1 - y_i) . \log\,(1 - p(y_i)) \quad (5)$$

Where N is the number of reviews/comments in the dataset, y is the actual class value, and p is the prediction

probability [20]. In addition to log loss, we have generated a Confusion, Precision, and Recall matrix that provides an overall picture of each model's achieved accuracy. These matrices help understand the efficiency of a classification model and summarize the predictions made by an ML model against the actual ground truth values.

5.3. Experimental Setup

To implement the suggested model, Python is used with the nltk, sklearn, and FuzzyWuzzy packages on a machine with an Intel Core i7 processor and 16 gigabytes of RAM. The NLTK package is used in Python to construct statistical Natural Language Processing (NLP) applications that employ human language data. Semantic reasoning, parsing, tokenization, classification, stemming, and tagging are text processing tools in this library. The Scikit-learn library includes powerful techniques for statistical modeling, machine learning, dimensionality reduction, regression, clustering, and classification. The FuzzyWuzzy library in Python calculates how similar two strings are. This uses the Levenshtein distance algorithm, which measures the minimum number of modifications required to transform one string into another. The configuration of many hyperparameter tunings with varying values is shown in Figures 3 to 6, and the optimal value is chosen for the outcome. Using the best hyperparameter values, the NeSS-Class model is tested on a test dataset and produces the best results.

6. Results and Discussion

The proposed NeSS-Class model is trained and evaluated on the Facebook and YouTube datasets, which include both main and nested text data. Its performance is compared with that of state-of-the-art methods on the same dataset. The performance of the proposed framework is also compared across different hyperparameters, and the best hyperparameter is selected based on the minimum log loss.

During the Univariate analysis of "keyword" and "text" features for our experiments, we collected information such as the number of unique words (in the text and keyword fields) in the train data, the distribution of word frequencies, potential methods for preparing useful features for text field, whether the text and keyword features are helpful for sentiment prediction, whether they are stable across train, test, and cross-validation datasets, etc. The training dataset for this study has a total of 10087 unique words, and we calculated each word frequency as well. Additionally, 96.532 percent of the words from the test data and 97.974 percent from the cross-validation data appeared in the train data. It indicates that the text feature is stable across train, test, and cross-validation datasets. For the feature ‘keyword’, we identified which type of feature it is, its importance for predicting the class, and whether it is stable. During the experiment, we got 41 categories of ‘keyword’ features. We have compared two featurization approaches for this feature: One-hot encoding and Response coding. We find that One-hot encoding gives higher accuracy than response coding for social media datasets. To find the stability of the ‘keyword’ feature, we calculated how many data points in the Test and CV datasets are covered by 41 ‘keywords’ in the train dataset. Our finding is that all 41 ‘keywords’ are present in the Test and CV datasets, indicating that the ‘keyword’ feature is stable. Upon request, the implementation code and experimental configurations can be provided through a public repository for research and academic use.

6.1. Logistic Regression with NeSS-Class Framework

Hyperparameter tuning and evaluation using the Confusion, Precision, and Recall matrix as shown in Figures 3 and 4. For the best value of hyperparameter = 1e-05, the log-loss value for the train dataset is 0.3752133257162475, the cross-validation dataset is 0.6518821321179812, and the test dataset is 0.6553180338440054.

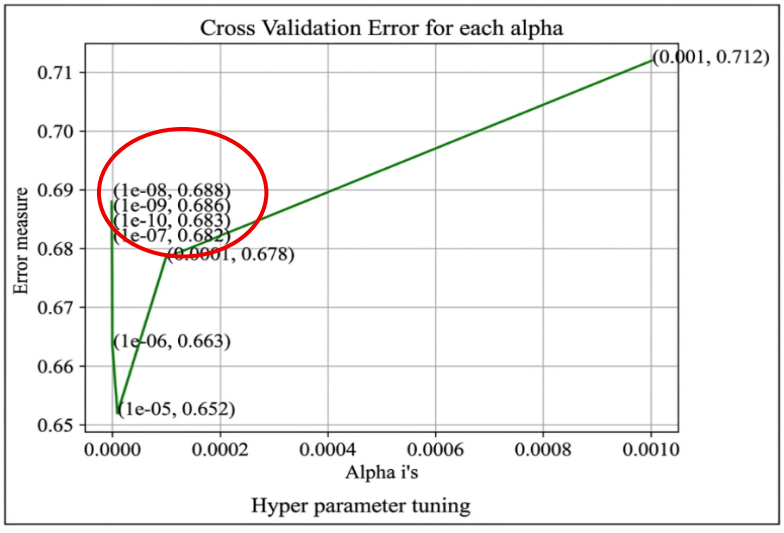

Fig. 3 Hyperparameter tuning - Logistic regression with NeSS-Class framework

6.2. Naive Bayes with NeSS-Class Framework

Hyperparameter tuning as shown in Figure 5, and evaluation using the Confusion, Precision, and Recall matrix as shown in Figure 6. For values of best hyperparameter = 0.1, the log-loss value for the train dataset is 0.6556633124946708, the cross-validation dataset is 0.7524033023721652, and the test dataset is 0.752403302372165.

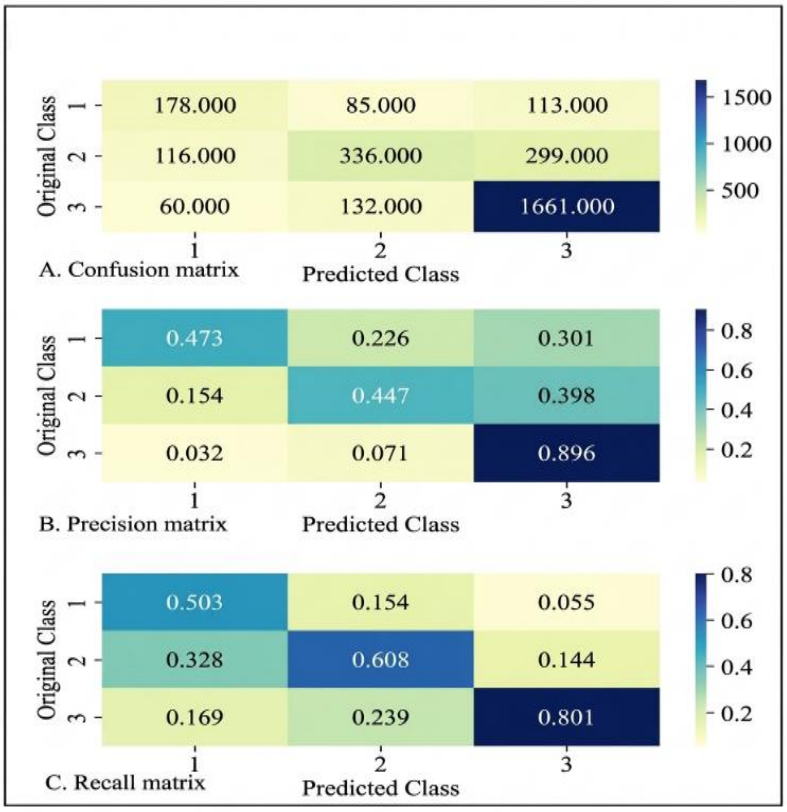

Fig. 4 Confusion Matrix - Logistic regression with NeSS-Class framework

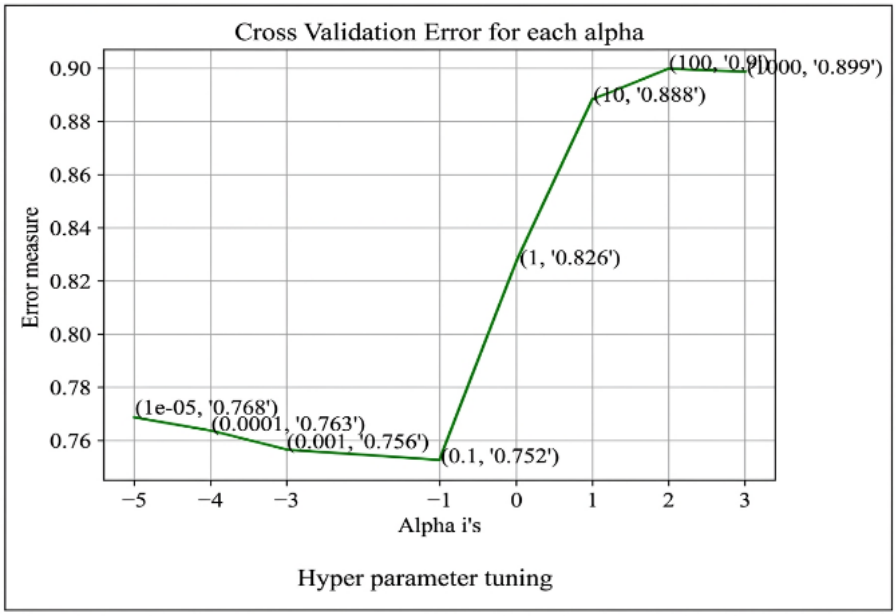

Fig. 5 Hyperparameter tuning - Naive bayes with NeSS-Class framework

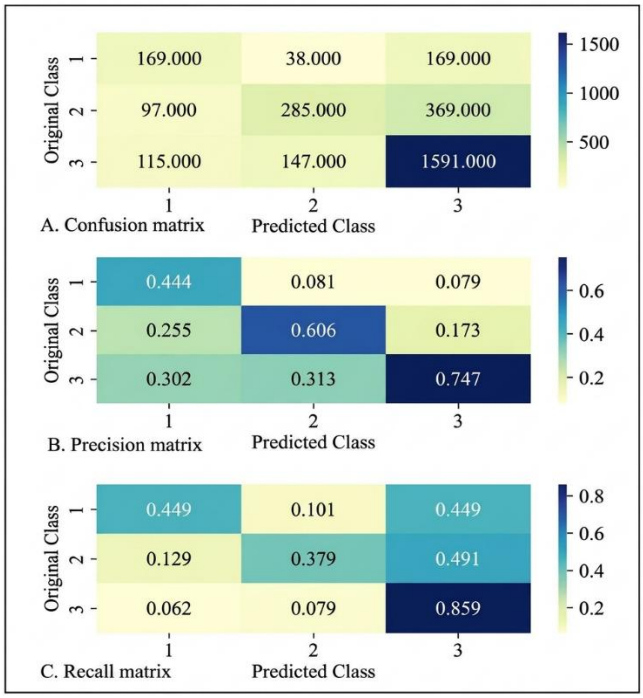

Fig. 6 Confusion Matrix - Naive bayes with NeSS-Class framework

6.3. Linear SVM with NeSS-Class Framework

Hyperparameter tuning as shown in Figure 7, and evaluation using the Confusion, Precision, and Recall matrix as shown in Figure 8. For the best hyperparameter = 1e-06, the log-loss value for the train dataset is 0.8133843042570321, the cross-validation dataset is 0.8075213489859844, and the test dataset is 0.80752134898598.

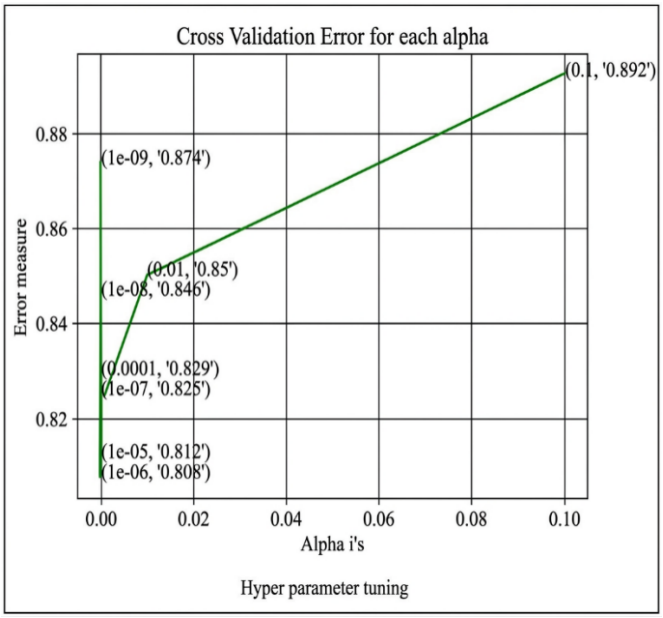

Fig. 7 Hyperparameter tuning - Linear SVM with NeSS-Class framework

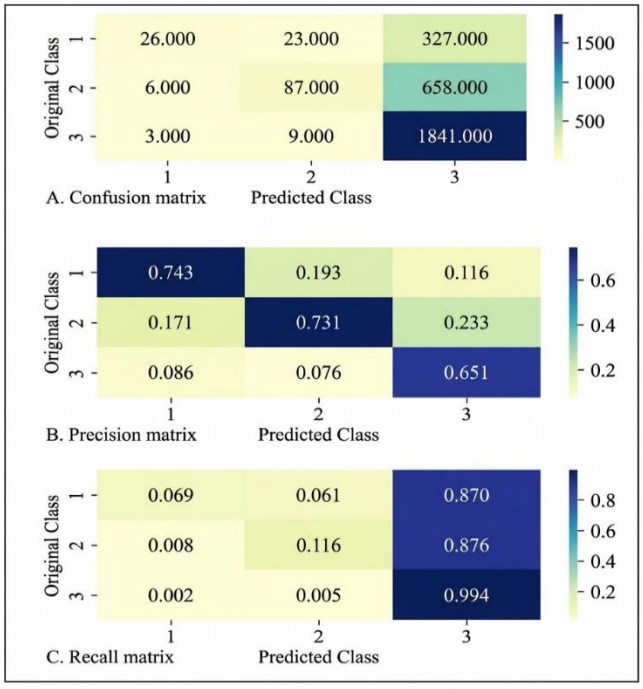

Fig. 8 Confusion matrix - Linear SVM with NeSS-Class framework

6.4. Random Forest with NeSS-Class Framework

Hyperparameter tuning as shown in Figure 9, and evaluation using the Confusion, Precision, and Recall matrix as shown in Figure 10. For the best estimator = 1000 and max depth = 10, the log loss value for the train dataset is 0.6400697934852775, the cross-validation dataset is 0.7108821778615249, and the test dataset is 0.710882177861531.

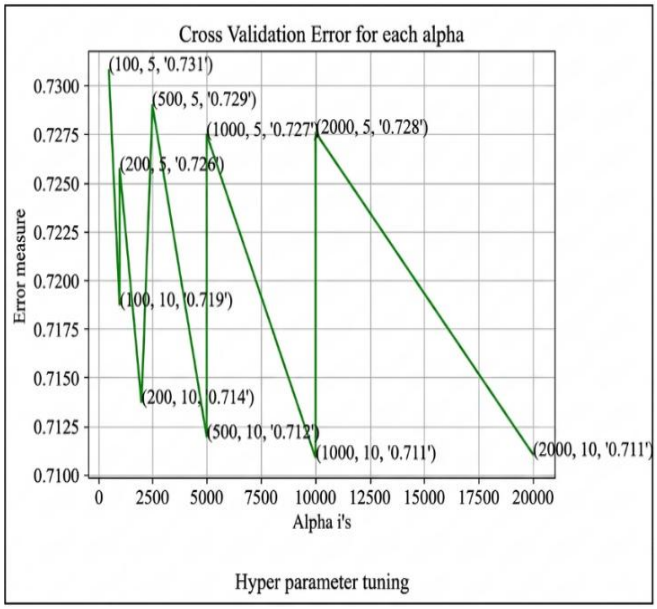

Fig. 9 Hyperparameter tuning - Random forest with NeSS-Class framework

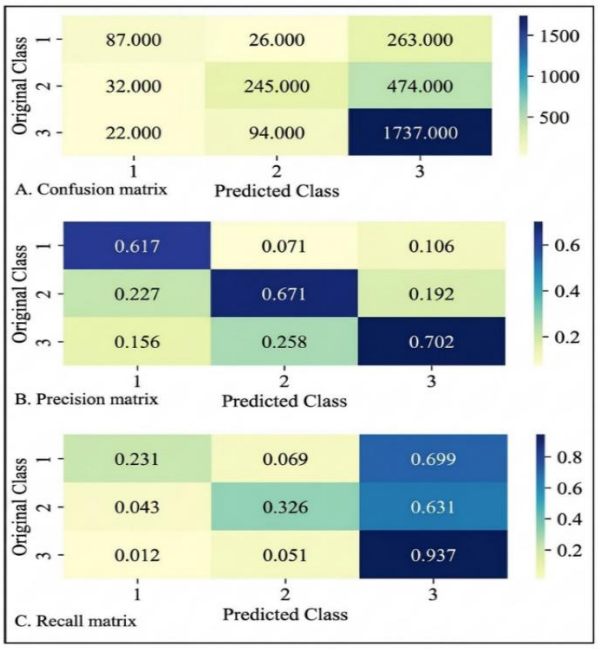

Fig. 10 Confusion matrix – Random Forest with NeSS-Class framework

Table 1 shows the evaluation metric log loss values for the Machine learning models mentioned, with and without the proposed NeSS class framework. Also, Table 2 shows the key features comparison of the proposed framework with the LSTM, BERT, RoBERTa, other traditional and Hybrid models.

6.5. Comparative Study

The performance of the NeSS-Class model is compared with state-of-the-art methods and models. Figures 11 to 13 compare the NeSS-Class model and performance without it in terms of train, cross-validation, and test logloss values. As per the findings, the training log loss is lower than the Cross-Validation (CV) and test log losses.

This indicates that the model is likely overfitting to the training data. Despite regularization and hyperparameter tuning, the training error is lower than the cross-validation and test logloss errors. One obvious solution is to gather more diverse training data, which will help the model to learn more general patterns.

Table 1. Log loss evaluation metric values with respective ML models

Model	NeSS-Class model			Without NeSS-Class model		
	Train log loss	Cross-validation log loss	Test log loss	Train log loss	Cross-validation log loss	Test log loss
Logistic regression	0.3752	0.6518	0.6553	0.4218	0.8991	0.9099
Naive Bayes	0.6556	0.7524	0.7654	0.6502	0.7455	0.7243
Linear SVM	0.8133	0.8075	0.8156	0.8186	0.9228	0.9347
Random Forest (Best estimator value=1000, max depth=10)	0.6400	0.7108	0.7187	0.6621	0.7221	0.7379

Table 2. Comparison of the proposed framework with recent models (BERT, RoBERTa, LSTMs)

Model / Framework	Nested Comment Support	Feature Engineering	Interpretability	Computational Cost	Key Limitation
BiLSTM / LSTM	Limited	Automatic feature learning	Low	High	Requires large training data and longer training time
BERT	Limited	Contextual embeddings	Low	Very High	Computationally expensive and less interpretable
RoBERTa	Limited	Advanced transformer embeddings	Low	Very High	Requires GPU infrastructure and extensive fine-tuning
RoBERTa + LSTM Hybrid	Partial	Deep contextual feature extraction	Low	Very High	High model complexity and difficult deployment
Traditional ML Models (NB, SVM, LR, RF)	No	Manual feature engineering	High	Low	Unable to capture conversational relationships

Proposed NeSS-Class Framework	Yes	Fuzzy string matching + contextual features	High	Moderate	Performance depends on the quality of conversational data

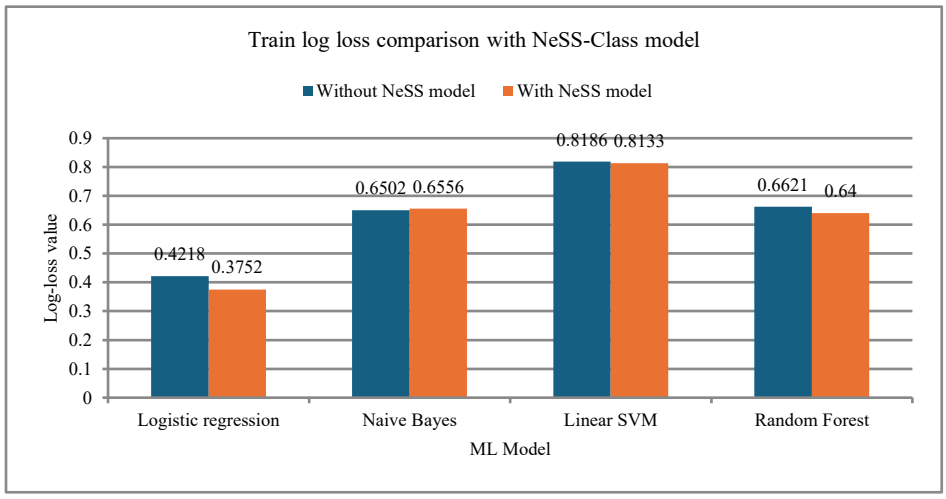

Fig. 11 Train log loss comparison with NeSS-Class model

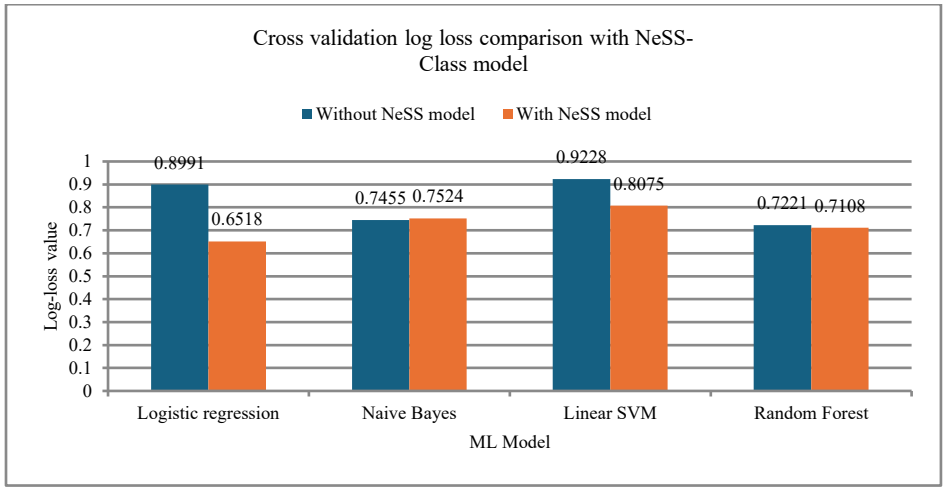

Fig. 12 Cross-validation log loss comparison with NeSS-Class model

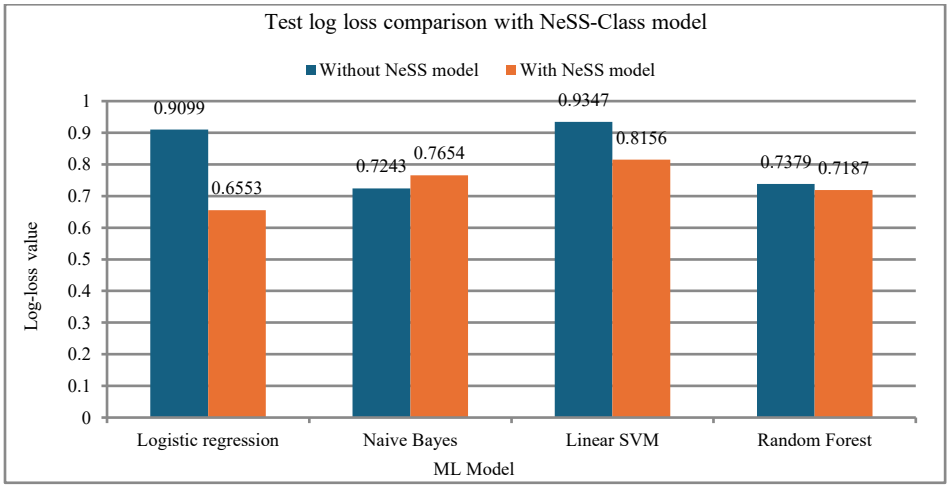

Fig. 13 Test log loss comparison with NeSS-Class model

Further, Figures 14 to 17 show performance comparison of the NeSS-Class model with Logistic regression, Naïve Bayes, Linear SVM, and Random Forest ML models. As per findings, Logistic regression with NeSS performs better.

Also, the Linear SVM with NeSS-Class model has approximately the same values for training, cross-validation, and test cross-validation log loss. Figures 18 and 19 show an overall comparison of the studied baseline ML models with and without the proposed NeSS-Class framework.

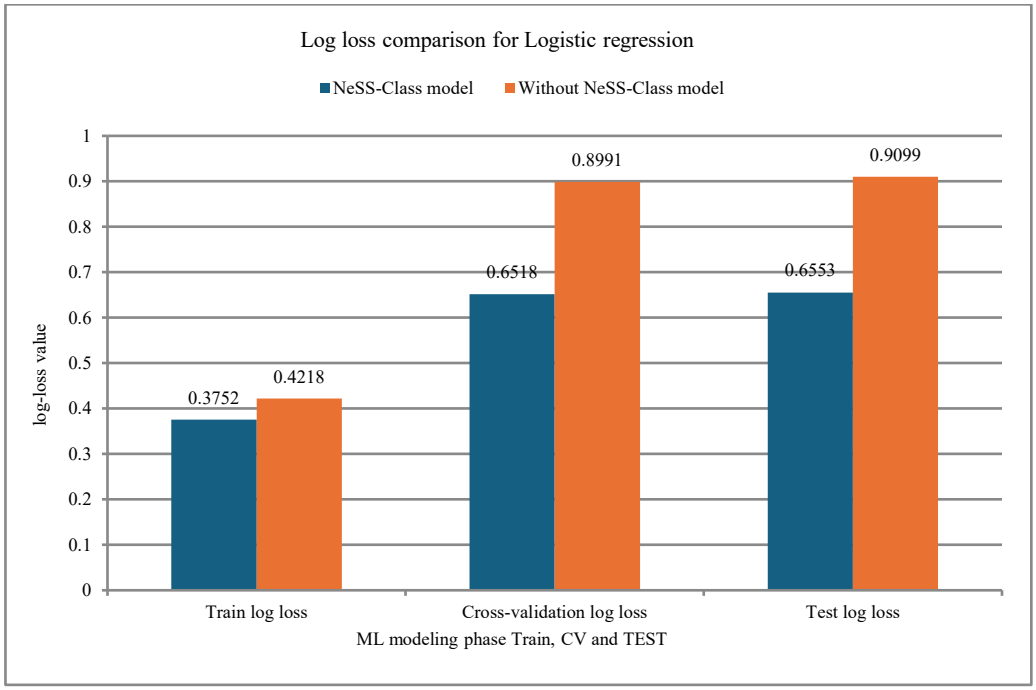

Fig. 14 Logistic regression with NeSS-Class model comparison

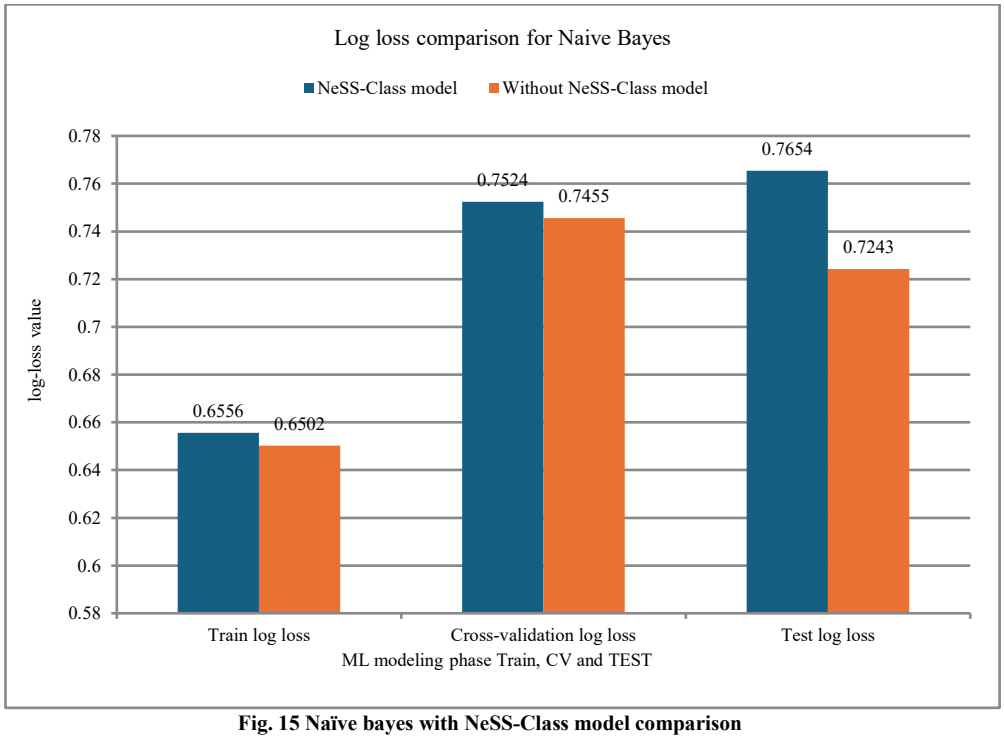

Fig. 15 Naïve bayes with NeSS-Class model comparison

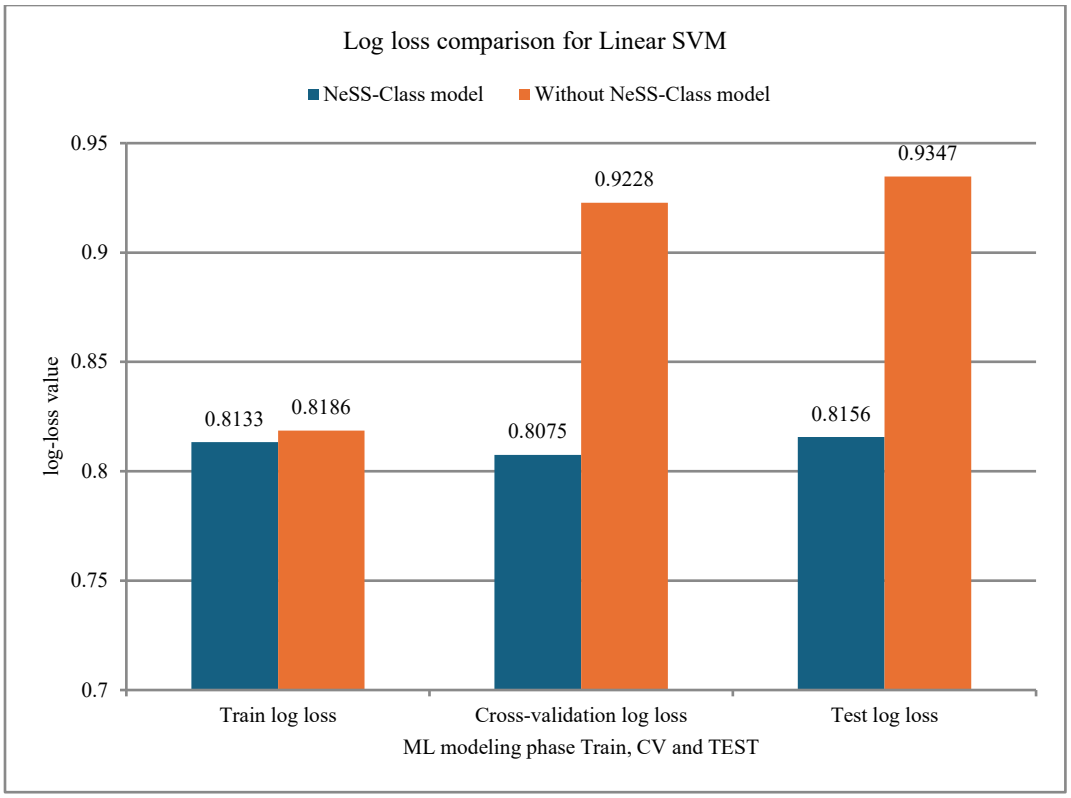

Fig. 16 Linear SVM with NeSS-Class model comparison

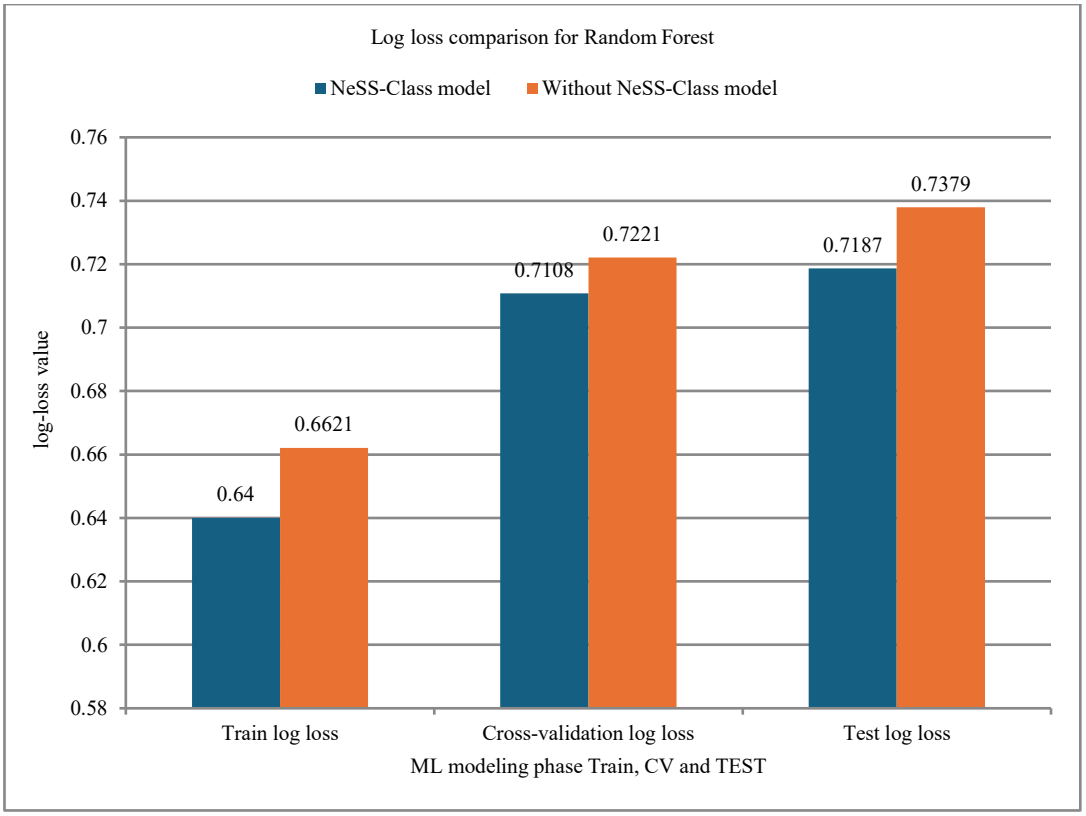

Fig. 17 Random forest with NeSS-Class model comparison

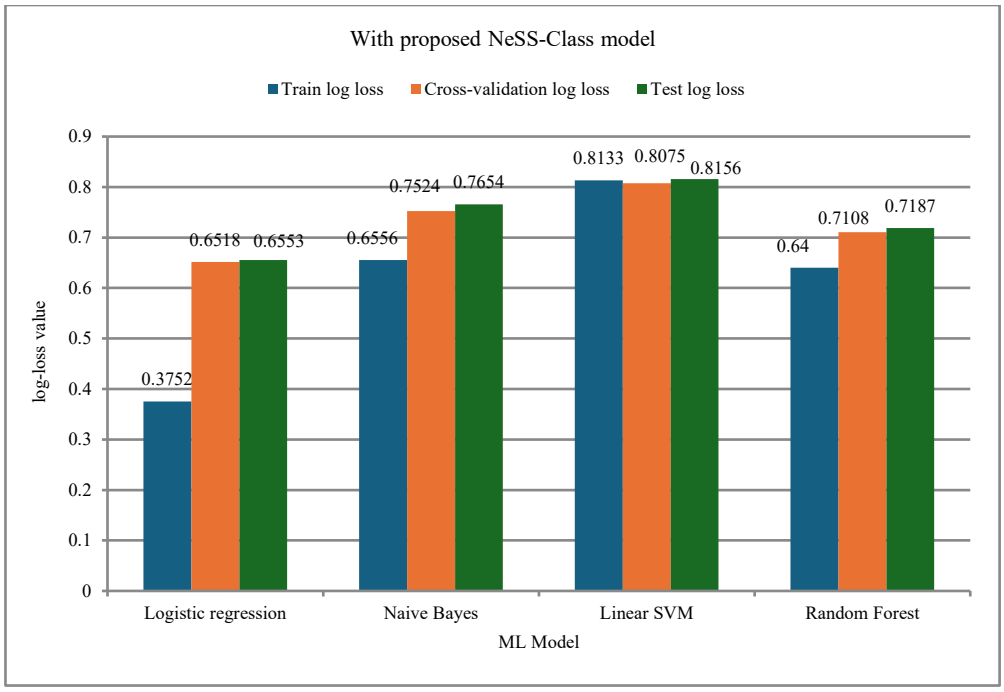

Fig. 18 Performance with NeSS-Class model

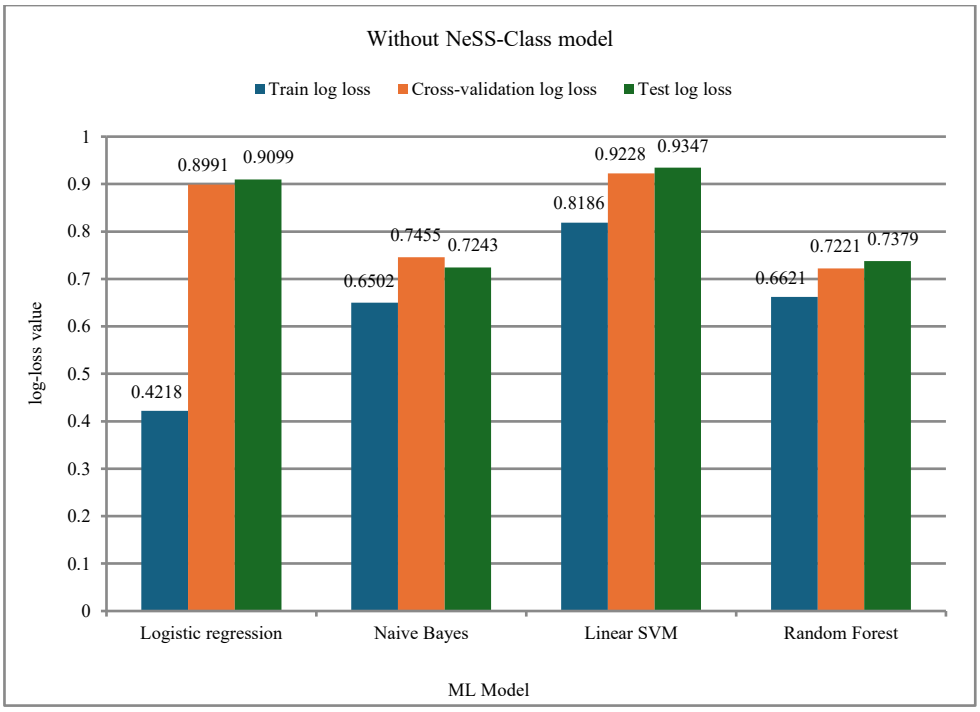

Fig. 19 Performance without NeSS-Class model

Also, we calculated log loss with a random model to determine the improvement in efficiency, and the value is 1.48. As shown in Figure 20, there is a significant improvement in efficiency for Logistic Regression, Linear SVM, and the Random Forest classifier. Out of these, Logistic regression performed the best, with the NeSS-Class framework, as its log loss is the lowest at 0.6553. After Logistic Regression, the second efficient model is the Random Forest model, followed by the Linear SVM model. One important note is that, with the NeSS-Class framework, the efficiency of Logistic Regression, Linear SVM, and Random Forest improves, but with the Naive Bayes model, there is no improvement.

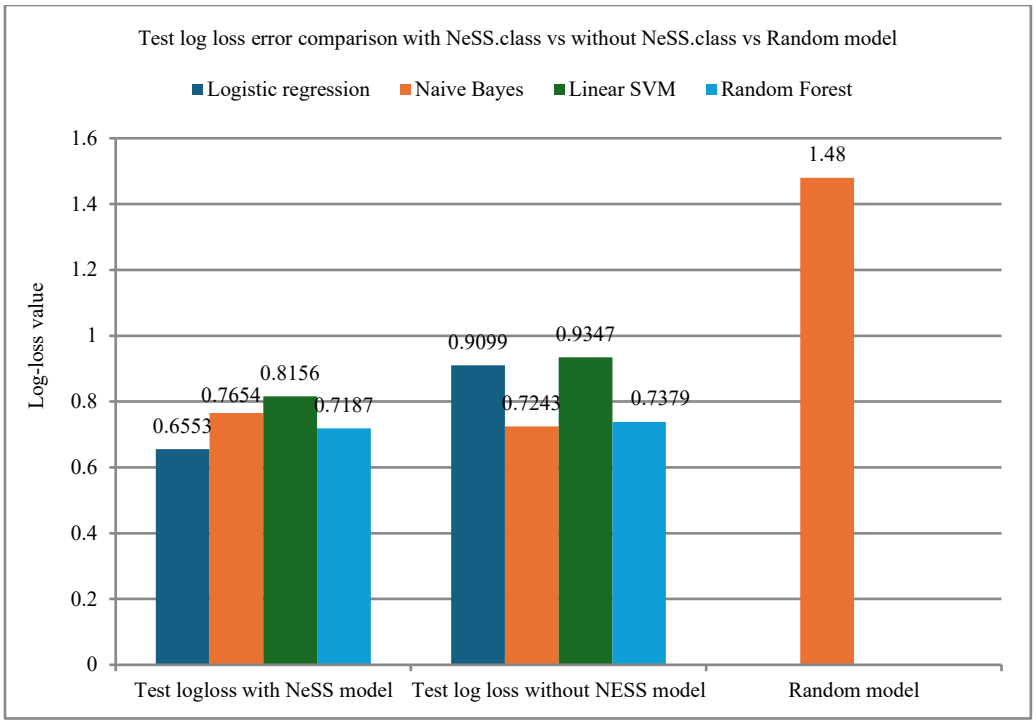

Fig. 20 Performance comparison of NeSS-Class framework with baseline models

7. Conclusion and Future Scope

This study extracts text data from multiple social media platforms and prepares a final source of preprocessed, integrated data, including the main and respective nested comments. After preprocessing, sanity checks and validations are performed, including feature stability and univariate analysis. A set of numerical, categorical, and text features is prepared for the proposed Nested Social Sentiments classification (NeSS-Class) approach.

For main and respective nested comments, five new sets of features are introduced using the Fuzzy string-matching algorithm: Simple ratio, Partial ratio, Token Sort ratio, Token set ratio, and Concatenation of main comment to nested comment. These proposed features are applied with baseline machine learning models, KNN, Naive Bayes, Linear SVM, and Random Forest for sentiment classification. According to the results, the proposed method with Logistic regression is the most efficient, with the lowest log loss of 0.6553; Logistic regression without NeSS-Class yields a log loss of 0.9099, and a random model yields a log loss of 1.48. It has been observed that a new set of features in the proposed NeSS-Class method improves sentiment classification performance, except for the Naive Bayes model. In future research, this work can be extended using the word2vec approach, which also considers semantic relationships. Also, by using various feature engineering approaches within the proposed NeSS-Class framework, a new set of advanced features can be generated to improve efficiency.

Conflicts of Interest

The authors declare that they have no conflict of interest.

Acknowledgments

Amit Kumar Jadiya: original draft of research paper, analysis and implementation. Ramesh Thakur: Conceptualization, Analysis and editing of the paper. Archana Thakur: Review and Conceptualization.

References

[1] Amira Samy Talaat, "Sentiment Analysis Classification System using Hybrid BERT Models," *Journal of Big Data*, vol. 10, no. 1, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] Andreas M. Kaplan, and Michael Haenlein, "Users of the World, Unite! The Challenges and Opportunities of Social Media," *Business Horizons*, vol. 53, no. 1, pp. 59-68, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[3] Gianfranco Lombardo et al., "A Combined Approach for the Analysis of Support Groups on Facebook-the Case of Patients of Hidradenitis Suppurativa," *Multimedia Tools and Applications*, vol. 78, no. 3, pp. 3321-3339, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[4] Jingfeng Cui et al., "Survey on Sentiment Analysis: Evolution of Research Methods and Topics," *Artificial Intelligence Review*, vol. 56, no. 8, pp. 8469-8510, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Qianwen Ariel Xu, Victor Chang, and Chrisina Jayne, "A Systematic Review of Social Media-based Sentiment Analysis: Emerging Trends and Challenges," *Decision Analytics Journal*, vol. 3, pp. 1-16, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[6] J. Fernando Sánchez-Rada, and Carlos A. Iglesias, "Social Context in Sentiment Analysis: Formal Definition, Overview of Current Trends and Framework for Comparison," *Information Fusion*, vol. 52, pp. 344-356, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[7] Vankayala Tejaswini, Korra Sathya Babu, and Bibhudatta Sahoo, "Depression Detection from Social Media Text Analysis using Natural Language Processing Techniques and Hybrid Deep Learning Model," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 23, no. 1, pp. 1-20, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[8] Gagandeep Kaur, and Amit Sharma, "A Deep Learning-based Model using Hybrid Feature Extraction Approach for Consumer Sentiment Analysis," *Journal of Big Data*, vol. 10, no. 1, pp. 1-23, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[9] Rohitash Chandra, and Aswin Krishna, "COVID-19 Sentiment Analysis Via Deep Learning During the Rise of Novel Cases," *PLOS One*, vol. 16, no. 8, pp. 1-26, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] Ishaani Priyadarshini, and Chase Cotton, "A Novel LSTM-CNN-Grid Search-based Deep Neural Network for Sentiment Analysis," *The Journal of Supercomputing*, vol. 77, no. 12, pp. 13911-13932, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[11] Marcin Pietras, "Sentence Sentiment Classification using Fuzzy Word Matching Combined with Fuzzy Sentiment Classifier," *Electrical Engineering Review*, vol. 1, no. 2, pp. 109-113, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[12] Margarita Rodríguez-Ibánez et al., "A Review on Sentiment Analysis from Social Media Platforms," *Expert Systems with Applications*, vol. 223, pp. 1-14, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] M. Harikrishnan, Understanding Fuzziness: Exploring the FuzzyWuzzy Algorithm, Medium, 2023. [Online]. Available: https://medium.com/@harikrishnanhari.india/understanding-fuzziness-exploring-the-fuzzywuzzy-algorithm-7e0b4b05f3d7

[14] Krishna Prakash Kalyanathaya, D. Akila, and G. Suseendren, "A Fuzzy Approach to Approximate String Matching for Text Retrieval in NLP," *Journal of Computational Information Systems*, vol. 15, no. 3, pp. 26-32, 2019. [Google Scholar]

[15] Mayur Wankhade, Annavarapu Chandra Sekhara Rao, and Chaitanya Kulkarni, "A Survey on Sentiment Analysis Methods, Applications, and Challenges," *Artificial Intelligence Review*, vol. 55, no. 7, pp. 5731-5780, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Jundong Li et al., "Feature Selection: A Data Perspective," *ACM Computing Surveys*, vol. 50, no. 6, pp. 1-45, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[17] Mucan Liu, Chonghui Guo, and Lianghen Xu, "An Interpretable Automated Feature Engineering Framework for Improving Logistic Regression," *Applied Soft Computing*, vol. 153, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[18] Tara Rawat, and Vineeta Khemchandani, "Feature Engineering (FE) Tools and Techniques for Better Classification Performance," *International Journal of Innovations in Engineering and Technology*, vol. 8, no. 2, pp. 169-179, 2017. [CrossRef] [Google Scholar]

[19] Advay Patil, Youtube Statistics, Kaggle, 2022. [Online]. Available: https://www.kaggle.com/datasets/advaypatil/youtube-statistics

[20] Marina D. Ibrishimova, and Kin F. Li, "Discerning Cyber Threatening Incidents from Ordinary Events using Sentiment Analysis and Logistic Regression," *Security and Privacy*, vol. 6, no. 4, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[21] Yue Teng, and Kai Yang, "Research on Social Recommendation Algorithm based on PSO_KFCM Clustering and CBAM Attention Mechanism of Graph Neural Networks," *IAENG International Journal of Computer Science*, vol. 51, no. 8, pp. 936-948, 2024. [Google Scholar]