Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7923068

FAKE NEWS DETECTION USING ENHANCED STACKING ENSEMBLE CLASSIFICATION ALGORITHM

Raut Rahul Ganpat and Dr. Sonawane Vijay Ramnath
Department of Computer Science and Engineering
Dr. A. P. J. Abdul Kalam University, Indore (M. P.) – 452010
Corresponding Author Email : braut_rahul@yahoo.co.in

Abstract:
Fake news refers to deliberately false or misleading information presented as factual news. The rise of fake news has led to the need for efficient detection techniques to prevent its harmful effects on individuals and society. Previous fake news detection techniques relied on features such as linguistic cues and stylistic features. However, these techniques had limitations in terms of accuracy and generalizability. To overcome these limitations, this paper presents a machine learning algorithm for detecting fake news namely an enhanced stacking ensemble classification algorithm (ES-ECA). The proposed algorithm uses preprocessing techniques to split the dataset into individual statements and generates n-grams as features. Feature extraction is then performed using the term frequency-inverse document frequency (TF-IDF) measure, which quantifies the importance of each feature to the document. The proposed algorithm uses an enhanced stacking ensemble approach that combines base classifiers, including an enhanced version of the J48 decision tree algorithm, an enhanced version of the Naive Bayes algorithm, and an enhanced version of the k-Nearest Neighbors algorithm. A meta-classifier is then created using a Random Forest algorithm boosted using the AdaBoostM1 algorithm. The RandomSubSpace algorithm is used to improve the performance of the stacking ensemble classifier. Experimental results show that the proposed algorithm outperforms previous techniques in terms of accuracy, precision, recall, and F1-score. The ensemble classifier achieved an accuracy of 75.18% and an F1-score of 81.81%, outperforming the individual classifiers. These results suggest that the proposed algorithm is effective in detecting fake news and can be used to mitigate the harmful effects of fake news on society.

Keywords: Classification, stacking, preprocessing, n-gram, TF-IDF

1. Introduction

Fake news has become a significant issue in recent years, particularly with the widespread availability of the internet and social media platforms. The ease of creating and disseminating false information has made it difficult to distinguish between real and fake news [1]. This issue is of great concern as it has the potential to cause harm to both individuals and society. For example, false information can cause panic, confusion, and even physical harm if it leads to people making decisions based on incorrect information [2].

To address this problem, researchers and practitioners have been working on developing fake news detection techniques [3]. These techniques aim to identify and prevent the spread of fake news before it can cause harm. One common approach used in these techniques is to rely on linguistic and stylistic features to distinguish between real and fake news [4].

However, there are limitations to using linguistic and stylistic features in fake news detection. One of the main challenges is that the features used may not always be reliable indicators of fake news [5]. For example, some fake news stories may be written using correct grammar and language, making it difficult to identify them based on linguistic and stylistic features alone. Additionally, these features may not be generalizable across different datasets, making it challenging to develop effective detection techniques that work across a range of scenarios. Therefore, a novel machine learning algorithm for detecting fake news using an enhanced stacking ensemble classification algorithm (ES-ECA) has been proposed.

The proposed algorithm splits the dataset into individual statements and generates n-grams as features. This approach enables the algorithm to capture more nuanced and subtle features that may be missed by traditional linguistic and stylistic feature extraction techniques [19]. The term frequency-inverse document frequency (TF-IDF) measure is used to extract features that are most relevant to the document. This approach assigns a weight to each feature based on its importance to the document, allowing the algorithm to focus on the most informative features.

The algorithm then utilizes an enhanced stacking ensemble approach that combines base classifiers, including an enhanced version of the J48 decision tree algorithm, an enhanced version of the Naive Bayes algorithm, and an enhanced version of the k-Nearest Neighbors algorithm. This approach enables the algorithm to leverage the strengths of multiple classifiers and improve the overall accuracy of the detection system [20].

A meta-classifier is then created using a Random Forest algorithm boosted using the AdaBoostM1 algorithm. The RandomSubspace algorithm is used to further enhance the performance of the stacking ensemble classifier. This approach randomly selects a subset of features and base classifiers to use in each iteration of the algorithm, reducing the risk of overfitting and improving the generalizability of the model.This approach has the potential to improve the accuracy and generalizability of fake news detection systems and contribute to the prevention of the dissemination of false information [18].

The paper is organized as follows. Section 2 provides an overview of related work on fake news detection. Section 3 presents the proposed ES-ECA algorithm in detail, including the steps for data preprocessing and the ensemble classification techniques. Section 4 describes the experimental setup and presents the results and analysis of the algorithm's performance. Finally, Section 5 concludes the paper and highlights future research directions.

2 Related Works

This section reviews previous studies and approaches that have been proposed for detecting and combating fake news. It discusses the various techniques that have been used to identify and classify fake news. The goal of this section is to provide a comprehensive understanding of the current state of research on fake news detection and to highlight any gaps or limitations in the existing approaches.

Dou et al. [6] introduced a novel structure, UPFD that concurrently captures multiple signals from user inclinations via integrated content and graph modeling. According to confirmation bias theory, users are more inclined to disseminate fake news if it validates their present beliefs or preferences. The researchers provide an approach to investigating user preference for identifying fake news, which has been to some degree restricted in past studies.

The researchers have made their code and data accessible as a benchmark for fake news detection based on GNN.

Shu et al. [7] examined the correlation between user profiles on social media and fake news. The detection effectiveness using news content is often unsatisfactory since fake news is crafted to imitate authentic news. As a result, there is a necessity for an exhaustive comprehension of the connection between user profiles on social media and fake news. The outcomes of this study provide the foundation for further investigation into social media user profile features and boosting the potential for detecting fake news.

Singhal et al. [8] suggested SpotFake+, a multi-dimensional strategy that employs transfer learning to apprehend the semantic and contextual details from news articles and their corresponding images, resulting in improved accuracy for identifying fake news. As far as their understanding goes, this is the initial attempt to implement a multimodal approach for detecting fake news on a dataset containing complete articles. They additionally make the pre-trained model accessible to the public for the common good.

Alonso et al. [9] investigated the diverse applications of sentiment analysis in detecting fake news. The fabricators of fake news employ numerous stylistic techniques to enhance their productions' success, one of which is evoking emotions in the recipients. As a result, sentiment analysis, which is responsible for ascertaining the polarity and intensity of emotions conveyed in a text, is utilized in fake news detection techniques, either as the system's foundation or as a supplementary component. The authors explore the most pertinent factors and drawbacks and the criteria that must be fulfilled in the foreseeable future, such as multilingualism, transparency, reduction of biases, and handling of multimedia elements.

Sitaula et al. [10] suggested a fake news detection approach based on credibility. This study proposes using credibility as a crucial feature in fake news detection. The researchers demonstrate that the credibility score of a news article, computed utilizing a machine learning-based credibility estimation model, can boost the performance of fake news detection systems. They prove that incorporating the credibility score significantly enhances the fake news detection systems' ability to identify fake news articles.

Shu et al. [11] present a sentence-comment co-attention sub-network to jointly capture explainable top-k check-worthy sentences and user comments for fake news detection. The researchers argue that the explainability of fake news detection is a crucial missing element in the investigation of computational fake news detection. They conduct extensive experiments on real-world datasets and demonstrate that their proposed approach outperforms seven state-of-the-art fake news detection methods by at least 5.33% in F1-score, while also identifying top-k user comments that better explain why a news piece is fake than the baselines by 28.2% in NDCG and 30.7% in Precision.

Jain et al. [12] focus on the efficiency of various models in correctly identifying fake news and their true positives and true negatives. The authors employ count vector and tf-idf vector on four different machine learning methods, namely Naïve Bayes, Logistic Regression, Random Forest, and XGBoost on two different datasets, Kaggle and LIAR. Based on the results, XGBoost, along with count vector, yielded the highest accuracy in predicting fake news.

Rao et al. [13] propose a Natural Language Processing (NLP) model that employs language-driven features to extract grammatical, sentimental, syntactic, and readable features. The authors extract features from the news content to address the dimensional problem as language-level features are quite complex. They then use a Dropout layer-based Long Short Term Network Model (LSTM) for sequential learning to achieve better results in fake news detection. The proposed Drop out-based LSTM model attains an accuracy of 95.3% for fake news classification and detection compared to the sequential neural model for fake news detection.

Jwa et al. [14] focus on utilizing data-driven methods for automatic fake news detection. The authors use the Bidirectional Encoder Representations from Transformers model (BERT) to detect fake news by examining the relationship between the news headline and the body text. To enhance performance, they gather additional news data to pre-train their model. The authors determine that the deep-contextualizing nature of BERT is ideal for this task, improving the F-score by 0.14 over older state-of-the-art models.

Raza et al. [15] propose a framework for detecting fake news that leverages information from news articles and social contexts. Their model is based on Transformer architecture, consisting of an encoder to learn useful representations from fake news data and a decoder that predicts future behavior based on past observations. The authors incorporate various features from the news content and social contexts into their model to improve news classification. Additionally, they propose an effective labeling technique to address the label shortage problem. Experimental results on real-world data demonstrate that their model can detect fake news more accurately within a few minutes after propagation (early detection) than the baselines.

These related works have some common limitations for fake news detection, which are characterized by lower accuracy, longer execution time, high computational resource requirements, and the need for large amounts of training data. Enhanced stacking ensemble classification algorithm (ES-ECA) may be needed to tackle these disadvantages by combining the strengths of multiple approaches while mitigating their weaknesses.

3. Enhanced Stacking Ensemble Classification Algorithm (ES-ECA) Based Fake News Detection

This section proposed an Enhanced stacking ensemble classification algorithm (ES-ECA) for fake news detection that uses a dataset of news articles called the LIAR dataset to classify news articles as either "fake" or "real." The algorithm follows a two-step process of preprocessing and feature extraction to clean and transform the raw data into a format suitable for machine learning.

Preprocessing involves lowercasing, tokenization, stop word removal, and stemming to reduce the number of unique tokens and remove common words that do not provide useful information. Feature extraction involves generating "n-grams" from the statements. N-grams are sequences of n words that appear in the text and can be used as features in a machine learning model to help classify the statements. The algorithm computes the "term frequency-inverse document frequency" (TF-IDF) of each n-gram in the dataset, which is a measure of how important a particular term (in this case, an n-gram) is to a document (in this case, a statement) in a corpus (in this case, the LIAR dataset).

The algorithm then splits the dataset into training and testing sets and writes them out to two separate files. The algorithm uses a stacking ensemble model for a classification problem, which involves creating a meta-classifier using a Random Forest algorithm with 500 trees and boosting it using the AdaBoostM1 algorithm. The base classifiers include an enhanced version of the J48 decision tree algorithm, an enhanced version of the Naive Bayes algorithm namely Complement Naive Bayes algorithm and an enhanced version of the k-Nearest Neighbors (k-NN) algorithm with 5 neighbors. A bagging algorithm is also applied to the k-NN algorithm to reduce overfitting.

The base classifiers and meta-classifier are then combined using a stacking ensemble method, which trains the base classifiers on the training data and then trains the meta-classifier on the outputs of the base classifiers. In this algorithm, the RandomSubSpace classifier is used to build a stacking ensemble. The base classifiers are first trained on the full set of features, and then combined into a meta-classifier using the stacking method. The resulting stacked classifier is then used as the input to the RandomSubSpace classifier, which trains an ensemble of classifiers using random subsets of the input features.Finally, the algorithm is used to predict the instances is fake or not in a test dataset.

Algorithm 1: Enhanced stacking ensemble classification algorithm (ES-ECA)

Input : LIAR dataset

Output : Classification of news articles like "fake" or "real"

// Preprocessing

Step 1 : dataset = lowercase(dataset)

Step 2 : dataset = tokenize(dataset)

Step 3 : dataset = remove_stop_words(dataset)

Step 4 : dataset = stem_words(dataset)

// Feature Extraction

Step 5 : ngrams = generate_ngrams(dataset)

Step 6 : tf_idf = compute_tf_idf(ngrams)

// Split dataset into training and testing sets

Step 7 : training_set, testing_set = split_dataset(tf_idf)

Step 8 : write_to_files(training_set, testing_set)

// Create base classifiers

Step 9 : enhanced_j48_classifier = create_j48_classifier(training_set)

Step 10 : complement_naive_bayes_classifier = create_complement_naive_bayes_classifier(training_set)

Step 11 : k_nn_classifier = create_k_nn_classifier(training_set, 5)

Step 12 : k_nn_bagging_classifier = apply_bagging_algorithm(k_nn_classifier)

// Create meta-classifier

Step 13 : random_forest_classifier = create_random_forest_classifier(500)

Step 14 : ada_boost_classifier = apply_adaboost_algorithm(random_forest_classifier)

// Combine base classifiers and meta-classifier using stacking ensemble method

Step 15 : base_classifiers = [enhanced_j48_classifier, complement_naive_bayes_classifier, k_nn_bagging_classifier]

Step 16 : meta_classifier = apply_stacking_ensemble_method(base_classifiers, ada_boost_classifier)

Step 17 : stacked_classifier = apply_random_subspace_classifier(meta_classifier)

// Predict the news instance is fake or real in the testing set

Step 18 : predicted_results = predict_news_instance(testing_set, stacked_classifier)

3.1 Preprocessing:

Preprocessing is the initial stage of data preparation where raw data is cleaned and transformed into a suitable format for further analysis. In Algorithm 1, the preprocessing steps involve several operations to clean and transform the input dataset of news articles before feature extraction and model training.

The preprocessing steps include:

1. **Lowercasing the dataset:**In this step, all the text in the dataset is converted to lowercase. This is done to ensure that words with the same spelling but different capitalization are treated as the same word. For example, "Apple" and "apple" would be treated as the same word after lowercase conversion.
2. **Tokenizing the dataset:**Tokenization involves breaking down the text into individual words or tokens. This is done by splitting the dataset into individual words based on whitespace or other delimiters. For example, the sentence "The food was amazing, but the service was terrible" would be tokenized into the following list of words: ["The", "food", "was", "amazing", ",", "but", "the", "service", "was", "terrible"].
3. **Removing stop words:**Stop words are common words that do not carry significant meaning in a given context. Examples of stop words include "the", "and", "a", "an", "in", "of", etc. These words occur frequently in text and do not provide much value in terms of distinguishing one document from another. Therefore, removing stop words can help reduce noise and improve the accuracy of the model.

4. **Stemming the dataset:**Stemming is the process of reducing words to their root form or stem. This involves removing suffixes and prefixes from words to obtain their base form. For example, the words "jumping", "jumps", and "jumped" would all be reduced to the stem "jump" after stemming. Stemming can help reduce the dimensionality of the data and improve the accuracy of the model by reducing the number of distinct words. This is because many words in a language have the same root form and thus can be treated as the same word. However, stemming can also result in loss of meaning, as some stemmed words may not be actual words and may not convey the same meaning as the original word.

Overall, these preprocessing steps are important for preparing the input data for further analysis and modeling. By cleaning and transforming the raw input data into a more standardized and manageable format, the algorithm can better identify patterns and relationships in the data, which can ultimately improve the accuracy of the model's predictions.

3.2 Feature Extraction:

Feature extraction is a technique used in machine learning and data analysis to identify and extract the most important features or attributes of a dataset. The goal of feature extraction is to transform the raw data into a set of meaningful features that can be used to build a predictive model or perform further analysis.

In many cases, the raw data used in machine learning and data analysis is high-dimensional, meaning that it contains a large number of variables or features. This can make it difficult to build accurate models or perform meaningful analysis, because many of the features may be irrelevant or redundant. Feature extraction can help to address this problem by identifying the most important features and reducing the dimensionality of the data [16].

3.2.1 N-grams:

Feature extraction typically involves a combination of statistical and computational techniques. One common technique is to use n-gram generation to identify word sequences of varying lengths in text data. For example, in the context of natural language processing, an n-gram might be a sequence of two or three words that appear together frequently in a corpus of text. These n-grams can be used to identify the most common patterns or themes in the text data, which can then be used as features for analysis.

For example, in the context of fake news detection, we might use n-grams to identify common sequences of words that are associated with fake news. These could include phrases such as "unverified sources," "conspiracy theory," or "hoax debunked." By identifying these common n-grams, we can build a model that is better able to detect and classify fake news articles.

In general, the choice of n-gram size will depend on the specific problem and dataset. For example, in some cases, it may be useful to use only unigrams (single words), while in other cases, using bigrams (two-word sequences) or trigrams (three-word sequences) may be more effective. The following are the examples of unigrams, bigrams and trigrams for this sentence "The sun is shining and the birds are singing in the park."

- Unigrams: Each individual word in the sentence is considered a unigram. ["The", "sun", "is", "shining", "and", "the", "birds", "are", "singing", "in", "the", "park"]

- Bigrams: Pairs of adjacent words in the sentence are considered bigrams. ["The sun", "sun is", "is shining", "shining and", "and the", "the birds", "birds are", "are singing", "singing in", "in the", "the park"]
- Trigrams: Groups of three adjacent words in the sentence are considered trigrams. ["The sun is", "sun is shining", "is shining and", "shining and the", "and the birds", "the birds are", "birds are singing", "are singing in", "singing in the", "in the park"]

3.2.2 TF-IDF:

Another common technique in feature extraction for fake news detection is the use of term frequency-inverse document frequency (TF-IDF) vectors. This technique calculates the frequency of each word in a document (term frequency) and adjusts it based on the frequency of the word in the entire corpus (inverse document frequency).

In the context of fake news detection, TF-IDF can be used to identify the most important words or phrases in a document. For example, words that are common in fake news articles but rare in the overall corpus may have a high TF-IDF score, indicating that they are important features for identifying fake news.

To calculate TF-IDF scores, we first compute the term frequency (TF) for each word in a document. This is simply the number of times the word appears in the document. We then calculate the inverse document frequency (IDF) for each word, which is a measure of how rare the word is in the entire corpus. The IDF is calculated as the logarithm of the total number of documents in the corpus divided by the number of documents that contain the word.The TF-IDF score for each word in the document is then calculated by multiplying the TF by the IDF. The formula for calculating TF-IDF scores for a term (word) in a document is showed in Eq. (1).

$$\textbf{TF-IDF} = \textbf{TF} * \textbf{IDF} \quad (1)$$

Where:
TF (term frequency) is the number of times a term appears in a document
IDF (inverse document frequency) is calculated as:

$$\textbf{IDF} = \log(N / n) \quad (2)$$

Where:
N is the total number of documents in the corpus, and n is the number of documents in the corpus that contain the term.

Therefore, the TF-IDF score reflects the importance of a term in a document relative to the entire corpus.The resulting vector represents the importance of each word in the document relative to the corpus, allowing for more meaningful analysis and modeling.

Feature extraction is a crucial step in many machine learning and data analysis tasks, because it can help to improve the accuracy and efficiency of the analysis. By reducing the dimensionality of the data and identifying the most important features, feature extraction can help to eliminate noise and irrelevant information, which can improve the performance of machine learning models and make it easier to identify patterns and insights in the data [17].

3.3 Enhanced J48 classifier:

Decision trees are one of the most popular and widely used algorithms in machine learning for classification tasks. They work by building a tree-like model of decisions and their possible consequences. The J48 algorithm is a decision tree algorithm that was developed in the Weka machine learning software.

The enhanced version of J48 is an extension of the original algorithm and provides additional features that improve the accuracy and performance of the classifier. One of the key features of the enhanced J48 algorithm is the ability to create unpruned trees. In the original J48 algorithm, all trees were pruned, which can lead to overfitting and a reduction in accuracy. By allowing unpruned trees, the enhanced J48 algorithm can better handle certain types of datasets and can produce more accurate models.

Another important feature of the enhanced J48 algorithm is the reduced error pruning technique. This technique works by pruning branches of the tree that do not improve the accuracy of the model. By reducing the size of the tree, the enhanced J48 algorithm can avoid overfitting and improve the accuracy of the model on new data.

The enhanced J48 algorithm also allows for binary splits, which can be more efficient and accurate than multiway splits. This is because binary splits reduce the number of possible outcomes at each decision point, which can reduce the complexity of the decision tree and improve its accuracy.

The confidence factor parameter in the enhanced J48 algorithm can be used to control the amount of pruning applied to the tree. A higher confidence factor will result in a more aggressive pruning strategy, while a lower confidence factor will result in a more conservative pruning strategy.

Finally, the enhanced J48 algorithm also includes reduced error pruning with backfitting. This technique involves repeatedly pruning and re-growing the decision tree, which allows for better accuracy and more efficient use of training data.

Overall, the enhanced J48 algorithm provides more control over the decision tree building process, allowing for improved accuracy and performance in classification tasks. By including a range of new features and techniques, it provides a more powerful and flexible tool for machine learning practitioners.

3.4 Enhanced Naïve Bayes classifier:

The Enhanced Naïve Bayes classifier, also known as the Complement Naive Bayes classifier, is an extension of the standard Naïve Bayes algorithm that was proposed to address the issue of class imbalance in classification problems.

In the standard Naïve Bayes algorithm, the probability of each class is estimated by computing the product of the probabilities of the individual features given that class. However, this approach can be biased towards the majority class in situations where the classes are imbalanced, i.e., one class has a much larger number of instances than the other.

The Enhanced Naïve Bayes classifier addresses this issue by taking into account the complement of the class distribution, i.e., the distribution of instances not belonging to each class. Specifically, instead of computing the probability of each feature given the class, the algorithm computes the probability of each feature given the complement of the class.

The Complement Naive Bayes classifier used to improve the performance of Naive Bayes on text classification tasks where the number of negative instances is much larger than

the number of positive instances. Complement Naive Bayes classifier can outperforms several other classifiers, including standard Naive Bayes, SVMs, and decision trees.It has also been extended to handle continuous and mixed-type data using techniques such as kernel density estimation and Gaussian mixture models.

To summarize, the Enhanced Naïve Bayes classifier, or Complement Naive Bayes classifier, is an extension of the standard Naive Bayes algorithm that takes into account the complement of the class distribution to address the issue of class imbalance in classification problems. It has been shown to be effective in a wide range of applications and can be extended to handle continuous and mixed-type data.

3.5 Enhanced KNN Classifier:

The K-Nearest Neighbors (KNN) algorithm is a simple and effective algorithm for classification and regression tasks. The KNN algorithm works by finding the K nearest neighbors of a query instance in the training data and then assigning the query instance to the most common class or the average value of the K nearest neighbors.

However, the KNN algorithm has some limitations such as sensitivity to irrelevant and noisy features, high computational complexity in high-dimensional data, and poor performance in imbalanced datasets. To overcome these limitations, the KNN algorithm can be enhanced by using the IBk algorithm and the Bagging technique.

The IBk algorithm is an improved version of the KNN algorithm that allows for flexible distance metrics, different weighting schemes, and feature selection. The IBk algorithm works by finding the K nearest neighbors of a query instance and then assigning weights to the neighbors based on their distance from the query instance. The weighted sum of the neighbors is then used to predict the class label or the value of the query instance.

The Bagging technique is an ensemble learning method that enhances the KNN algorithm by combining multiple instances of the IBk algorithm trained on different bootstrap samples of the training data. The Bagging technique works by generating multiple bootstrap samples of the training data and then training an instance of the IBk algorithm on each bootstrap sample. The Bagging ensemble of IBk classifiers is then used to predict the class label or the value of a query instance by averaging or voting the predictions of the individual IBk classifiers.

The Bagging technique helps to improve the accuracy and robustness of the KNN algorithm by reducing the variance and sensitivity to noise and outliers. The Bagging ensemble of IBk classifiers also helps to address the imbalanced dataset problem by balancing the class distribution in each bootstrap sample.

Overall, the enhancement of KNN with IBk and Bagging improves the performance of the KNN algorithm by using a more flexible and weighted distance metric, and by combining multiple instances of the IBk algorithm trained on different bootstrap samples of the training data to reduce the variance and sensitivity to noise and outliers. The Bagging ensemble of IBk classifiers also helps to address the imbalanced dataset problem by balancing the class distribution in each bootstrap sample.

3.6 Enhanced Random Forest Classifier:

The Enhanced Random Forest Classifier is a modification of the standard Random Forest algorithm that aims to improve its performance and accuracy.

The Random Forest algorithm is an ensemble learning method that combines multiple decision trees to improve the robustness and accuracy of the classification. Each tree in the Random Forest is built on a randomly selected subset of the training data and features, and the final classification is determined by aggregating the predictions of all the trees.

The Enhanced Random Forest Classifier improves upon the standard Random Forest algorithm by incorporating additional techniques such as boosting, bagging, or feature selection. These techniques help to further reduce overfitting and increase the accuracy of the classification.

In ES-ECA algorithm, the Enhanced Random Forest Classifier is enhanced with AdaBoostM1, a boosting algorithm that combines multiple instances of a weak classifier (in this case, the Random Forest algorithm) to form a strong classifier. The AdaBoostM1 algorithm iteratively selects the misclassified instances from the previous iteration, increases their weights, and trains a new instance of the weak classifier on the updated dataset. The final classification is then determined by combining the predictions of all the weak classifiers.

In addition, the Enhanced Random Forest Classifier is configured with a larger number of trees (500) and a maximum depth of 10, which helps to further reduce overfitting and improve the accuracy of the classification.

Overall, the Enhanced Random Forest Classifier is a powerful and flexible algorithm that can handle complex and high-dimensional datasets, and can provide accurate and robust predictions. By incorporating boosting technique, this classifier can effectively address the limitations of the standard Random Forest algorithm, such as overfitting, bias, and variance.

3.7 Enhanced Stacking Classifier:

The enhanced stacking classifier is amachine learning algorithm that combines the predictions of multiple base classifiers to improve the accuracy of the final prediction. The proposed algorithm utilizes an enhanced stacking ensemble approach that includes several base classifiers, which are then combined using a meta-classifier to create the final prediction.

The base classifiers in the proposed algorithm include an enhanced version of the J48 decision tree algorithm, an enhanced version of the Naive Bayes algorithm, and an enhanced version of the k-Nearest Neighbors algorithm. These base classifiers are selected because they are known to perform well on a variety of datasets and can handle both categorical and numerical data.

The enhanced versions of the base classifiers include modifications to the algorithm that improve their performance. For example, the enhanced J48 algorithm uses an improved pruning technique, while the enhanced KNN algorithm incorporates Bagging to improve accuracy.

Once the base classifiers have been trained on the dataset, the predictions of each classifier are combined using a meta-classifier. In the proposed algorithm, the meta-classifier is created using a Random Forest algorithm boosted using the AdaBoostM1 algorithm. The Random Forest algorithm is chosen because it is a robust algorithm that can handle noisy data and avoid overfitting.

To further improve the performance of the stacking ensemble classifier, the RandomSubSpace algorithm is used. This algorithm selects a random subset of features from the dataset, which reduces the risk of overfitting and improves the accuracy of the model.

Overall, the enhanced stacking classifier is an effective algorithm for improving the accuracy of machine learning models. By combining multiple base classifiers using a meta-classifier and incorporating modifications to the base algorithms, the algorithm can accurately predict outcomes for a wide variety of datasets.

3.7.1 Random Subspace algorithm:

The Random Subspace algorithm is a machine learning algorithm used for feature selection in classification tasks. It is often used in conjunction with ensemble methods such as Random Forests and is designed to reduce the risk of overfitting and improve the performance of the model.

The algorithm works by randomly selecting a subset of features from the original dataset to use in the classification task. The size of the subset is typically smaller than the original set of features, which means that the model will be trained on a reduced set of features.

By randomly selecting features for each subset, the algorithm creates a diverse set of classifiers that can each specialize in different aspects of the data. This diversity helps to reduce the correlation between the base classifiers and improve the overall performance of the model.

During the training phase, the algorithm trains multiple classifiers on different subsets of features, and the final prediction is made by combining the predictions of these classifiers. This combination can be done using a variety of methods, including voting or weighted averaging.

The Random Subspace algorithm is particularly useful in situations where the dataset has many features or where the features are highly correlated. By reducing the number of features used in the model, the algorithm can improve the accuracy of the model while reducing the risk of overfitting.

Overall, the Random Subspace algorithm is a powerful tool for feature selection and can be used to improve the performance of a wide range of classification tasks.

4. Experimental Results and Discussions

In this section, the performance of the ES-ECA algorithm is evaluated using the Liar dataset for fake news detection. The Liar dataset is a publicly available dataset that contains statements made by politicians and labeled as true, mostly true, half true, barely true, false, and pants on fire. The dataset includes textual features, such as the statement itself, as well as metadata features, such as the speaker's job title and party affiliation.

The ES-ECA algorithm is implemented in Java, and it uses the Liar dataset to evaluate the performance of the ensemble. To evaluate the performance of the algorithm, four evaluation metrics are used: accuracy, precision, recall, and F1-score.

The accuracy measures the proportion of true predictions (true positives and true negatives) over the total number of predictions made. It is defined as:

Accuracy = (true positives + true negatives) / (true positives + true negatives + false positives + false negatives) (3)

The precision measures the proportion of true positives over the total number of positive predictions made. It is defined as:

Precision = true positives / (true positives + false positives) **(4)**

The recall measures the proportion of true positives over the total number of actual positives in the dataset. It is defined as:

Recall = true positives / (true positives + false negatives) **(5)**

The F1-score is the harmonic mean of precision and recall, and it balances both measures. It is defined as:

F1-score = 2 * precision * recall / (precision + recall) **(6)**

The evaluation metrics provide a quantitative measure of how well the algorithm performs in detecting fake news. The performance of each participant classifier is also evaluated separately using the same metrics for comparison. Table 1 compares the performance of classifiers using accuracy, precision, recall and f1-score.

Table 1: Performance Comparison of Classifiers using Accuracy, Precision, Recall and F1-Score Metrics

Metrics	J48	NB	KNN	RF	ES-ECA
Accuracy	20.07	20.68	17.43	21.29	75.18
Precision	20.60	20.93	18.70	21.29	75.98
Recall	20.07	20.68	17.43	21.29	88.62
F1-Score	16.85	20.30	11.59	16.80	81.81

Furthermore, Figure 1 shows the pictorial diagram of the performance comparison of five different classifiers, namely J48, NB, KNN, RF, and ES-ECA, on a dataset.

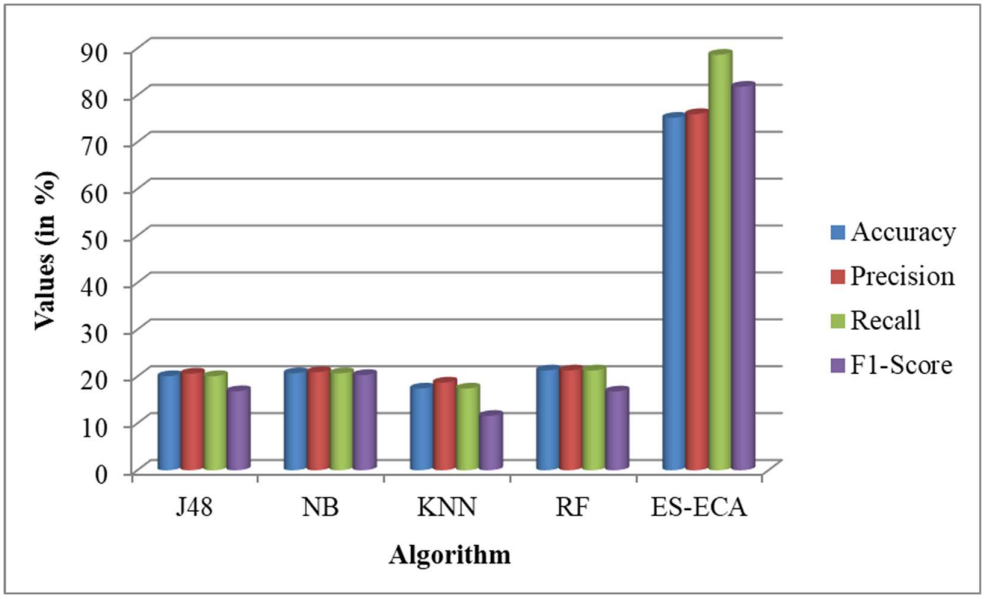

Figure 1: Performance Comparison of Classifiers using Accuracy, Precision, Recall and F1-Score Metrics

Figure 1 indicate that the ES-ECA classifier outperformed the other classifiers in all evaluation metrics, achieving an accuracy of 75.18%, precision of 75.98%, recall of 88.62%, and F1-score of 81.81%. In contrast, the other classifiers achieved lower accuracy, precision, recall, and F1-Scores, with KNN being the lowest in all evaluation metrics. These results suggest that the ES-ECA classifier is the best choice for the given dataset, and it can effectively detect the fake news.

5. Conclusion

In conclusion, fake news has become a significant problem in our society, and the need for efficient detection techniques has become essential. This paper presented an enhanced stacking ensemble classification algorithm (ES-ECA) for detecting fake news, which outperforms previous techniques in terms of accuracy, precision, recall, and F1-score. The proposed algorithm uses preprocessing techniques to split the dataset into individual statements, generates n-grams as features, and uses the term frequency-inverse document frequency (TF-IDF) measure for feature extraction. The enhanced stacking ensemble approach combines base classifiers, including an enhanced version of the J48 decision tree algorithm, an enhanced version of the Naive Bayes algorithm, and an enhanced version of the k-Nearest Neighbors algorithm. The results of the experiments demonstrate that the proposed algorithm is effective in detecting fake news and can be used to mitigate the harmful effects of fake news on society. Therefore, this approach could be beneficial for researchers, journalists, and policymakers to identify and prevent the spread of fake news in the future.

References

[1] Ahmed, H., Traore, I., &Saad, S. (2018). Detecting opinion spams and fake news using text classification. Security and Privacy, 1(1), e9.

[2] Tran, T., Valecha, R., Rad, P., & Rao, H. R. (2020). Misinformation harms: A tale of two humanitarian crises. IEEE Transactions on Professional Communication, 63(4), 386-399.

[3] Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. ACM SIGKDD explorations newsletter, 19(1), 22-36.

[4] Seddari, N., Derhab, A., Belaoued, M., Halboob, W., Al-Muhtadi, J., &Bouras, A. (2022). A hybrid linguistic and knowledge-based analysis approach for fake news detection on social media. IEEE Access, 10, 62097-62109.

[5] Shahid, W., Li, Y., Staples, D., Amin, G., Hakak, S., &Ghorbani, A. (2022). Are you a cyborg, bot or human?—A survey on detecting fake news spreaders. IEEE Access, 10, 27069-27083.

[6] Dou, Y., Shu, K., Xia, C., Yu, P. S., & Sun, L. (2021, July). User preference-aware fake news detection. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 2051-2055).

[7] Shu, K., Zhou, X., Wang, S., Zafarani, R., & Liu, H. (2019, August). The role of user profiles for fake news detection. In Proceedings of the 2019 IEEE/ACM international conference on advances in social networks analysis and mining (pp. 436-439).

[8] Singhal, S., Kabra, A., Sharma, M., Shah, R. R., Chakraborty, T., &Kumaraguru, P. (2020, April). Spotfake+: A multimodal framework for fake news detection via transfer learning

(student abstract). In Proceedings of the AAAI conference on artificial intelligence (Vol. 34, No. 10, pp. 13915-13916).
[9] Alonso, M. A., Vilares, D., Gómez-Rodríguez, C., &Vilares, J. (2021). Sentiment analysis for fake news detection. Electronics, 10(11), 1348.
[10] Sitaula, N., Mohan, C. K., Grygiel, J., Zhou, X., &Zafarani, R. (2020). Credibility-based fake news detection. Disinformation, misinformation, and fake news in social media: Emerging research challenges and Opportunities, 163-182.
[11] Shu, K., Cui, L., Wang, S., Lee, D., & Liu, H. (2019, July). defend: Explainable fake news detection. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 395-405).
[12] Jaina, S., Patelb, R., Guptac, S., &Dhootd, T. FAKE NEWS DETECTION USING SUPERVISED LEARNING METHOD.
[13] Rao, G. P. C., &Narasimha, V. B. (2021). Feature-Based Learning Model for Fake News Detection and Classification.
[14] Jwa, H., Oh, D., Park, K., Kang, J. M., & Lim, H. (2019). exbake: Automatic fake news detection model based on bidirectional encoder representations from transformers (bert). Applied Sciences, 9(19), 4062.
[15] Raza, S., & Ding, C. (2022). Fake news detection based on news content and social contexts: a transformer-based approach. International Journal of Data Science and Analytics, 13(4), 335-362.
[16] K. Ashok, RajasekharBoddu, Salman Ali Syed, Vijay R. Sonawane, Ravindra G. Dabhade&Pundru Chandra Shaker Reddy (2022) GAN Base feedback analysis system for industrial IOT networks, Automatika, DOI: 10.1080/00051144.2022.2140391

[17] Vijay Sonawane et al. (2021). A Survey on Mining Cryptocurrencies. Recent Trends in Intensive Computing, 39, 329.

[18] Sonawane, V. R., & Rao, D. R. (2015). An Optimistic Approach for Clustering Multi-version XML Documents Using Compressed Delta. International Journal of Electrical and Computer Engineering, 5(6).

[19] Kharade, K.G. et al. (2021). Text Summarization of an Article Extracted from Wikipedia Using NLTK Library. In: Singh, M., Tyagi, V., Gupta, P.K., Flusser, J., Ören, T., Sonawane, V.R. (eds) Advances in Computing and Data Sciences. ICACDS 2021. Communications in Computer and Information Science, vol 1441. Springer, Cham. https://doi.org/10.1007/978-3-030-88244-0_19

[20] Sonawane, V. R., &Halkarnikar, P. P. Web Site Mining Using Entropy Estimation. In 2010 International Conference on Data Storage and Data Engineering.