Brajesh Kumar SHRIVASH, PhD Candidate (corresponding author)
brajesh.kumar.shrivash@gmail.com
Jaypee University of Engineering & Technology, Guna, India

Dinesh Kumar VERMA, PhD
dinesh.verma@juet.ac.in
Jaypee University of Engineering & Technology, Guna, India

Prateek PANDEY, PhD
prateek.pandey@juet.ac.in
Jaypee University of Engineering & Technology, Guna, India

A Novel Framework for Text Preprocessing using NLP Approaches and Classification using Random Forest Grid Search Technique for Sentiment Analysis

Abstract. *Text preprocessing is a process to organize and prepare raw text for machine learning (ML) and deep learning (DL) models. This process is most widely used in Sentiment Analysis (SA). To process raw text data, Natural Language Processing (NLP) plays a vital role in data preprocessing by cleaning text, removing punctuations and stopwords, stemming and lemmatization. The classification of text data is a common use of NLP. An effective data preprocessing approach also identifies text features efficiently. The most essential component of text classification is feature extraction from raw text data to input into ML and DL models. Feature extraction prepares training data sets effectively for ML and DL models to find good results. In this paper, we proposed a novel framework for text preprocessing using NLP approaches, and feature extraction, and expanded it for the Random Forest ML classifier. To improve the performance of the random forest classifier, we examined the grid search technique with estimator Random Forest Classifier and measured the performance of the model. The accuracy measure for the random forest model was 93%, while the accuracy measure using the grid search technique was 94%, which shows that the grid search technique enhances the model performance.*

Keywords: *Sentiment Analysis, Natural Language Processing, data preprocessing, classification, machine learning.*

JEL Classification: C02, C81, D83.

Received: 8 March 2025	Revised: 24 May 2025	Accepted: 12 June 2025

1. Introduction

Sentiment analysis, also known as opinion mining, is crucial for assessing public opinion in different domains like business, healthcare, and social media. In modern era, people share their opinions online through social media, shopping sites,

DOI: 10.24818/18423264/59.2.25.06

and reviews; it is important for businesses, policymakers, and researchers to understand how people feel about their products and services.

With the exponential growth of textual data, machine learning (ML) approaches have become critical for automated sentiment classification. However, the accuracy and efficiency of these models rely heavily on adequate text pre-processing. Text processing is a subset of NLP, which aims to build an intelligent system to do sentiment analysis by analyzing opinions to automatically extract and classify emotions towards an entity like happy/positive, unhappy/negative, and neutral. It is important to eliminate irrelevant words, letters, or punctuation, as they increase computation time and add no value to text processing.

This paper introduces a novel framework using NLP techniques to pre-process text and enhances sentiment categorization accuracy. The framework incorporates tokenization, stopword removal, lemmatization, and word embeddings to enhance text classification. Additionally, to achieve high classification accuracy, this approach integrates the Random Forest classifier with Grid Search optimization, a powerful ensemble learning technique that selects the best hyperparameters for improved model performance. By combining robust preprocessing techniques with an optimized machine learning model, the proposed framework aims to enhance sentiment classification accuracy, reduce computational complexity, and improve generalization across different datasets. This study also attempts to improve sentiment analysis accuracy and scalability for disaster sentiment datasets by combining sophisticated text preparation and an efficient classification technique.

Overall, a well-structured preprocessing framework improves domain adaptability and is compatible with both traditional ML models and advanced deep learning techniques. An optimized preprocessing method results in more reliable sentiment categorization by lowering computational complexity and enhancing model performance, including accuracy, precision, and recall. Using NLP-based preprocessing not only improves input data but also increases the effectiveness of sentiment analysis models in real-world applications.

2. Literature review

SA is an essential application of NLP, aimed at determining the sentiment polarity of text. Various studies have explored different methods to enhance sentiment classification performance.

Ahuja et al. (2019) studied different text preprocessing techniques for feature extraction and analyzed their performance to improve model performance by.

Fouad et al. (2018) explored feature extraction methods like Bag-of-Words, lexicons, and part-of-speech tagging, employing classifiers such as Naïve Bayes, Logistic Regression, and ensemble learning. Other traditional sentiment analysis methods that do not rely on emotional words but instead compute sentiment based on sentence structure were discussed by Li, J., & Qiu, L. (2017).

Kamale et al. (2015) examined classification methods for feature-sentiment analysis and found that Naïve Bayes and SVM algorithms improve sentiment

classification. Zhang et al. (2018) evaluated deep learning algorithms for SA, focusing on RNNs and LSTMs, and found that they outperformed standard approaches. Morente-Molinera et al. (2019) used SA in group decision-making on social networks, whereas Al-Smadi et al. (2019) improved aspect-based SA for Arabic hotel reviews by incorporating syntactic and semantic data. Sajadi et al. (2018) developed a fuzzy supervisor controller to optimize DC machine drivers in robotics.

The impact of domain-specific text analysis was examined by Spatiotis et al. (2017) studies Greek-language feedback from e-learning platforms and analyzed using part-of-speech-based feature extraction. A comprehensive study by Prusa et al. (2015) also examined ten feature selection strategies across four classifiers, concluding that feature selection enhances sentiment classification accuracy. Twitter sentiment analysis has been explored in multiple studies. Nazare et al. (2018) did a study on a dataset including 1,000 tweets which were classified using ensemble techniques and categorizing them as positive, negative, or neutral. Agarwal et al. (2011) focused on Twitter data preprocessing by removing stop words, URLs, and symbols, showing that correcting slang and spelling errors significantly improves classification accuracy with SVM models.

Hussein (2018) explored sentiment classification techniques and their performance measures at multiple levels document, sentence, word, and aspects. Alaei et al. (2019) highlighted the challenge of categorizing vast and unstructured social media data of tourism sentiments. Maximum Entropy Probabilistic Latent Semantic Analysis models were used by Xie et al. (2019) and Alam & Yao (2019) to separate sentiment words, demonstrating effective preprocessing techniques to machine learning models. The significance of GloVe-DCNN model was studied by Jianqiang et al. (2018) and Sankar et al (2019) integrated word embeddings, n-grams, and polarity scores, effectively improving sentiment classification.

Various hybrid methods have been explored to enhance the performance of models for SA, combining different techniques and methodologies to improve accuracy, address the limitations of individual models, and refine the overall performance of sentiment detection in diverse contexts. A fuzzy-based method was proposed by Dragoni & Petrucci (2018) to compute text polarity across multiple domains. Pham & Le (2018) introduced a novel approach for sentiment classification using word embedding and compositional vector models to analyze customer reviews.

Tu et al. (2012) developed a sentiment analysis model using deep learning techniques. Nguyen et al. (2014) and Bahaghighat et al. (2019) implemented a two-stage sentiment classification model which includes combined Naïve Bayes and SVM classification by processing simpler texts with NB and complex cases with SVM. Nguyen et al. (2014) incorporated unigram, bigram, and n-gram-based rating features for document-level sentiment analysis to further enhancing classification accuracy.

These studies highlight that integration of DL, NLP, and ensemble techniques demonstrates the growing sophistication of sentiment analysis methodologies, leading to more reliable and domain-adaptive classification models.

3. Research Methodology

The figure 1 shows the proposed process flow of the framework for text preprocessing and feature extraction using NLP. First, we imported an earthquake dataset (disaster sentiments) which was extracted from Kaggle. Then, data visualization techniques were applied, followed by data preprocessing. Finally, text features were extracted using the TF-IDF Vectorizer method.

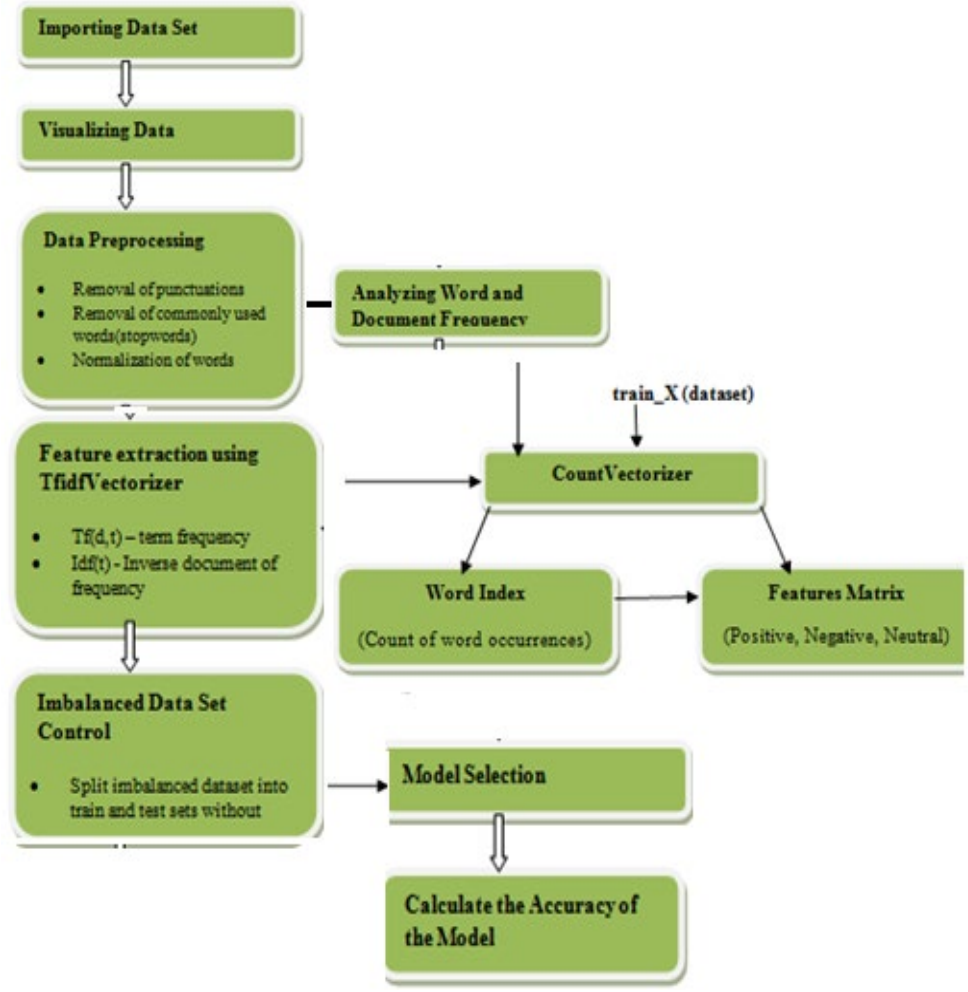

Figure 1. Proposed framework for text preprocessing
Source: Authors' own creation.

To address potential biases caused by imbalanced datasets, the methodology employs a simplified technique to balance the data without stratification, making it suitable for robust model training. The parameters are then fine-tuned using a Random classification technique paired with Grid Search optimization, resulting in optimal model performance. The final phase entails a detailed evaluation of the model's accuracy, which measures the value of the implemented model's predictive capabilities in the context of disaster (earthquake data set) sentiments. This systematic approach emphasizes the significance of diligent data processing and sentiment classification.

The following section describes the different components used in this method.

3.1 Dataset

The earthquake dataset was used for this study. The data was extracted from the Kaggle data source site. It contains a total of 7613 records, of which 3271 were negative tweets, and 4342 were positive tweets.

3.2 Pre-processing techniques

Data pre-processing techniques can be employed to eliminate irrelevant or unusual text from the dataset that lacks sentimental value. These techniques not only reduce the overall data size but also significantly enhance training efficiency by streamlining the input, ultimately leading to faster model convergence and improved performance. In this paper, the following pre-processing techniques were used.

3.2.1 Tokenization

To split up a sentence into smaller meaningful units is known as tokenization. For example –

I love fruits

I	love	fruits	} Tokens

3.2.2 Normalization

Text normalization is an overlooked preprocessing phase. This is a process of transforming the text into a standard form. To achieve normalization, several tasks are carried out at the same time. It includes converting text into lower to upper or upper to lower case, removing punctuation, and converting numerals to their equivalent words. The following table shows a group of words and their normalization during the pre-processing process.

Table 1. Text normalization

S. No	Raw	Normalized
1	gooood	good
	gud	
2	tomrw	tomorrow

S. No	Raw	Normalized
	2moro	
	tomorow	
	2mrrw	

Source: authors' processing.

This pre-processing step increases the uniformity of each text.

i. **Stopwords:** Stop Words are words that have no sentimental/ semantic impacts or the emotions of the sentence on analysis in a sentence, like "is", "am", "are", "of", "the", "in", "english", etc. So, we can ignore these.
ii. **Non-alphanumeric & non-printable characters:** These characters were removed and replaced using a predefine replace() function.
iii. **Email address:** Email addresses were removed and replaced using a predefine .replace() function.
iv. **Numeric characters:** These characters were removed and replaced using a predefined .replace() function.
v. **URLs:** URLs are the web links that start with 'https://….' and have null sentimental significance. These are all removed and replaced using a predefine .replace () function.
vi. **Converts all characters to lowercase:** This step gives uniformity to the text before analyzing it i.e. convert all text into lowercase. This also aids in the reduction of vocabulary size and the elimination of word repetition. Here, we used the lower () method to translate text into lowercase.
vii. **Multiple whitespaces to one:** Multiple whitespaces have been converted into a single one space using .str.strip() function.

3.2.3 Lemmatization

The process of lemmatization involves identifying a word's lemma from its meaning. This process combines multiple words into a single word by analyzing their morphology and removing word endings, such as transforming "tasting" to "delicious" or "cud" to "could."

3.2.4 Noise removal

Noise removal is one of the most essential text preprocessing steps. We need to clean the raw data and remove any noises using regular expressions, which is a technique of NLP. When noise is removed from a dataset, letters, numerals, and fragments of text along with a small number of rows are eliminated. This results in a reduction in the dataset's accuracy.

3.3 Feature extraction

Feature extraction involves converting features into vector representations for machine comprehension. These vectors are then used as input for classifier models. Named Entity Recognition (NER), Bag-of-Words (BoW), and TF-IDF techniques were applied in this study.

3.3.1 TF-IDF

The term frequency-inverse document frequency, also known as TF-IDF, is a popular method for determining the relevance of a word within a text. TF-IDF assesses a word's relevance by taking into account both its frequency in a specific document and its rarity over a group of documents, making it an important method for detecting meaningful terms in text analysis.

- Term Frequency: This is a scoring system that determines how often a certain term appears in the current text.
- Inverse Document Frequency: This is a scoring system that determines how uncommon the word is in all of the documents.

The frequency of a term, denoted as t, is calculated by comparing its occurrences in the text to the total word count. IDF measures the significance of a term across multiple documents. There are several words and phrases, such as "is," "an," "and," etc., that are used often yet do not have any significant meaning.

TF-IDF can effectively transform textual information into a vector space model. For example, if a document contains 200 words and the term "earthquake" appears 10 times, its term frequency (TF) is 0.05. In a corpus of 50,000 documents, if only 500 include the word "earthquake," its inverse document frequency (IDF) helps adjust the weight, making the term's significance more relevant. Then the IDF (earthquake) would be 50000/500=100, and the TF-IDF (earthquake) would be 0.04*100= 4.

3.4 Performance parameters used to measure the performance

The following performance parameters were used to measure the performance of the proposed framework.

Accuracy: Accuracy measures the proportion of correctly classified instances (true positives and true negatives) out of all instances, considering false positives and false negatives.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}} \quad (1)$$

Precision: "Precision" refers to the ratio of correctly predicted positive observations to the total predicted positives. It can be computed as follows:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \quad (2)$$

Recall: Recall is a ratio of correctly predicted positive observations to actual positive observations, and can be calculated as:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \quad (3)$$

F-score: The term "f-score" refers to the weighted average of recall and accuracy.

$$\text{F} - \text{score} = \frac{2*(\text{Precision}*\text{Recall}}{\text{Precision}*\text{Recall}} \quad (4)$$

When there is an unequal distribution in the data, the importance of this parameter over accuracy increases. It can be computed by F-score formula.

4. Results and discussion

This section describes the results of the proposed framework.

4.1 Data set control

A dataset control ensures the quality, consistency, and representativeness of the data used for training. Proper dataset control is crucial for improving model performance, avoiding overfitting, and ensuring generalizability to real-world scenarios.

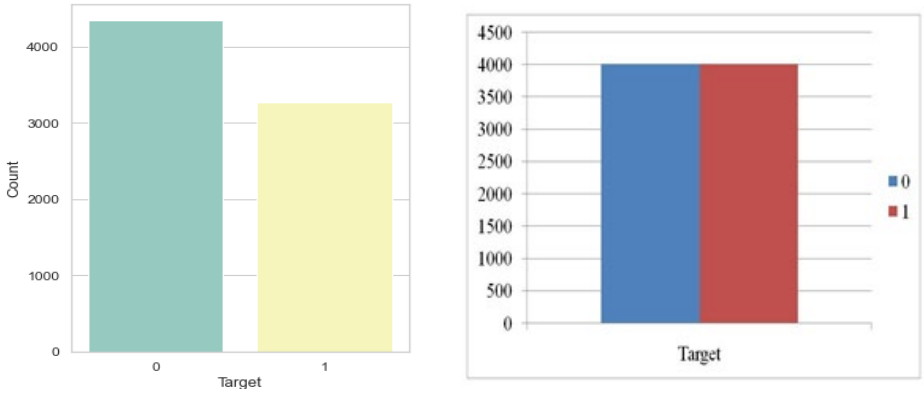

Figure 2.
(a) Imbalanced data set **(b) Balanced data set**
Source: Authors' own creation.

Figure 2(a) tells us that the given data was an imbalanced one with less amount of "1". To make the date set balance, an over-sampling classification model was used. To do this, the SMOTE method in the learning package in Python was implemented to make the data set balance (Figure 2(b)).

4.2 Performance evaluation of various ML models with data split

This section outlines the performance evaluation of various ML models. After balancing the dataset, we assessed the models' performance by dividing the data into different training and test splits: 70:30, 80:20, and 90:10. These splits were used to determine how well each model generalizes to unseen data under different conditions. After evaluating the performance, the proposed Random Forest Grid Search Technique was applied to enhance the performance of the outperform model.

Table 2. ML models performance evaluation with 70:30 data split (in %)

Models	Accuracy	Precision	Recall	F-1 Score
Naïve Bayes	72.46	74.57	69.21	71.73
MLP Classifier	78.1	75.83	72.34	74.05
Passive Aggressive Classifier	76.54	77.01	74.87	75.93
SVM Algorithm	79.03	86.18	60.04	70.77
Logistic Algorithm	78.22	84.35	61.45	71.85
Random Forest Classifier	81.45	80.12	75.22	77.56
Proposed Model	**82.24**	**81.02**	**75.96**	**78.32**

Source: ML and proposed model results, findings by the author.

Table 2 presents a performance evaluation of various ML models, using a 70:30 training-test split. The Random Forest ensemble learning algorithm is the top performer, achieving 81.45% accuracy with a strong balance between precision and recall, effectively identifying positives and minimizing false positives. Naïve Bayes has the lowest scores across all metrics, while SVM and Logistic Regression show strong precision but have lower recall rates. Overall, Random Forest appears to be the most effective model in this comparison.

Proposed Model - After applying the parameter tuning on the Random Forest Classifier, the proposed "Random Forest Grid Search Technique" outperforms all others in terms of accuracy, precision, recall, and F1 score.

In Random Forest-GridSearch technique, we have applied estimator=RandomForestClassifier(), param_grid='max_depth': [2, 4, 5, 6, 7, 8, 9, 10, 12, 16], 'n_estimators': [10, 15, 18, 50, 100], and 'random_state': [3, 42, 777]. After getting the results, we have found that the best parameters for improving performance measures in GridSearch were 'max_depth': 6, 'n_estimators': 15 and 'random_state': 777.

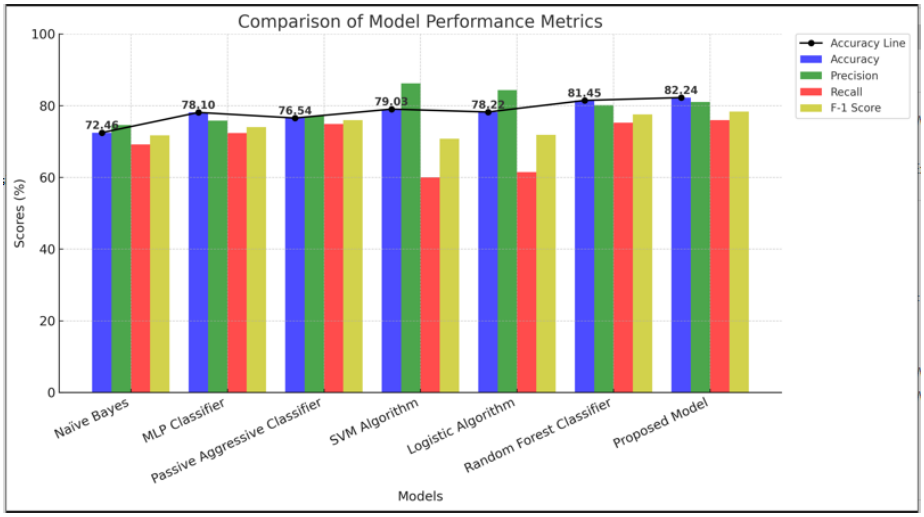

Figure 3. Comparison of model performance with a 70:30 data split
Source: Illustration by authors.

Figure 3 provides a graphical representation of various ML models using key evaluation metrics to understand easily. Among these models, the Random Forest ensemble learning algorithm appears to be the most balanced and effective model, achieving consistently high scores across all metrics, particularly excelling in accuracy. Naïve Bayes consistently shows the lowest performance among the models. While models like SVM and Logistic Algorithm have high precision, their recall is comparatively lower, affecting their F1 Score.

Overall, it highlights the differences in model performance with the proposed Grid Search technique achieving slightly higher accuracy and performing better in recall and F1 score. This suggests that the Grid Search technique improves the model's ability to identify relevant patterns in the data.

Table 3. ML models performance evaluation with 80:20 data split (in %)

Models	Accuracy	Precision	Recall	F-1 Score
Naïve Bayes	81.35	84.13	69.72	76.25
MLP Classifier	78.86	76.35	73.55	74.92
Passive Aggressive Classifier	73.34	68.67	69.72	69.2
SVM Algorithm	79.03	86.18	60.04	70.77
Logistic Algorithm	79.33	84.79	62.32	71.84
Random Forest Classifier	79.97	79.93	71.25	75.34
Proposed Model	**81.02**	**81.8**	**71.96**	**76.26**

Source: ML and proposed model results, findings by the author.

Table 3 shows the performance measures using the 80:20 training-test split of data sets for different ML models. Naïve Bayes performs slightly better, while Random Forest consistently performed well specifically with Recall, and F-1 Score. MLP Classifier shows a good balance of recall, while models like Passive Aggressive Classifier and SVM have lower Recall and F1-scores. Logistic Regression has high precision but a slightly lower recall value.

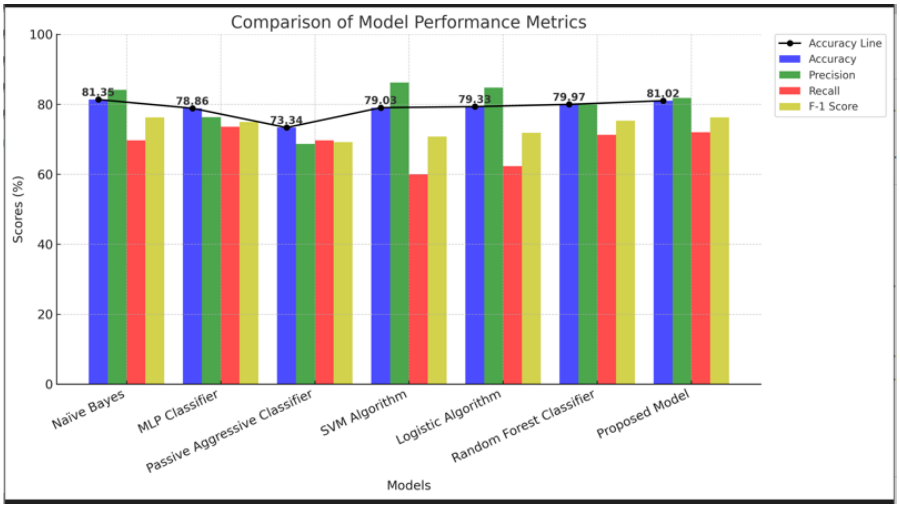

Figure 4. Comparison of model performance with a 80:20 data split
Source: Illustration by authors.

Figure 4 shows the graphical representation of performance for various machine learning models. The Random Forest model demonstrates balanced performance across all metrics compared to other models. Overall, the figure illustrates that the proposed Grid Search technique consistently outperforms the others. This comparison highlights the impact of hyperparameter tuning on model performance.

Table 4. ML models performance evaluation with 90:10 split (in %)

Models	Accuracy	Precision	Recall	F-1 Score
Naïve Bayes	85.42	84.37	78.92	81.53
MLP Classifier	89.12	87.34	84.25	85.78
Passive Aggressive Classifier	88.56	86.6	85.15	85.87
SVM Algorithm	91.05	90.42	88.3	89.34
Logistic Algorithm	90.56	89.7	87.9	88.79
Random Forest Classifier	93	91.2	95	93
Proposed Model	**94**	**91**	**97**	**94**

Source: ML and proposed model results, findings by the author.

Table 4 presents performance measures of 90:10 training and test data for different ML models with a 90:10 training-test split. The Random Forest Classifier delivers the highest accuracy with 93% and performs well, including precision and recall. Naïve Bayes shows the lowest scores, but still performs well compared to other models. SVM and Logistic Regression also perform strongly with high precision and recall. Random Forest again is the best-performing model as compared to other models closely followed by SVM and Logistic algorithms, while the performance of the proposed model stands out as the best-performing model.

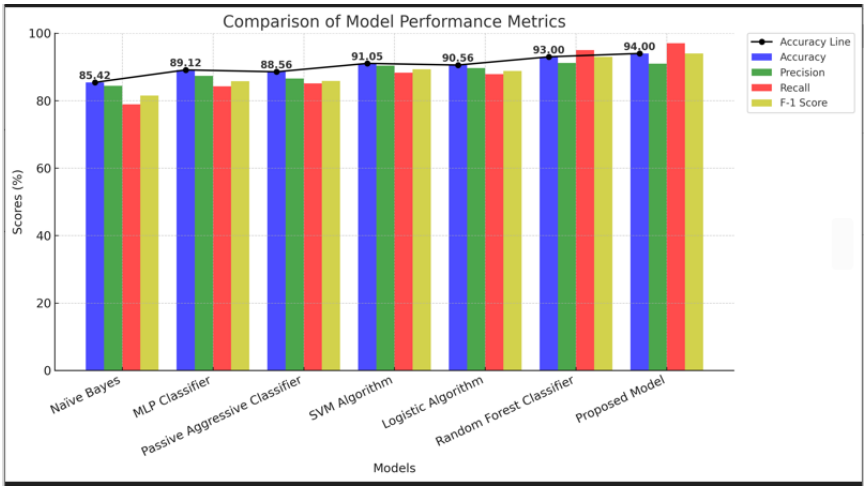

Figure 5. Comparison of model performance with a 90:10 data split
Source: Illustration by authors.

Figure 5 compares the performance of various machine learning models with a 90:10 training-test split in a graphical way. Random Forest is the most effective model, achieving the highest overall scores as compared to other ML models, while the Proposed Grid Search technique outperforms and suggests that hyperparameter tuning with Grid Search improves the model's ability to identify relevant instances with improved performance.

Table 5. Performance measures comparison

Model	Data Set	Features	Accuracy	Precision	Recall	F-1 Score
Decision Tree (Ahmed et al., 2023)	Customer reviews, AmazonHelp	TF-IDF	75	-	-	75
Random Forest (Khan et al., 2024)	-	-	78.56	78.56	78.73	78.52
ULMFiT-SVM (AlBadani, Shi, Dong, 2022)	Airline tweets	TF-IDF	78.5	-	-	-
SVM (Alzyout et al., 2021)	Arabic dialect tweets	TF-IDF	78.25	-	-	-

Model	Data Set	Features	Accuracy	Precision	Recall	F-1 Score
Random Forest (Madhuri, 2019)	Twitter (Indian Railways)	-	90.5	89.2	87	81.5
Random Forest (Aufar, Andreswari, Pramesti, 2020)	Product Reviews	TF-IDF	88.2	-	-	-
Random Forest Classifier (Measured by the author)	**Disaster Sentiments**	**TF-IDF**	**93**	**91.2**	**95**	**93**
Proposed Model (Measured by the author)	**Disaster Sentiments**	**TF-IDF**	**94**	**91**	**97**	**94**

Source: Performance measures comparison by the author.

Table 5 compares the performance of various models using different datasets and features, using accuracy, precision, recall, and F-1 score.

Performance measures of this study are highlighted in the table 2, table 3, and table 4. We found that the Random Forest classifier outperforms. The Proposed model Random Forest Grid Search Technique using hyperparameter tuning improves the performance of the Random Forest classifier on the Disaster Sentiments dataset.

4.3 Confusion Matrix and AUC Curve

4.3.1 Confusion matrix

A confusion matrix is a tool used to evaluate the performance of a classification algorithm by displaying true positives, true negatives, false positives, and false negatives. It provides insights into how accurately the model is making predictions. When a dataset contains more than two classes or if the classes are imbalanced with unequal numbers of observations, relying solely on classification accuracy can be misleading. The confusion matrix provides a clearer picture by displaying the model's correct predictions as numerical values and highlighting the errors it is making, allowing for a more detailed evaluation of model performance.

Fig. 6. (a) shows a confusion matrix for the Random Forest classifier. The prediction of the model depicts that 905 were True Positive, and 93 were False Positive while 42 were False Negative and 960 were True Negative.

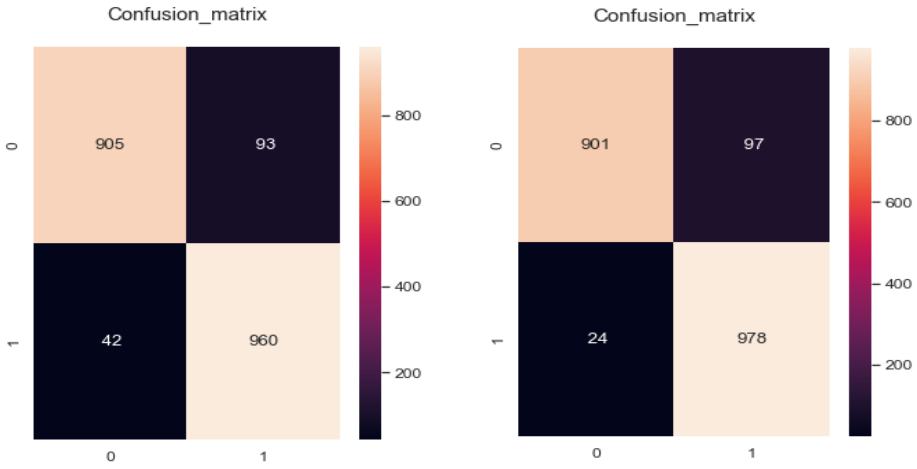

Figure 6.

(a) Confusion matrix for Random Forest **(b) Confusion matrix for the proposed model**

Source: Illustration by authors.

Figure 6 (b) shows a Confusion matrix of proposed model for the Random Forest-Grid Search technique using hyperparameter tuning. The prediction of the model depicts that 901 were True Positive and 97 were False Positive, while 24 were False Negative and 97 were True Negative.

4.3.2 ROC Curve

The ROC curve is often used to evaluate the performance of classification models. Figure 7 demonstrates the performance for proposed model with Random forest classifier which was performing well as compare to other ML models used in this study for sentiment classification.

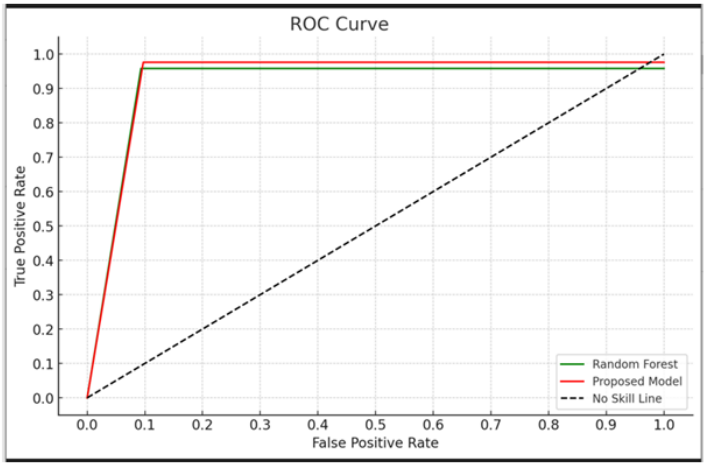

Figure 7. ROC Curve

Source: Illustration by authors.

TPR, also known as recall or sensitivity, measures the proportion of positives correctly identified as such. Figure 7 presents the higher TPR value for the proposed model; it correctly identifies 97.60% of all actual positives, compared to 95.81% by the Random Forest.

The FPR represents the fraction of true negatives that are incorrectly labeled as positives. Both models have low FPRs; however, the Proposed Model has a slightly higher FPR, indicating that it mistakenly identifies 9.72% of all negatives as positives. This implies a modest trade-off in the Proposed Model.

The diagonal dashed line depicts a no-skill classifier that generates random predictions. From this figure, it can be easily depicted that the ROC point is located far from this line which indicates that the performance of the classifiers is good.

Both points are in the top left corner, indicating that the models outperform random guessing. The closer a point is to the top left corner, the better the model's ability to differentiate between the positive and negative classes without making many errors.

The Proposed Model's point is slightly to the right and higher than the Random Forest's point. This position reflects its higher sensitivity and slightly higher error rate on false positives. The ideal point on a ROC curve is the top left corner (TPR = 1, FPR = 0), representing perfect sensitivity without false positives. Both models are close to this ideal, but not at it, showing room for improvement.

5. Conclusions

In this paper, a framework for text preprocessing using NLP approaches and classification using a grid search technique for sentiment analysis was presented. It has been examined that text preprocessing plays a vital role in ML and DL models to improve performance. NLP for text preprocessing comprises different techniques like tokenization, stop word removal, removal of punctuations, part-of-speech tagging, stemming, lemmatization, etc. After applying text preprocessing techniques to the dataset, the Random Forest model achieved a performance of 93%. To improve the performance of the random forest, we used a grid search technique with estimator=RandomForestClassifier(), param_grid='max_depth': [2, 4, 5, 6, 7, 8, 9, 10, 12, 16], 'n_estimators': [10, 15, 18, 50, 100], and ‘random_state': [3, 42, 777]. We found that the best parameters for improving performance measures in GridSearch were 'max_depth': 6, 'n_estimators': 15 and 'random_state': 777. The accuracy measure for the grid search was 94%.

References

[1] Agarwal, A., Xie, B., Vovsha, I., Rambow, O., Passonneau, R. (2011), *Sentiment Analysis of Twitter Data. Proceedings of the Workshop on Languages in Social Media,* 30-38. Portland, Oregon, USA: Association for Computational Linguistics.

[2] Ahmed, C., ElKorany, A., ElSayed, E. (2023), *Prediction of customer’s perception in social networks by integrating sentiment analysis and machine learning. Journal of Intelligent Information Systems,* 60(3), 829-851.

[3] Ahuja, R., Chug, A., Kohli, S., Gupta, S., Ahuja, P. (2019), *The impact of feature extraction on sentiment analysis. Procedia Computer Science,* 152, 341-348.

[4] Alaei, A.R., Becken, S., Stantic, B. (2019), *Sentiment Analysis in Tourism: Capitalizing on Big Data, J Travel Res,* 58(2), 175-191.

[5] Alam, S., Yao, N. (2019), *The Impact of Preprocessing Steps on the Accuracy of Machine Learning Algorithms in Sentiment Analysis. Computational and Mathematical Organization Theory,* 25, 319-335.

[6] AlBadani, B., Shi, R., Dong, J. (2022), *A novel machine learning approach for sentiment analysis on Twitter incorporating the universal language model fine-tuning and SVM. Applied Systems Innovation,* 5(1), 13.

[7] Al-Smadi, M., Al-Ayyoub, M., Jararweh, Y., Qawasmeh, O. (2019), *Enhancing Aspect-Based Sentiment Analysis of Arabic Hotels' Reviews Using Morphological, Syntactic, and Semantic Features. Information Processing & Management,* 56(2), 308-319.

[8] Alzyout, M., Bashabsheh, E.A., Najadat, H., Alaiad, A. (2021), *Sentiment analysis of Arabic tweets about violence against women using machine learning. In 2021 12th International Conference on Information and Communication Systems (ICICS),* 171-176, IEEE, Valencia, Spain (held virtually).

[9] Aufar, M., Andreswari, R., Pramesti, D. (2020), *Sentiment Analysis on YouTube Social Media Using Decision Tree and Random Forest Algorithm: A Case Study. In 2020 International Conference on Data Science and Its Applications (ICoDSA) Bandung, Indonesia.,* 1-7. IEEE, Bandung, Indonesia.

[10] Bahaghighat, M., Akbari, L., Xin, Q. (2019), *A Machine Learning-Based Approach for Counting Blister Cards within Drug Packages,* IEEE Access, 7, 83785-83796.

[11] Dragoni, M., Petrucci, G. (2018), *A Fuzzy-Based Strategy for Multi-Domain Sentiment Analysis. International Journal of Approximate Reasoning,* 93, 59-73.

[12] Fouad, M.M., Gharib, T.F., Mashat, A.S. (2018), *Efficient Twitter sentiment analysis system with feature selection and classifier ensemble. In The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2018)*, 516-527. Springer, Cairo, Egypt.

[13] Hussein, D.M.E.-D.M. (2018), *A Survey on Sentiment Analysis Challenges. Journal of King Saud University - Engineering Sciences,* 30(4), 330-338.

[14] Jianqiang, Z., Xiaolin, G., Xuejun, Z. (2018), *Deep Convolution Neural Networks for Twitter Sentiment Analysis, IEEE Access,* 6, 23253–23260.

[15] Kamale, A.S., Deshmukh, P.K., Dhainje, P.B. (2015), *A Survey on Classification Techniques for Feature-Sentiment Analysis. International Journal on Recent and Innovation Trends in Computing and Communication,* 3(7), 4823-4829.

[16] Khan, T.A., Sadiq, R., Shahid, Z., Alam, M.M., Su'ud, M.B.M. (2024), *Sentiment analysis using support vector machine and random forest. Journal of Informatics and Web Engineering,* 3(1), 67-75.

[17] Li, J., Qiu, L. (2017*), A Sentiment Analysis Method of Short Texts in Microblog, IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC),* Guangzhou, 776-779.

[18] Madhuri, D.K. (2019), *A machine learning-based framework for sentiment classification: Indian railways case study. International Journal of Innovative Technology and Exploring Engineering (IJITEE),* 8(4), 441-445.

[19] Morente-Molinera, J.A., Kou, G., Samuylov, K., Ureña, R., Herrera-Viedma, E. (2019), *Carrying Out Consensual Group Decision-Making Processes under Social Networks Using Sentiment Analysis over Comparative Expressions. Knowledge-Based System,* 165, 335-345.

[20] Nazare, S.P., Nar, P.S., Phate, A.S., Ingle, D.R. (2018), *Sentiment Analysis in Twitter. International Research Journal of Engineering and Technology (IRJET),* 5(1), 880-886.

[21] Nguyen, D.Q., Vu, T., Pham, S.B. (2014), *Sentiment classification on polarity reviews: an empirical study using rating-based features*. In *Proceedings of the 5th workshop on computational approaches to subjectivity, sentiment and social media analysis*, 128-135. Association for Computational Linguistics. Baltimore, Maryland, USA.

[22] Pham, D.H., Le, A.C. (2018), *Learning Multiple Layers of Knowledge Representation for Aspect-Based Sentiment Analysis, Data & Knowledge Engineering,* 114, 26-39.

[23] Prusa, J.D., Khoshgoftaar, T.M., Dittman, D.J. (2015), *Impact of Feature Selection Techniques for Tweet Sentiment Classification*. *In Proceedings of the 28th International Florida Artificial Intelligence Research Society Conference*, 299-304. Association for the Advancement of Artificial Intelligence. Hollywood, Florida, USA.

[24] Sajadi, M.S.S., Babaie, M., Bahaghighat, M. (2018), *Design and Implementation of Fuzzy Supervisor Controller on Optimized DC Machine Driver*. *In 2018 8th Conference of AI & Robotics and 10th RoboCup Iranopen International Symposium*, 26-31. IEEE. Qazvin, Iran.

[25] Sankar, H., Subramaniyaswamy, V., Vijayakumar, V., Arun Kumar, S., Logesh, R., Umamakeswari, A.J.S.P. (2020), *Intelligent sentiment analysis approach using edge computing-based deep learning technique. Software: Practice and Experience,* 50(5), 645-657.

[26] Spatiotis, N., Paraskevas, M., Perikos, I., Mporas, I. (2017), *Examining the impact of feature selection on sentiment analysis for the Greek language. In Speech and Computer: 19th International Conference, SPECOM 2017, Hatfield, UK*. Springer International Publishing. 353-361.

[27] Tu, Z., He, Y., Foster, J., Van Genabith, J., Liu, Q., Lin, S. (2012), *Identifying High-Impact Sub-Structures for Convolution Kernels in Document-Level Sentiment Classification. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2, Association for Computational Linguistics,* 338-343.

[28] Xie, X., Ge, S., Hu, F., Xie, M., Jiang, N. (2019), *An Improved Algorithm for Sentiment Analysis Based on Maximum Entropy, Soft Computing,* 23(2), 599-611.

[29] Zhang, L., Wang, S., Liu, B. (2018). *Deep learning for sentiment analysis: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery,* 8(4), e1253.