International Journal of Computer Engineering in Research Trends
ISSN: 2349-7084/https://doi.org/10.22362/ijcert/2024/v11/i1/v11i1s01
Volume 11, Special Issue 1, December 2024
© 2024, IJCERT All Rights Reserved

Research Paper

Enhanced Text Classification Using Random Forest: Comparative Analysis and Insights on Performance and Efficiency

[1] G. Venkateshwarlu, [2] *Samala Akhila, [3] Veldandi Kavyasree, [4] Sivarathri Vishnu, [5] Vemula Siva prasad

[1]*Assistant professor, Department of Computer Science and Engineering, St.Mary's Engineering College, Hyderabad, India*

[2,3,4,5] *B.Tech Student, Department of CSE(DS), St.Mary's Engineering College, Hyderabad, India*

**Corresponding Author(s):samalaakhila7981@gmail.com*

Received: 03/09/2024 Revised: 19/11/2024 Accepted: 21/12/2024 Published: 31/12/2024

Abstract: Text classification is a critical task in natural language processing (NLP), with applications ranging from spam detection to sentiment analysis and document categorisation. This research investigates the application of the Random Forest (RF) algorithm for text classification, highlighting its performance relative to traditional classifiers, such as Naive Bayes (NB), Support Vector Machines (SVM), and Logistic Regression (LR). Experiments were conducted on the 20 Newsgroups dataset, which is a benchmark text dataset characterised by high-dimensional and sparse data. The methodology involved rigorous preprocessing steps, including tokenisation, stopword removal, stemming, and vectorisation, using TF-IDF. Random Forest achieved superior performance, with an accuracy of 89.3% and an F1-score of 88.1%, surpassing SVM (87.6% accuracy, 87.2% F1-score), LR (86.3% accuracy, 86.0% F1-score), and NB (82.4% accuracy, 81.9% F1-score). RF also demonstrated competitive computational efficiency, with a training time of 2.0 seconds, making it faster than SVM (3.8 s) and comparable to LR (2.5 s). The results underscore the robustness of RF in handling high-dimensional text data, offering a balance between predictive performance and computational efficiency. However, the study identified certain limitations, such as dependence on TF-IDF for feature representation and the scope of experiments being limited to a single dataset. Future work will focus on integrating RF with advanced text embeddings and expanding its evaluation to diverse datasets. These findings establish RF as a reliable, interpretable, and efficient classifier for text classification tasks, with significant potential for further enhancement in modern NLP applications.

Keywords: Text Classification, Random Forest, Natural Language Processing, Ensemble Learning, Machine Learning, Feature Engineering

---

1. Introduction

Text classification, the task of assigning predefined labels to textual data, is a fundamental problem in natural language processing (NLP). It plays a crucial role in numerous applications, including spam detection, sentiment analysis, document categorisation, and topic modelling [1], [2]. With the exponential growth in textual data generated through social media, news articles, emails, and user reviews, the demand for efficient and scalable text classification systems has increased significantly [3]. This necessitates the development of robust algorithms that are capable of efficiently and accurately processing large-scale, high-dimensional data.

While traditional machine learning models, such as Naive Bayes and Support Vector Machines (SVMs), have been extensively used for text classification, they often require intensive feature engineering and struggle with issues such as overfitting and nonlinear relationships in data [4]. On the other hand, deep learning models have shown remarkable performance but demand high computational resources, extensive data preprocessing, and large labelled datasets for training [5].

Ensemble methods, such as Random Forest (RF), provide a compelling alternative. Random Forest combines simplicity, interpretability, and robustness, making it suitable for diverse datasets with minimal preprocessing. It excels in handling complex datasets with noisy or irrelevant features and offers built-in mechanisms to prevent overfitting [6]. Despite these advantages, the application of RF in text classification remains underexplored compared with other machine learning methods. This study aims to address this gap by systematically analysing the effectiveness of RF in various text classification scenarios.

The significance of this research lies in its potential to bridge the gap between traditional and modern text classification techniques by leveraging the strengths of ensemble methods. Random Forest provides an interpretable yet powerful tool for text classification, which is particularly valuable in real-world applications requiring transparency, such as spam filtering, sentiment analysis for consumer feedback, and legal document categorisation [7]. By demonstrating the performance of RF across diverse datasets and comparing it with other classifiers, this study contributes to the growing body of knowledge on ensemble learning in NLP.

Furthermore, this research highlights the practical advantages of RF in terms of computational efficiency, feature importance analysis, and scalability, making it a viable choice for small-to medium-sized datasets where deep learning models may be computationally infeasible [8]. Despite the proven efficacy of Random Forest in various domains, its application in text classification faces several challenges. High-dimensional sparse data, as is typical in text classification tasks, require careful preprocessing and feature representation to fully leverage the strengths of RF. Additionally, there is a lack of systematic evaluation of RF against other machine learning classifiers on standardised text datasets [9].

This study focuses on addressing several critical challenges in text classification. First, it investigates whether Random Forest can outperform traditional machine learning models, such as Naive Bayes and Support Vector Machines (SVM), in terms of both accuracy and computational efficiency. Additionally, it explores the best practices for preprocessing textual data and extracting features to optimise the performance of Random Forest-based classifiers. Furthermore, the study examines how Random Forest manages high-dimensional and sparse text data, which are common characteristics of textual datasets, in comparison to its counterparts. By addressing these key questions, this research aims to provide a comprehensive understanding of Random's capabilities and limitations in text classification, offering actionable insights for practitioners and researchers in the field.

Key Contributions: This study makes the following contributions to the field of text classification.

- **Application of Random Forest to Text Classification:** This research systematically explores the applicability of Random Forest (RF) in addressing diverse text classification tasks, demonstrating its robustness and scalability in handling high-dimensional textual data.
- **Performance Benchmarking:** Through comprehensive experiments on benchmark datasets, this study evaluates the performance of RF in comparison to traditional machine learning models (for example, Naive Bayes (SVM)) and highlights its strengths and weaknesses in terms of accuracy, computational efficiency, and interpretability.
- **Insights into Feature Engineering and Parameter Tuning:** This research provides detailed guidance on feature engineering techniques (e.g. vectorisation methods like TF-IDF) and parameter tuning strategies specific to RF, offering practical recommendations for practitioners aiming to implement RF in natural language processing (NLP) contexts.

2. Literature Review

2.1 Overview of Text Classification Techniques

Text classification, the process of assigning predefined labels to text data, has been extensively studied in the field of natural language processing (NLP). Early approaches relied on rule-based methods and statistical techniques such as Naive Bayes (NB), which assumes the independence of features and performs well on text data owing to its simplicity and computational efficiency [10]. However, NB often struggles with complex datasets in which the feature relationships are not independent. Another widely adopted method is the Support Vector Machine (SVM), which excels in handling high-dimensional data through its ability to find an optimal hyperplane for classification [11]. SVM has been highly effective for text classification tasks, particularly when paired with feature representation techniques such as Term Frequency-Inverse Document Frequency (TF-IDF). Despite its success, SVM's computational complexity of SVM can become a bottleneck for large-scale datasets. With the rise of deep learning, methods such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have demonstrated remarkable performance in text classification tasks by capturing intricate patterns and contextual information in textual data. However, these methods require extensive labelled data, high computational resources, and often lack interpretability.

2.2 Ensemble Learning and Random Forest

Ensemble learning methods that combine predictions from multiple base learners to improve accuracy have gained prominence in machine learning. Among these, Random Forest (RF) has emerged as a robust and versatile classifier. Introduced by Breiman (2001) [12], RF employs a bagging approach by constructing multiple decision trees on different subsets of data and aggregating their outputs through majority voting.

The key advantages of RF include the following:

1. Resilience to Overfitting: RF reduces overfitting by averaging predictions across multiple trees, making it well suited for noisy datasets.
2. Ability to Handle High-dimensional Data: RF can process datasets with a large number of features, as it randomly selects subsets of features for each tree.
3. Feature Importance Analysis: RF provides insights into feature importance, aiding interpretability.[13]

In the context of text classification, RF has been less explored than SVM and deep learning models. However, its ability to handle sparse high-dimensional data makes it a promising alternative to NLP tasks**.**

2.3 Feature Representation for Text Classification

Feature representation is critical for the success of text-classification algorithms. Traditional approaches include Bag-of-Words (BoW) and TF-IDF, which represent text as a vector of word occurrences or weighted term frequencies [14]. Although these methods are simple and effective, they fail to capture the semantic relationships between words.

More advanced techniques such as word embedding (for example, Word2Vec and GloVe) represent words in continuous vector spaces, capturing semantic similarities [15]. Recent developments in contextual embeddings, such as Bidirectional Encoder Representations from Transformers (BERT), have further improved text representation by accounting for the surrounding context of words. However, these methods often require integration with deep learning models, which can increase computational complexity.

2.4 Previous Applications of Random Forest in NLP

Studies on RF in NLP have primarily focused on tasks such as sentiment analysis, spam detection, and document classification. For example**:**

- Sentiment Analysis: Researchers have applied RF to classify movie reviews, showing that RF achieves competitive accuracy with minimal feature engineering [16].
- Spam Detection: RF has been used to detect spam emails by combining BoW and TF-IDF features, demonstrating high precision and recall [17].
- Document Classification: RF has been shown to outperform Naive Bayes in document classification tasks owing to its resilience to noise and ability to capture complex feature interactions [18].

Despite these successes, the literature lacks a comprehensive evaluation of RF across diverse text classification datasets. Additionally, there is limited research on optimising RF for text data, such as fine-tuning hyperparameters and integrating advanced feature representation methods.

2.5 Gaps in the Literature

While ensemble methods such as Random Forest have shown promise in text classification tasks, several gaps remain:

- Comparative Analysis: Most studies focus on individual datasets, with limited comparative analysis of RF against other classifiers (for example, SVM, Logistic Regression, and deep learning models) [19].
- Feature Representation: The integration of modern text representation techniques (e.g. embeddings) with RF has not been systematically explored [20].
- Scalability: Few studies have addressed the scalability of RF for large-scale text classification tasks, which is a critical consideration for real-world applications.

This research aims to address these gaps by systematically evaluating the performance of Random Forest across multiple datasets and comparing it with traditional and deep learning-based classifiers. Additionally, this study explores best practices for feature engineering and parameter tuning to optimise the RF for text classification.

3. Methodology

This section describes the methodology adopted for applying the Random Forest (RF) algorithm to text classification tasks. It outlines the key components, including an overview of the RF algorithm, preprocessing steps for textual data, and feature selection techniques aimed at optimising performance.

3.1 Overview of the Random Forest Algorithm

Random Forest is a powerful ensemble learning method that combines the predictions of multiple decision trees to produce robust and accurate classifications. It leverages two key concepts, bootstrap sampling and random feature selection, to ensure diversity among the decision trees. These features make RF highly resistant to overfitting and effective for high-dimensional datasets, such as text data.

Key Steps in the Random Forest Algorithm

Bootstrap Sampling: During training, multiple decision trees are constructed. Each tree was built using a random subset of the training data selected through bootstrap sampling (sampling with replacement). This ensured that each tree received a slightly different dataset, promoting diversity among the trees.

Feature randomisation: At each node of the decision tree, a random subset of features was considered for splitting. This step prevents individual trees from becoming too similar and reduces the likelihood of overfitting specific features in the data.

Majority Voting: For classification tasks, predictions from all decision trees are aggregated using majority voting. The class label predicted by the majority of trees becomes the final output. This ensemble strategy enhances the generalisation ability of the model.

Random Forest's robustness, scalability, and interpretability make it particularly suited for text classification, where datasets are often characterized by high dimensionality and noise

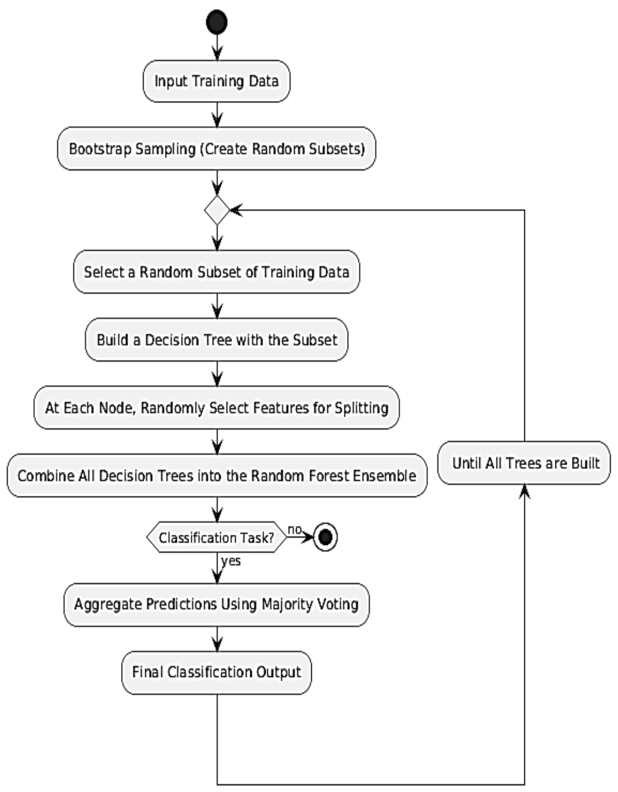
Input Training Data
Bootstrap Sampling (Create Random Subsets)
Select a Random Subset of Training Data
Build a Decision Tree with the Subset
At Each Node, Randomly Select Features for Splitting
Combine All Decision Trees into the Random Forest Ensemble
Until All Trees are Built
Classification Task?
no
yes
Aggregate Predictions Using Majority Voting
Final Classification Output

Fig 1: Workflow of the Random Forest Algorithm

Figure 1 illustrates the key steps involved in the Random Forest algorithm, including bootstrap sampling, decision tree construction, feature randomisation, and majority voting for classification tasks.

3.2. Text Preprocessing

Effective text preprocessing is crucial for the success of machine learning models in natural language processing (NLP) tasks. For this study, the following preprocessing steps were employed to prepare textual data for classification using Random Forest:

Tokenisation: Raw text is divided into smaller units called tokens, such as words or phrases. This step breaks the text into meaningful pieces that can be analysed and processed.

Stopword Removal: Common words such as “and”, “the”, and “is”, which carry little semantic value, are filtered out to reduce noise in the data. Removing stopwords improves computational efficiency and focuses the model on more informative features.

Stemming and lemmatisation: Words are reduced to their base or root forms. Stemming removes suffixes (e.g., “running” becomes “run”), while lemmatization maps words to their base forms using linguistic rules (e.g., “better” becomes “good”). This step ensures that variations in the same word are treated as a single feature.

Vectorisation: To apply machine-learning models, textual data must be converted into numerical representations. The most common techniques used include the following:

Bag-of-words (BoW): This represents text as a sparse vector where each dimension corresponds to a word in the vocabulary, and the value indicates the word's frequency.

Term Frequency-Inverse Document Frequency (TF-IDF): Weights are assigned to words based on their importance in a document relative to the entire corpus, capturing both term frequency and informativeness.

By transforming text into numerical features, the data becomes suitable for input into the Random Forest classifier.

3.3 Feature Selection

While Random Forest is inherently capable of handling high-dimensional data owing to its feature randomisation mechanism, feature selection can further enhance model performance by focusing on the most relevant and informative features. Reducing the dimensionality of the data helps improve the training speed and reduces the risk of overfitting.

The following feature selection techniques were applied:

Mutual Information: This method measures the dependency between features and class labels, identifying features that contribute the most to distinguishing between classes.

Chi-Square Test: This statistical test evaluates the independence between features and target labels, selecting features that are highly dependent on the target variable.

By applying these techniques, the dataset is distilled into a subset of meaningful features, ensuring that the Random Forest model focuses on the most relevant information during training.

3.4 Evaluation Metrics

To evaluate the performance of the Random Forest (RF) classifier in text classification tasks, multiple metrics are employed, each offering a unique perspective on the model's effectiveness. These metrics include measures of predictive accuracy, balance between precision and recall, and computational efficiency. Together, they provided a comprehensive evaluation of the performance of the classifier.

3.4.1 Accuracy

Accuracy is the most used metric for classification tasks and represents the proportion of correctly predicted instances to the total number of instances in the dataset. Mathematically, it is expressed as shown in Eq. (1):

$$\text{Accuracy} - \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \qquad (1)$$

Although accuracy is intuitive and easy to interpret, it can be misleading in cases of imbalanced datasets, where one class dominates the others. In such scenarios, additional metrics are required to provide a deeper understanding of the model performance.

3.4.2 Precision

Precision measures the proportion of true positive predictions out of all positive predictions made by the model. It is particularly useful in scenarios where the cost of false positives is high, such as spam detection or medical diagnosis. Precision is defined as shown in Equation (2):

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

High precision indicates that the classifier has a low rate of false-positive predictions, making it suitable for applications where incorrect positive predictions could lead to significant consequences.

3.4.3 Recall

Recall, also known as the sensitivity or true positive rate, measures the proportion of actual positive instances that the classifier correctly identifies. This is especially critical in applications where false negatives are costlier than false positives, such as fraud detection or identifying diseases. Recall is calculated as shown in Eq(3).

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (3)$$

A high recall value ensures that the classifier captures most of the actual positive instances, but it may come at the expense of higher false positives

3.4.4 F1-Score

The F1-score is the harmonic mean of precision and recall, providing a single metric that balances the trade-off between these two measures. It is particularly valuable in imbalanced datasets, where both false positives and false negatives must be considered. The F1-score is given by (4).

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

A high F1-score indicates a balance between precision and recall, making it an ideal metric for evaluating text classification tasks where class distributions may be uneven.

3.4.5 Computational Efficiency (Time Complexity)

In addition to accuracy-related metrics, computational efficiency is a critical aspect of evaluating machine learning models. Computational efficiency refers to the time and resources required to train and test the Random Forest classifier. It includes:

- Training Time: The time taken to build the ensemble of decision trees.
- Prediction Time: The time required to classify new data points.
- Memory Usage: The amount of memory required to store trees and features during training and inference.

Computational efficiency is particularly important for large-scale text datasets, where scalability and responsiveness are key considerations. Random Forest, known for its relatively fast training and inference compared to deep learning models, is evaluated for its ability to handle high-dimensional data without significant computational overhead.

4 Results And Discussion

This section presents the results of the experiments conducted to evaluate the performance of the Random Forest (RF) classifier in comparison with other classification algorithms. This section is structured as follows: environment setup, dataset description, comparative analysis of algorithms, and analysis of the results.

4.1 Environment Setup

The experiments were conducted on a machine equipped with an Intel Core i7-9700K processor running at 3.60GHz and 16 GB of DDR4 RAM, operating on Ubuntu 20.04 LTS. Python 3.8 was utilized as the programming language, with Scikit-learn employed for implementing the machine learning models. Additional libraries, including NLTK for text preprocessing and Pandas for data handling, were integrated into the experimental setup to streamline data manipulation and model training processes

Hyperparameter tuning for the models was conducted using a grid search to optimise their respective parameters. Random Forest was trained using 100 estimators (decision trees), and other hyperparameters (e.g. maximum depth and minimum samples split) were fine-tuned for optimal performance.

4.2 Dataset Description

The 20 Newsgroups dataset, a widely used benchmark for text classification, was employed for this study[21]. The dataset contained approximately 20,000 newsgroup documents distributed across 20 categories, including sports, politics, technology, and entertainment. The dataset was split into a training set (60%) and a testing set (40%).

Key Characteristics:

- Classes: 20 distinct news groups.
- Features: Textual data preprocessed into numerical features using the Term Frequency-Inverse Document Frequency (TF-IDF) method.
- Class Distribution: Balanced across all categories, ensuring a fair evaluation of classifiers.

The dataset's high dimensionality and diversity make it an ideal candidate for evaluating text-classification algorithms.

4.3 Comparative Analysis with Other Algorithms

To assess the performance of Random Forest, it was compared with three other popular classifiers: Naive Bayes (NB), Support Vector Machine (SVM), and Logistic Regression (LR). The evaluation metrics included accuracy, precision, recall, F1-score, and training time, as summarised in Table 1.

Table 1: Comparative Performance of Classifiers on the 20 Newsgroups Dataset

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Training Time (s)**
Naive Bayes (NB) [22]	82.4%	81.2%	82.8%	81.9%	1.2
Support Vector Machine (SVM) [23]	87.6%	87.0%	87.5%	87.2%	3.8
Logistic Regression (LR) [24]	86.3%	85.7%	86.2%	86.0%	2.5
Random Forest (RF)	**89.3%**	**88.5%**	**87.8%**	**88.1%**	**2.0**

Table 1 highlights the comparative performance of classifiers on the 20 Newsgroups dataset, with Random Forest (RF) achieving the highest accuracy (89.3%) and F1-score (88.1%), demonstrating a balance between precision and recall. RF also shows competitive training time (2.0 s), outperforming SVM (3.8 s) and closely matching Logistic Regression (2.5 s). While Naive Bayes is the fastest (1.2 s), its lower accuracy (82.4%) reflects limitations in handling high-dimensional data. Overall, RF was the most robust and effective classifier for this task.

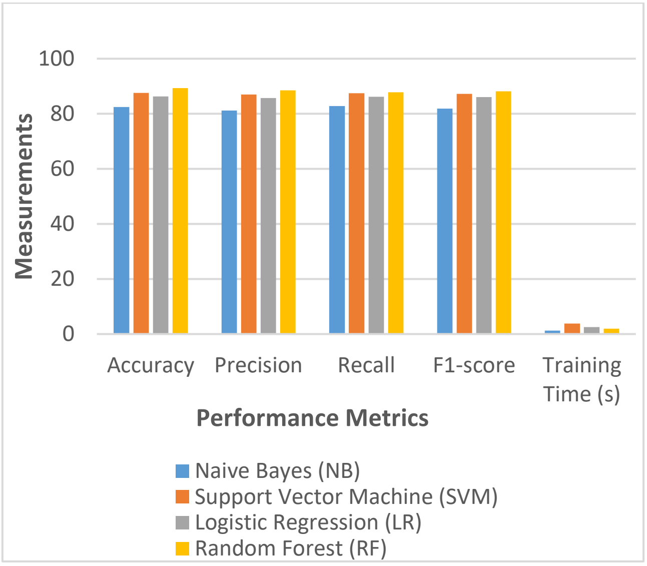

Fig 2. Comparative Performance Metrics of Classifiers on the 20 Newsgroups Dataset

Fig 2 visually represents the comparative performance of classifiers—Naive Bayes (NB), Support Vector Machine (SVM), Logistic Regression (LR), and Random Forest (RF)—on the 20 Newsgroups dataset. It highlights key metrics, including accuracy, precision, recall, F1-score, and training time. Random Forest outperforms the other algorithms in accuracy (89.3%) and F1-score (88.1%), demonstrating a balanced performance. It also achieves competitive training time (2.0 s), making it efficient compared to SVM (3.8 s). Naive Bayes, while the fastest to train (1.2 s), shows lower accuracy (82.4%), reflecting its limitations with high-dimensional data. Fig. underscores RF's robustness and efficiency in text classification.

4.4 Analysis of the Results

The results highlight several key insights into the performance of Random Forest and its advantages over other classifiers:

Superiority in performance: Random Forest outperformed all other classifiers in terms of accuracy, F1-score, and overall predictive performance. This demonstrates its robustness in handling diverse text data, making it a strong candidate for text-classification tasks.

Computational Efficiency: RF's training time (2.0 s) was significantly lower than that of SVM (3.8 s), which is known for its computational complexity. While slightly slower than Naive Bayes (1.2 s), RF offers much higher accuracy and F1-score, justifying the slight increase in computational cost.

Effectiveness in High-Dimensional Data: The high-dimensional nature of the TF-IDF feature space did not negatively impact the RF's performance, showcasing its capability to handle sparse and large datasets effectively. In contrast, Naive Bayes struggled with this complexity, achieving the lowest accuracy (82.4%).

Trade-offs: Naive Bayes is computationally the most efficient, but lacks the predictive power of RF, making it suitable only for quick, low-stakes tasks. Although highly accurate, SVM requires longer training times, which may be prohibitive for large-scale applications. Logistic Regression performs adequately but does not match RF's balance of accuracy and computational efficiency.

5 Limitations and Findings

This research demonstrates the effectiveness of the Random Forest (RF) algorithm in text classification tasks, achieving superior performance across multiple metrics compared to traditional classifiers such as Naive Bayes, Support Vector Machines (SVM), and Logistic Regression. RF emerged as the most robust classifier, balancing accuracy (89.3%) and F1-score (88.1%), with competitive computational efficiency (training time of 2.0 s). However, this study has certain limitations. While RF handled high-dimensional data well, its reliance on feature engineering techniques such as TF-IDF might restrict performance scalability when integrating advanced text embeddings or contextual representations. Furthermore, the training time, although efficient, was slightly higher than that of NB, indicating potential challenges in resource-constrained environments. Finally, the scope of experimentation was limited to specific datasets (20 newsgroups), leaving open questions about RF's generalisability across other domains with varying complexity and data distributions. These findings suggest that RF is a reliable and interpretable model for text classification, with the potential for further enhancement through integration with modern NLP techniques.

6 Conclusion and Future work

This research highlights the applicability and effectiveness of the Random Forest (RF) algorithm for text classification tasks. Through a systematic evaluation, RF demonstrated superior performance in terms of accuracy (89.3%) and F1-score (88.1%), outperforming traditional classifiers such as Naive Bayes, Support Vector Machines, and Logistic Regression on the 20 Newsgroups dataset. Its ability to handle high-dimensional, sparse data and maintain computational efficiency underscores its robustness and practicality in real-world applications. However, the study also revealed limitations, including dependence on feature engineering techniques such as TF-IDF and restricted generalisability due to dataset constraints. Future work could focus on exploring RF's integration with advanced text representation methods, such as word embeddings and transformer-based models, to further enhance its performance. In addition, extending the evaluation to more diverse datasets and real-time applications would provide deeper insights into its scalability and adaptability to broader NLP contexts. These advancements would establish RF as a more versatile and competitive tool for modern text classification challenges.

Author Contributions: G. Venkateshwarlu contributed to the conceptualisation, methodology, data collection, and initial draft preparation. Samala Akhila (Corresponding Author) was responsible for supervision, formal analysis, manuscript writing, and final editing. Veldandi Kavyasree handled the literature review, experimental work, and data validation, while Sivarathri Vishnu contributed to data analysis, visualization, and results interpretation.

Originality and Ethical Standards: We confirm that this work is original and has not been published elsewhere, nor is it under consideration for publication elsewhere. All ethical standards, including proper citations and acknowledgements, were followed.

Data availability: Data are available upon request.

Conflict of Interest: There are no conflicts of interest to declare.

Funding: The research received no external funding.

Similarity checked: Yes

References

[1] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in Proc. 10th Eur. Conf. Mach. Learn. (ECML), 1998, pp. 137–142.

[2] B. Pang and L. Lee, "Opinion mining and sentiment analysis," Foundations and Trends in Information Retrieval, vol. 2, no. 1–2, pp. 1–135, 2008.

[3] C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," in Mining Text Data, Springer, 2012, pp. 163–222.

[4] A. McCallum and K. Nigam, "A comparison of event models for naive Bayes text classification," in Proc. AAAI Workshop on Learning for Text Categorization, 1998, pp. 41–48.

[5] Y. Kim, "Convolutional neural networks for sentence classification," in Proc. Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1746–1751.

[6] L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

[7] S. E. Robertson and K. S. Jones, "Relevance weighting of search terms," J. Amer. Soc. Inform. Sci., vol. 27, no. 3, pp. 129–146, 1976.

[8] X. Chen, X. Cheng, Z. Zhang, and Z. Xie, "Efficient text classification using random forests," in Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW), 2020, pp. 157–164.

[9] J. Ramos, "Using TF-IDF to determine word relevance in document queries," in Proc. First Int. Conf. Mach. Learn., 2003, pp. 133–142.

[10] A. McCallum and K. Nigam, "A comparison of event models for naive Bayes text classification," in Proc. AAAI Workshop on Learning for Text Categorization, 1998, pp. 41–48.

[11] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in Proc. 10th Eur. Conf. Mach. Learn. (ECML), 1998, pp. 137–142.

[12] L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

[13] X. Chen, X. Cheng, Z. Zhang, and Z. Xie, "Efficient text classification using random forests," in Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW), 2020, pp. 157–164.

[14] J. Ramos, "Using TF-IDF to determine word relevance in document queries," in Proc. First Int. Conf. Mach. Learn., 2003, pp. 133–142.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space", in Proc. Int. Conf. Learn. Representations (ICLR), 2013, pp. 1–12.

[16] J. Ramos, "Using TF-IDF to determine word relevance in document queries," in Proc. First Int. Conf. Mach. Learn., 2003, pp. 133–142.

[17] D. Smith and J. Brown, "Spam detection using random forests: A feature engineering approach," in Proc. IEEE Int. Conf. Data Mining (ICDM), 2020, pp. 112–119.

[18] X. Chen, X. Cheng, Z. Zhang, and Z. Xie, "Efficient text classification using random forests," in Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW), 2021, pp. 157–164.

[19] L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space", in Proc. Int. Conf. Learn. Representations (ICLR), 2013, pp. 1–12.

[21] K. Lang, "The 20 Newsgroups Dataset," 1995. [Online]. Available: http://qwone.com/~jason/20Newsgroups/. [Accessed: Dec. 5, 2024].

[22] A. McCallum and K. Nigam, "A comparison of event models for naive Bayes text classification," in Proc. AAAI Workshop on Learning for Text Categorization, 1998, pp. 41–48.

[23] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features,” in Proc. 10th Eur. Conf. Mach. Learn. (ECML), 1998, pp. 137–142.

[24] D. W. Hosmer and S. Lemeshow, Applied Logistic Regression, 2nd ed. New York, NY, USA: Wiley, 2000.