Hindawi
Complexity
Volume 2020, Article ID 8885861, 11 pages
https://doi.org/10.1155/2020/8885861

Research Article

Fake News Detection Using Machine Learning Ensemble Methods

Iftikhar Ahmad,[1] **Muhammad Yousaf,**[1] **Suhail Yousaf**,[1]
and Muhammad Ovais Ahmad[2]

[1]*Department of Computer Science and Information Technology, University of Engineering and Technology, Peshawar, Pakistan*
[2]*Department of Mathematics and Computer Science, Karlstad University, Karlstad, Sweden*

Correspondence should be addressed to Muhammad Ovais Ahmad; ovais.ahmad@kau.se

Received 4 September 2020; Revised 14 September 2020; Accepted 16 September 2020; Published 17 October 2020

Academic Editor: M. Irfan Uddin

The advent of the World Wide Web and the rapid adoption of social media platforms (such as Facebook and Twitter) paved the way for information dissemination that has never been witnessed in the human history before. With the current usage of social media platforms, consumers are creating and sharing more information than ever before, some of which are misleading with no relevance to reality. Automated classification of a text article as misinformation or disinformation is a challenging task. Even an expert in a particular domain has to explore multiple aspects before giving a verdict on the truthfulness of an article. In this work, we propose to use machine learning ensemble approach for automated classification of news articles. Our study explores different textual properties that can be used to distinguish fake contents from real. By using those properties, we train a combination of different machine learning algorithms using various ensemble methods and evaluate their performance on 4 real world datasets. Experimental evaluation confirms the superior performance of our proposed ensemble learner approach in comparison to individual learners.

1. Introduction

The advent of the World Wide Web and the rapid adoption of social media platforms (such as Facebook and Twitter) paved the way for information dissemination that has never been witnessed in the human history before. Besides other use cases, news outlets benefitted from the widespread use of social media platforms by providing updated news in near real time to its subscribers. The news media evolved from newspapers, tabloids, and magazines to a digital form such as online news platforms, blogs, social media feeds, and other digital media formats [1]. It became easier for consumers to acquire the latest news at their fingertips. Facebook referrals account for 70% of traffic to news websites [2]. These social media platforms in their current state are extremely powerful and useful for their ability to allow users to discuss and share ideas and debate over issues such as democracy, education, and health. However, such platforms are also used with a negative perspective by certain entities commonly for monetary gain [3, 4] and in other cases for creating biased opinions, manipulating mindsets, and spreading satire or absurdity. The phenomenon is commonly known as fake news.

There has been a rapid increase in the spread of fake news in the last decade, most prominently observed in the 2016 US elections [5]. Such proliferation of sharing articles online that do not conform to facts has led to many problems not just limited to politics but covering various other domains such as sports, health, and also science [3]. One such area affected by fake news is the financial markets [6], where a rumor can have disastrous consequences and may bring the market to a halt.

Our ability to take a decision relies mostly on the type of information we consume; our world view is shaped on the basis of information we digest. There is increasing evidence that consumers have reacted absurdly to news that later proved to be fake [7, 8]. One recent case is the spread of novel corona virus, where fake reports spread over the Internet about the origin, nature, and behavior of the virus [9]. The situation worsened as more people read about the fake contents online. Identifying such news online is a daunting task.

Fortunately, there are a number of computational techniques that can be used to mark certain articles as fake on the basis of their textual content [10]. Majority of these techniques use fact checking websites such as “PolitiFact” and “Snopes.” There are a number of repositories maintained by researchers that contain lists of websites that are identified as ambiguous and fake [11]. However, the problem with these resources is that human expertise is required to identify articles/websites as fake. More importantly, the fact checking websites contain articles from particular domains such as politics and are not generalized to identify fake news articles from multiple domains such as entertainment, sports, and technology.

The World Wide Web contains data in diverse formats such as documents, videos, and audios. News published online in an unstructured format (such as news, articles, videos, and audios) is relatively difficult to detect and classify as this strictly requires human expertise. However, computational techniques such as natural language processing (NLP) can be used to detect anomalies that separate a text article that is deceptive in nature from articles that are based on facts [12]. Other techniques involve the analysis of propagation of fake news in contrast with real news [13]. More specifically, the approach analyzes how a fake news article propagates differently on a network relative to a true article. The response that an article gets can be differentiated at a theoretical level to classify the article as real or fake. A more hybrid approach can also be used to analyze the social response of an article along with exploring the textual features to examine whether an article is deceptive in nature or not.

A number of studies have primarily focused on detection and classification of fake news on social media platforms such as Facebook and Twitter [13, 14]. At conceptual level, fake news has been classified into different types; the knowledge is then expanded to generalize machine learning (ML) models for multiple domains [10, 15, 16]. The study by Ahmed et al. [17] included extracting linguistic features such as n-grams from textual articles and training multiple ML models including K-nearest neighbor (KNN), support vector machine (SVM), logistic regression (LR), linear support vector machine (LSVM), decision tree (DT), and stochastic gradient descent (SGD), achieving the highest accuracy (92%) with SVM and logistic regression. According to the research, as the number of *n* increased in *n*-grams calculated for a particular article, the overall accuracy decreased. The phenomenon has been observed for learning models that are used for classification. Shu et al. [12] achieved better accuracies with different models by combining textual features with auxiliary information such as user social engagements on social media. The authors also discussed the social and psychological theories and how they can be used to detect false information online. Further, the authors discussed different data mining algorithms for model constructions and techniques shared for features extraction. These models are based on knowledge such as writing style, and social context such as stance and propagation.

A different approach is followed by Wang [18]. The author used textual features and metadata for training various ML models. The author focused mainly on using convolutional neural network (CNN). A convolutional layer is used to capture the dependency between the metadata vectors, followed by a bidirectional LSTM layer. The max-pooled text representations were concatenated with the metadata representation from the bidirectional LSTM, which was fed to fully connected layer with a softmax activation function to generate the final prediction. The research is conducted on a dataset from political domain which contains statements from two different parties. Along with that, some metadata such as subject, speaker, job, state, party, context, and history are also included as a feature set. Accuracy of 27.7% was achieved with combination of features such as text and speaker, whereas 27.4% accuracy was achieved by combining all the different metadata elements with text. A competitive solution is provided by Riedel et al. [19], which is a stance detection system that assigns one of four labels to an article, “agree,” “disagree,” “discuss,” or “unrelated,” depending on the conformity of article headline with article text. The authors used linguistic properties of text such as term frequency (TF) and term frequency-inverse document frequency (TF-IDF) as a feature set, and a multilayer perceptron (MLP) classifier is used with one hidden layer and a softmax function on the output of the final layer. The dataset contained articles with a headline, body, and label. The system’s accuracy on the “disagree” label on test examples was poor, whereas it performs best with respect to the “agree” label. The authors used a simple MLP with some fine-tuned hyperparameters to achieve an overall accuracy of 88.46%. Shu et al. [12] also discussed several varieties of veracity assessment methods to detect fake news online. Two major categories of assessment methods are explored: one is linguistic cue approaches and the other is network analyses approaches. A combination of both creates a more robust hybrid approach for fake news detection online. Linguistic approaches involve deep syntax, rhetorical structure, and discourse analysis. These linguistic approaches are used to train classifiers such as SVM or naïve Bayes models. Network-based approaches included analyzing and processing social network behavior and linked data. A unique approach is followed by Vosoughi et al. [13] to explore the properties of news spread on social media; i.e., the authors discussed the spread of news (rumors) on social media such as Twitter and analyzed how the spread of fake news differs from real news in terms of its diffusion on Twitter. Multiple analysis techniques are discussed in the paper to explore the spread of fake news online, such as the depth, the size, the maximum breadth, the structural virality, the mean breadth of true and false rumor cascades at various depths, the number of unique Twitter users reached at any depth, and the number of minutes it takes for true and false rumor cascades to reach depth and number of Twitter users.

1.1. Our Contributions.

In the current fake news corpus, there have been multiple instances where both supervised and unsupervised learning algorithms are used to classify text [20, 21]. However, most of the literature focuses on specific datasets or domains, most prominently the politics domain [10, 19, 21]. Therefore, the algorithm trained works best on a particular type of article's domain and does not achieve optimal results when exposed to articles from other domains. Since articles from different domains have a unique textual structure, it is difficult to train a generic algorithm that works best on all particular news domains. In this paper, we propose a solution to the fake news detection problem using the machine learning ensemble approach. Our study explores different textual properties that could be used to distinguish fake contents from real. By using those properties, we train a combination of different machine learning algorithms using various ensemble methods that are not thoroughly explored in the current literature. The ensemble learners have proven to be useful in a wide variety of applications, as the learning models have the tendency to reduce error rate by using techniques such as bagging and boosting [22]. These techniques facilitate the training of different machine learning algorithms in an effective and efficient manner. We also conducted extensive experiments on 4 real world publicly available datasets. The results validate the improved performance of our proposed technique using the 4 commonly used performance metrics (namely, accuracy, precision, recall, and F-1 score).

2. Materials and Methods

In the following, we describe our proposed framework, followed by the description of algorithms, datasets, and performance evaluation metrics.

2.1. Proposed Framework. In our proposed framework, as illustrated in Figure 1, we are expanding on the current literature by introducing ensemble techniques with various linguistic feature sets to classify news articles from multiple domains as true or fake. The ensemble techniques along with Linguistic Inquiry and Word Count (LIWC) feature set used in this research are the novelty of our proposed approach.

There are numerous reputed websites that post legitimate news contents, and a few other websites such as PolitiFact and Snopes which are used for fact checking. In addition, there are open repositories which are maintained by researchers [11] to keep an up-to-date list of currently available datasets and hyperlinks to potential fact checking sites that may help in countering false news spread. However, we selected three datasets for our experiments which contain news from multiple domains (such as politics, entertainment, technology, and sports) and contain a mix of both truthful and fake articles. The datasets are available online and are extracted from the World Wide Web. The first dataset is ISOT Fake News Dataset [23]; the second and third datasets are publicly available at Kaggle [24, 25]. A detailed description of the datasets is provided in Section 2.5.

The corpus collected from the World Wide Web is preprocessed before being used as an input for training the models. The articles' unwanted variables such as authors, date posted, URL, and category are filtered out. Articles with no body text or having less than 20 words in the article body are also removed. Multicolumn articles are transformed into single column articles for uniformity of format and structure. These operations are performed on all the datasets to achieve consistency of format and structure.

Once the relevant attributes are selected after the data cleaning and exploration phase, the next step involves extraction of the linguistic features. Linguistic features involved certain textual characteristics converted into a numerical form such that they can be used as an input for the training models. These features include percentage of words implying positive or negative emotions; percentage of stop words; punctuation; function words; informal language; and percentage of certain grammar used in sentences such as adjectives, preposition, and verbs. To accomplish the extraction of features from the corpus, we used the LIWC2015 tool which classifies the text into different discrete and continuous variables, some of which are mentioned above. LIWC tool extracts 93 different features from any given text. As all of the features extracted using the tool are numerical values, no encoding is required for categorical variables. However, scaling is employed to ensure that various feature's values lie in the range of (0, 1). This is necessary as some values are in the range of 0 to 100 (such as percentage values), whereas other values have arbitrary range (such as word counts). The input features are then used to train the different machine learning models. Each dataset is divided into training and testing set with a 70/30 split, respectively. The articles are shuffled to ensure a fair allocation of fake and true articles in training and tests instances.

The learning algorithms are trained with different hyperparameters to achieve maximum accuracy for a given dataset, with an optimal balance between variance and bias. Each model is trained multiple times with a set of different parameters using a grid search to optimize the model for the best outcome. Using a grid search to find the best parameters is computationally expensive [26]; however, the measure is taken to ensure the models do not overfit or underfit the data.

Novel to this research, various ensemble techniques such as bagging, boosting, and voting classifier are explored to evaluate the performance over the multiple datasets. We used two different voting classifiers composed of three learning models: the first voting classifier is an ensemble of logistic regression, random forest, and KNN, whereas the second voting classifier consists of logistic regression, linear SVM, and classification and regression trees (CART). The criteria used for training the voting classifiers is to train individual models with the best parameters and then test the model based on the selection of the output label on the basis of major votes by all three models. We have trained a bagging ensemble consisting of 100 decision trees, whereas two boosting ensemble algorithms are used, XGBoost and AdaBoost. A k-fold ($k = 10$) cross validation model is employed for all ensemble learners. The learning models used are described in detail in Section 2.2. To evaluate the performance of each model, we used accuracy, precision, recall, and F1 score metrics as discussed in Section 2.6.

2.2. Algorithms. We used the following learning algorithms in conjunction with our proposed methodology to evaluate the performance of fake news detection classifiers.

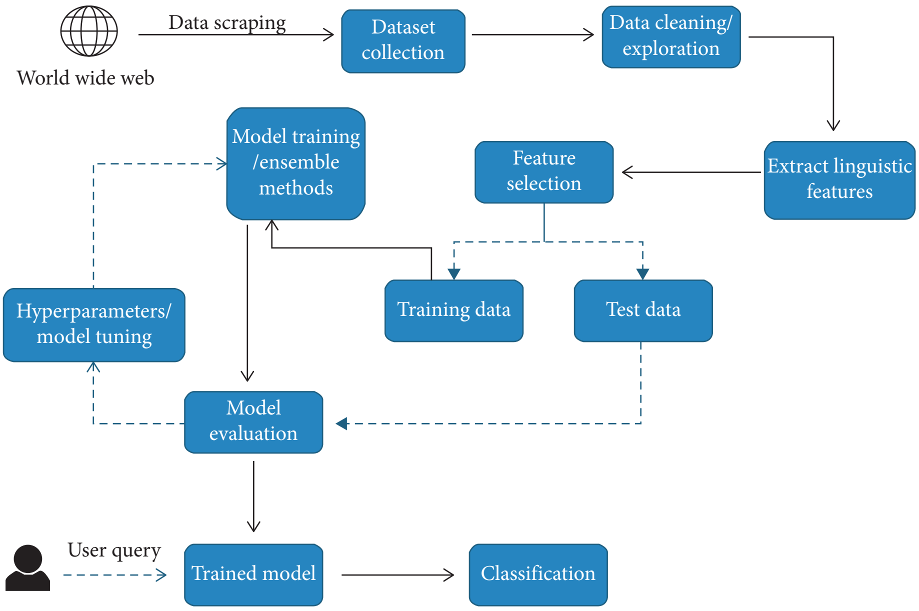

Figure 1: Workflow for training algorithms and classification of news articles.

2.2.1. Logistic Regression. As we are classifying text on the basis of a wide feature set, with a binary output (true/false or true article/fake article), a logistic regression (LR) model is used, since it provides the intuitive equation to classify problems into binary or multiple classes [27]. We performed hyperparameters tuning to get the best result for all individual datasets, while multiple parameters are tested before acquiring the maximum accuracies from LR model. Mathematically, the logistic regression hypothesis function can be defined as follows [27]:

$$h_\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}. \tag{1}$$

Logistic regression uses a sigmoid function to transform the output to a probability value; the objective is to minimize the cost function to achieve an optimal probability. The cost function is calculated as shown in

$$\text{Cost}\,(h_\theta(x), y) = \begin{cases} \log\,(h_\theta(x)), & y = 1, \\ -\log\,(1 - h_\theta(x)), & y = 0. \end{cases} \tag{2}$$

2.2.2. Support Vector Machine. Support vector machine (SVM) is another model for binary classification problem and is available in various kernels functions [28]. The objective of an SVM model is to estimate a hyperplane (or decision boundary) on the basis of feature set to classify data points [29]. The dimension of hyperplane varies according to the number of features. As there could be multiple possibilities for a hyperplane to exist in an N-dimensional space, the task is to identify the plane that separates the data points of two classes with maximum margin. A mathematical representation of the cost function for the SVM model is defined as given in [30] and shown in

$$J(\theta) = \frac{1}{2}\sum_{j=1}^{n}\theta_j^2, \tag{3}$$

such that

$$\theta^T x^{(i)} \geq 1, \quad y^{(i)} = 1, \tag{4}$$

$$\theta^T x^{(i)} \leq -1, \quad y^{(i)} = 0. \tag{5}$$

The function above uses a linear kernel. Kernels are usually used to fit data points that cannot be easily separable or data points that are multidimensional. In our case, we have used sigmoid SVM, kernel SVM (polynomial SVM), Gaussian SVM, and basic linear SVM models.

2.2.3. Multilayer Perceptron. A multilayer perceptron (MLP) is an artificial neural network, with an input layer, one or more hidden layers, and an output layer. MLP can be as simple as having each of the three layers; however, in our experiments we have fine-tuned the model with various parameters and number of layers to generate an optimum predicting model. A basic multilayered perceptron model with one hidden layer can be represented as a function as shown below [31]:

$$f(x) = g\left(b^{(2)} + W^{(2)}\left(s\left(b^{(1)} + W^{(1)}x\right)\right)\right). \tag{6}$$

Here, $b^{(1)}$ and $b^{(2)}$ are the bias vectors, $W^{(1)}$ and $W^{(2)}$ are the weight matrices, and g and s are the activation functions. In our case, the activation function is ReLU and the Adam solver, with 3 hidden layers.

2.2.4. K-Nearest Neighbors (KNN). KNN is an unsupervised machine learning model where a dependent variable is not required to predict the outcome on a specific data. We provide enough training data to the model and let it decide to which particular neighborhood a data point belongs. KNN model estimates the distance of a new data point to its nearest neighbors, and the value of K estimates the majority of its neighbors' votes; if the value of K is 1, then the new data point is assigned to a class which has the nearest distance. The mathematical formulae to estimate the distance between two points can be calculated as follows [31]:

$$\text{Euclidean distance} = \sqrt{\sum_{i=1}^{k} \left(x_i - y_i\right)^2}, \tag{7}$$

$$\text{Manhattan distance} = \sum_{i=1}^{k} \left|x_i - y_i\right|, \tag{8}$$

$$\text{Minkowski distance} = \left(\sum_{i=1}^{k} \left|x_i - y_i\right|^q\right)^{1/q}. \tag{9}$$

2.3. Ensemble Learners. We proposed using existing ensemble techniques along with textual characteristics as feature input to improve the overall accuracy for the purpose of classification between a truthful and a false article. Ensemble learners tend to have higher accuracies, as more than one model is trained using a particular technique to reduce the overall error rate and improve the performance of the model. The intuition behind the ensemble modeling is synonymous to the one we are already used to in our daily life such as requesting opinions of multiple experts before taking a particular decision in order to minimize the chance of a bad decision or an undesirable outcome. For example, a classification algorithm can be trained on a particular dataset with a unique set of parameters that can produce a decision boundary which fits the data to some extent. The outcome of that particular algorithm depends not only on the parameters that were provided to train the model, but also on the type of training data. If the training data contains less variance or uniform data, then the model might overfit and produce biased results over unseen data. Therefore, approaches like cross validation are used to minimize the risk of overfitting. A number of models can be trained on different set of parameters to create multiple decision boundaries on randomly chosen data points as training data. Hence, using ensemble learning techniques, these problems can be addressed and mitigated by training multiple algorithms, and their results can be combined for near optimum outcome. One such technique is using voting classifiers where the final classification depends on the major votes provided by all algorithms [32]. However, there are other ensemble techniques as well that can be used in different scenarios such as the following.

2.3.1. Random Forest (RF). Random forest (RF) is an advanced form of decision trees (DT) which is also a supervised learning model. RF consists of large number of decision trees working individually to predict an outcome of a class where the final prediction is based on a class that received majority votes. The error rate is low in random forest as compared to other models, due to low correlation among trees [33]. Our random forest model was trained using different parameters; i.e., different numbers of estimators were used in a grid search to produce the best model that can predict the outcome with high accuracy. There are multiple algorithms to decide a split in a decision tree based on the problem of regression or classification. For the classification problem, we have used the Gini index as a cost function to estimate a split in the dataset. The Gini index is calculated by subtracting the sum of the squared probabilities of each class from one. The mathematical formula to calculate the Gini index (G_{ind}) is as follows [34]:

$$G_{\text{ind}} = 1 - \sum_{i=1}^{c} \left(P_i\right)^2, \tag{10}$$

2.3.2. Bagging Ensemble Classifiers. bootstrap aggregating, or in short bagging classifier, is an early ensemble method mainly used to reduce the variance (overfitting) over a training set. Random forest model is one of the most frequently used as a variant of bagging classifier. Intuitively, for a classification problem, the bagging model selects the class on the basis of major votes estimated by M number of trees to reduce the overall variance, while the data for each tree is selected using random sampling with replacement from overall dataset. For regression problems, however, the bagging model averages over multiple estimates.

2.3.3. Boosting Ensemble Classifiers. boosting is another widely used ensemble method to train weak models to become strong learners. For that purpose, a forest of randomized trees is trained, and the final prediction is based on the majority vote outcome from each tree. This method allows weak learners to correctly classify data points in an incremental approach that are usually misclassified. Initially equal weighted coefficients are used for all data points to classify a given problem. In the successive rounds, the weighted coefficients are decreased for data points that are correctly classified and are increased for data points that are misclassified [35]. Each subsequent tree formed in each round learns to reduce the errors from the preceding round and to increase the overall accuracy by correctly classifying data points that were misclassified in previous rounds. One major problem with boosting ensemble is that it might overfit to the training data which may lead to incorrect predictions for unseen instances [36]. There are multiple boosting algorithms available that can be used for both the purposes of classification and regression. In our experiments we used XGBoost [37] and AdaBoost [38] algorithms for classification purpose.

2.3.4. Voting Ensemble Classifiers. voting ensemble is generally used for classification problems as it allows the combination of two or more learning models trained on the whole dataset [39]. Each model predicts an outcome for a sample data point which is considered a "vote" in favor of the class that the model has predicted. Once each model predicts the outcome, the final prediction is based on the majority vote for a specific class [32]. Voting ensemble, as compared to bagging and boosting algorithms, is simpler in terms of implementation. As discussed, bagging algorithms create multiple subsets of data by random sampling and replacement from the whole dataset, thus creating a number of datasets. Each dataset is then used to train a model, while the final result is an aggregation of outcome from each model. In case of boosting, multiple models are trained in a sequential manner where each model learns from the previous by increasing weights for the misclassified points, thus creating a generic model that is able to correctly classify the problem. However, voting ensemble on the other hand is a combination of multiple independent models that produces classification results that contribute to the overall prediction by majority voting.

2.4. Benchmark Algorithms. In this section, we discuss the benchmark algorithms with which we compare the performance of our methodology.

2.4.1. Linear SVM. We use linear SVM approach proposed in [21]. To ensure a meaningful comparison, we trained the linear SVM on the feature set as discussed in [21] with 5-fold cross validation. Note that the approach is referred to as Perez-LSVM in the text.

2.4.2. Convolutional Neural Network. Wang [18] used convolutional neural network (CNN) for automatic detection of fake news. We employed the same approach using our dataset. However, we could not use the feature set of Wang [18] as the dataset contains only short statements. The approach is referred to as Wang-CNN in the text.

2.4.3. Bidirectional Long Short-Term Memory Networks. Wang [18] also used bidirectional long short-term memory networks (Bi-LSTM), and we used the same approach with different feature sets. The approach is referred to as Wang-Bi-LSTM in the text.

2.5. Datasets. The datasets we used in this study are open source and freely available online. The data includes both fake and truthful news articles from multiple domains. The truthful news articles published contain true description of real world events, while the fake news websites contain claims that are not aligned with facts. The conformity of claims from the politics domain for many of those articles can be manually checked with fact checking websites such as politifact.com and snopes.com. We have used three different datasets in this study, a brief description of which is provided as follows.

The first dataset is called the "ISOT Fake News Dataset" [23] (hereafter referred to as DS1) which contains both true and fake articles extracted from the World Wide Web. The true articles are extracted from reuters.com which is a renowned news website, while the fake articles were extracted from multiple sources, mostly websites which are flagged by politifact.com. The dataset contains a total of 44,898 articles, out of which 21,417 are truthful articles and 23,481 fake articles. The total corpora contain articles from different domains, but most prominently target political news.

The second dataset is available at Kaggle [24] (hereafter referred to as DS2) which contains a total of 20,386 articles used for training and 5,126 articles used for testing. The dataset is built from multiple sources on the Internet. The articles are not limited to a single domain such as politics as they include both fake and true articles from various other domains.

The third dataset is also available at Kaggle [25] (hereafter referred to as DS3); it includes a total of 3,352 articles, both fake and true. The true articles are extracted from trusted online sources such as CNN, Reuters, the New York Times, and various others, while the fake news articles are extracted from untrusted news websites. The domains it covered include sports, entertainment, and politics.

A combined dataset is the collection of articles from the three datasets (hereafter referred to as DS4). As the articles vary in nature in each dataset, the fourth dataset is created to evaluate the performance of algorithms on datasets which cover a wide array of domains in a single dataset.

2.6. Performance Metrics. To evaluate the performance of algorithms, we used different metrics. Most of them are based on the confusion matrix. Confusion matrix is a tabular representation of a classification model performance on the test set, which consists of four parameters: true positive, false positive, true negative, and false negative (see Table 1).

2.6.1. Accuracy. Accuracy is often the most used metric representing the percentage of correctly predicted observations, either true or false. To calculate the accuracy of a model performance, the following equation can be used:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{11}$$

In most cases, high accuracy value represents a good model, but considering the fact that we are training a classification model in our case, an article that was predicted as true while it was actually false (false positive) can have negative consequences; similarly, if an article was predicted as false while it contained factual data, this can create trust issues. Therefore, we have used three other metrics that take into account the incorrectly classified observation, i.e., precision, recall, and F1-score.

2.6.2. Recall. Recall represents the total number of positive classifications out of true class. In our case, it represents the number of articles predicted as true out of the total number of true articles.

Table 1: Confusion matrix.

	Predicted true	Predicted false
Actual true	True positive (TP)	False negative (FN)
Actual false	False positive (FP)	True negative (TN)

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \qquad (12)$$

2.6.3. Precision. Conversely, precision score represents the ratio of true positives to all events predicted as true. In our case, precision shows the number of articles that are marked as true out of all the positively predicted (true) articles:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \qquad (13)$$

2.6.4. F1-Score. F1-score represents the trade-off between precision and recall. It calculates the harmonic mean between each of the two. Thus, it takes both the false positive and the false negative observations into account. F1-score can be calculated using the following formula:

$$\text{F1} - \text{score} = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \qquad (14)$$

3. Results and Discussion

Table 2 summarizes the accuracy achieved by each algorithm on the four considered datasets. It is evident that the maximum accuracy achieved on DS1 (ISOT Fake News Dataset) is 99%, achieved by random forest algorithm and Perez-LSVM. Linear SVM, multilayer perceptron, bagging classifiers, and boosting classifiers achieved an accuracy of 98%. The average accuracy attained by ensemble learners is 97.67% on DS1, whereas the corresponding average for individual learners is 95.25%. The absolute difference between individual learners and ensemble learners is 2.42% which is not significant. Benchmark algorithms Wang-CNN and Wang-Bi-LSTM performed poorer than all other algorithms. On DS2, bagging classifier (decision trees) and boosting classifier (XGBoost) are the best performing algorithms, achieving an accuracy of 94%. Interestingly, linear SVM, random forest, and Perez-LSVM performed poorly on DS2. Individual learners reported an accuracy of 47.75%, whereas ensemble learners' accuracy is 81.5%. A similar trend is observed for DS3, where individual learners' accuracy is 80% whereas ensemble learners' accuracy is 93.5%. However, unlike DS2, the best performing algorithm on DS3 is Perez-LSVM which achieved an accuracy of 96%. On DS4 (DS1, DS2, and DS3 combined), the best performing algorithm is random forest (91% accuracy). On average, individual learners achieved an accuracy of 85%, whereas ensemble learners achieved an accuracy of 88.16%. The worst performing algorithm is Wang-Bi-LSTM which achieved an accuracy of 62%.

Table 2: Overall accuracy score for each dataset.

	DS1	DS2	DS3	DS4
Logistic regression (LR)	0.97	0.91	0.91	0.87
Linear SVM (LSVM)	0.98	0.37	0.53	0.86
Multilayer perceptron	0.98	0.35	0.94	0.9
K-nearest neighbors (KNN)	0.88	0.28	0.82	0.77
Ensemble learners				
Random forest (RF)	**0.99**	0.35	0.95	**0.91**
Voting classifier (RF, LR, KNN)	0.97	0.88	0.94	0.88
Voting classifier (LR, LSVM, CART)	0.96	0.86	0.92	0.85
Bagging classifier (decision trees)	0.98	**0.94**	0.94	0.9
Boosting classifier (AdaBoost)	0.98	0.92	0.92	0.86
Boosting classifier (XGBoost)	0.98	**0.94**	0.94	0.89
Benchmark algorithms				
Perez-LSVM	**0.99**	0.79	**0.96**	0.9
Wang-CNN	0.87	0.66	0.58	0.73
Wang-Bi-LSTM	0.86	0.52	0.57	0.62

Figure 2 summarizes the average accuracy of all algorithms over the 4 datasets. Overall, the best performing algorithm is bagging classifier (decision trees) (accuracy 94%), whereas the worst performing algorithm is Wang-Bi-LSTM (accuracy 64.25%). Individual learners' accuracy is 77.6% whereas the accuracy of ensemble learners is 92.25%. Random forest achieved better accuracy on all datasets except DS2. However, accuracy score alone is not a good measure to evaluate the performance of a model; therefore, we also evaluate performance of learning models on the basis of recall, precision, and F1-score.

Tables 3–5 summarize the recall, precision, and F1 score of each algorithm on all the four datasets. In terms of average precision (Table 3), boosting classifier (XGBoost) achieved the best results. The average precision of boosting classifier (XGBoost) on all the four datasets is 95.25%. Random forest (RF) achieved a precision of 79.75%; however, on the three datasets (removing the dataset with the lowest score, i.e., DS2), the average precision of random forest jumped to 96.3%. The corresponding score for boosting classifier (XGBoost) is 96.3% as well.

Based on the recall performance metric, bagging classifier (decision trees) stands best by achieving a recall score of 0.942. This is closely followed by boosting classifier (XGBoost) which achieved a recall of 0.94. Among the benchmark algorithms, Perez-LSVM is found to be best performing algorithm, achieving a recall score of 0.92. The algorithms exhibited a similar performance behavior on F1-score as that of precision. Boosting classifier (XGBoost) achieved F1-score of 0.945, the best among all the techniques, followed by bagging classifier (decision trees) and logistic regression (LR).

Figure 3 is a graphical representation of average performance of learning algorithms on all datasets using precision, recall, and F1-score. It can be seen that there is not much difference between the performance of learning algorithms using various performance metrics except for linear SVM, KNN, Wang-CNN, and Wang-Bi-LSTM.

The ensemble learner XGBoost performed better in comparison to other learning models on all performance

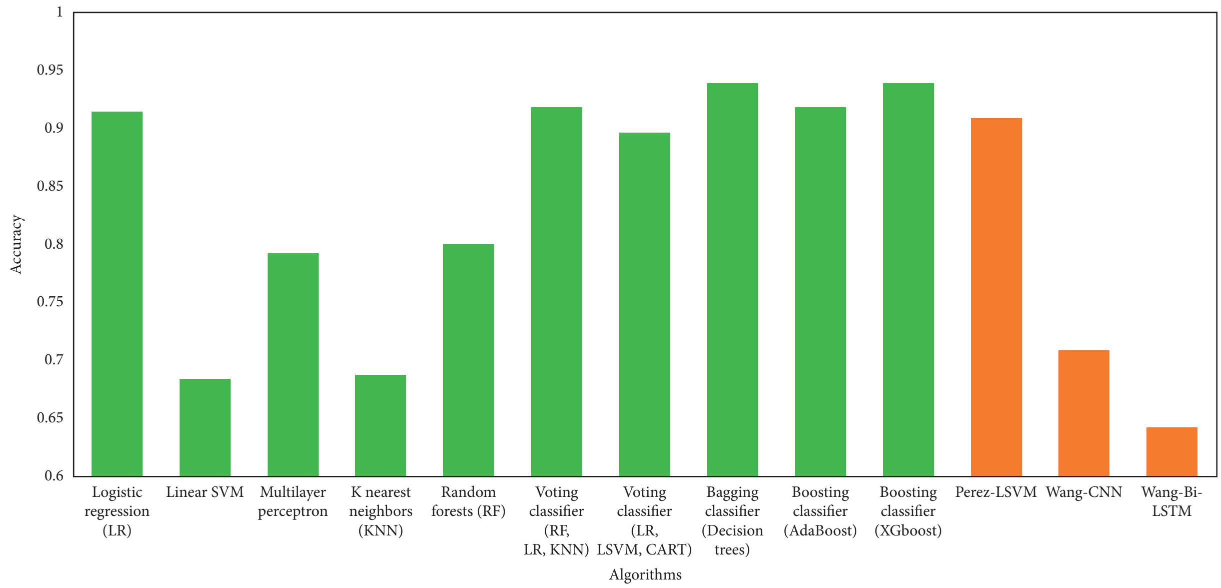

Figure 2: Average accuracy over all datasets.

Table 3: Precision on the 4 datasets.

	DS1	DS2	DS3	DS4
Logistic regression (LR)	0.98	0.92	0.93	0.88
Linear SVM (LSVM)	0.98	0.31	0.54	0.88
Multilayer perceptron	0.97	0.32	0.93	**0.92**
K-nearest neighbors (KNN)	0.91	0.22	0.85	0.8
Ensemble learners				
Random forest (RF)	0.99	0.3	0.98	0.92
Voting classifier (RF, LR, KNN)	0.96	0.88	0.92	0.86
Voting classifier (LR, LSVM, CART)	0.94	0.86	0.88	0.83
Bagging classifier (decision trees)	0.98	**0.94**	0.93	0.9
Boosting classifier (AdaBoost)	0.98	0.92	0.92	0.86
Boosting classifier (XGBoost)	**0.99**	**0.94**	**0.96**	**0.92**
Benchmark algorithms				
Perez-LSVM	**0.99**	0.79	**0.96**	0.9
Wang-CNN	0.84	0.65	0.48	0.72
Wang-Bi-LSTM	0.92	0.43	0.5	0.65

Table 4: Recall on the 4 datasets.

	DS1	DS2	DS3	DS4
Logistic regression (LR)	0.98	0.9	0.92	0.86
Linear SVM (LSVM)	0.98	0.32	**1**	0.86
Multilayer perceptron	**1**	0.36	0.96	0.88
K-nearest neighbors (KNN)	0.87	0.24	0.81	0.74
Ensemble learners				
Random forest (RF)	**1**	0.34	0.93	**0.91**
Voting classifier (RF, LR, KNN)	0.97	0.89	0.96	0.9
Voting classifier (LR, LSVM, CART)	0.97	0.87	0.96	0.89
Bagging classifier (decision trees)	0.97	**0.95**	0.94	**0.91**
Boosting classifier (AdaBoost)	0.98	0.93	0.92	0.86
Boosting classifier (XGBoost)	0.99	0.94	0.94	0.89
Benchmark algorithms				
Perez-LSVM	0.99	0.81	0.97	**0.91**
Wang-CNN	0.9	0.71	0.29	0.75
Wang-Bi-LSTM	0.78	0.59	0.35	0.61

Table 5: F1-score on the 4 datasets.

	DS1	DS2	DS3	DS4
Logistic regression (LR)	0.98	0.91	0.92	0.87
Linear SVM (LSVM)	0.98	0.32	0.7	0.87
Multilayer perceptron	0.98	0.34	0.95	0.9
K-nearest neighbors (KNN)	0.89	0.23	0.83	0.77
Ensemble learners				
Random forest (RF)	**0.99**	0.32	0.95	**0.91**
Voting classifier (RF, LR, KNN)	0.97	0.88	0.94	0.88
Voting classifier (LR, LSVM, CART)	0.96	0.86	0.92	0.86
Bagging classifier (decision trees)	0.98	**0.94**	0.94	0.9
Boosting classifier (AdaBoost)	0.98	0.92	0.92	0.86
Boosting classifier (XGBoost)	**0.99**	**0.94**	0.95	0.9
Benchmark algorithms				
Perez-LSVM	**0.99**	0.8	**0.96**	0.9
Wang-CNN	0.87	0.67	0.31	0.73
Wang-Bi-LSTM	0.84	0.44	0.35	0.57

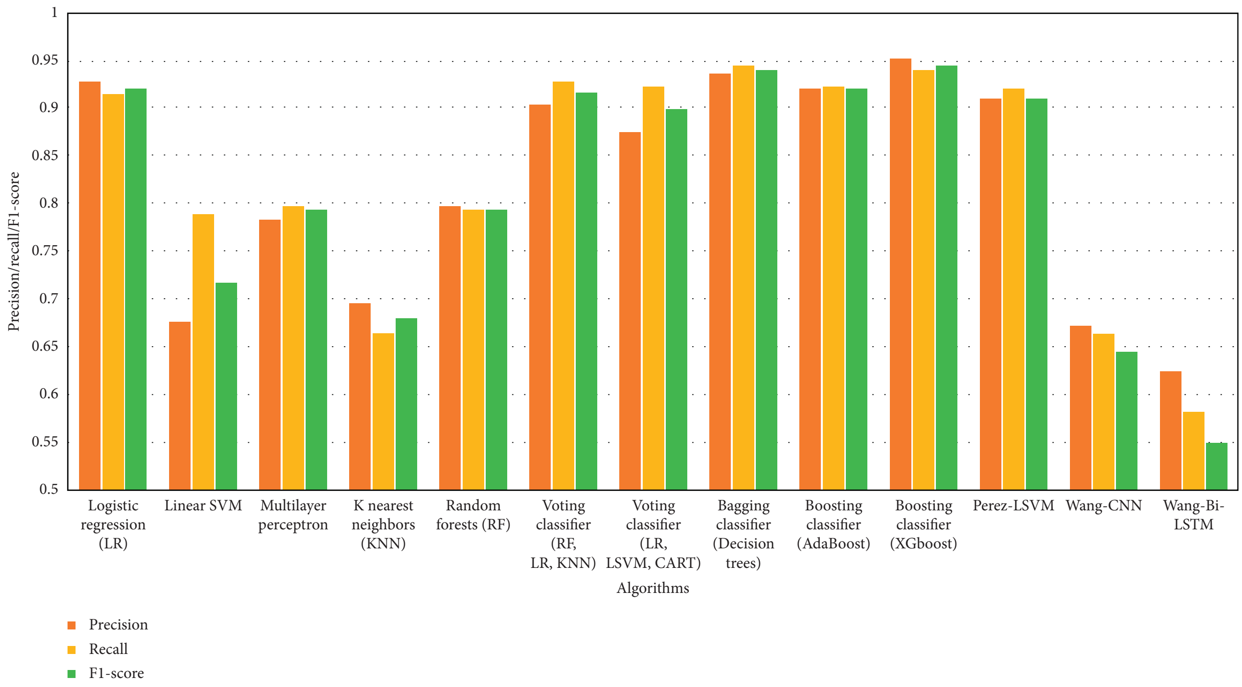

Figure 3: Precision, recall, and F1-score over all datasets.

metrics. The main factor leading to the superior performance of XGBoost is the working principle which efficiently identifies errors and minimizes them in each iteration. The basic intuition behind the working of XGBoost is to use multiple classification and regression trees (CART) which combine multiple weak learners to assign higher weights to misclassified data points. Therefore, during each subsequent iteration, the model is able to correctly identify the misclassified points whereas regularization parameters are used to reduce the overfitting problem.

Logistic regression is a relatively simpler model but achieved an average accuracy of over 90% on the three datasets (DS1, DS2, and DS3). There can be multiple explanations for achieving a high average accuracy; firstly, logistic regression model is fine-tuned using an extensive grid search with different hyperparameters; secondly, some of the datasets (such as DS1) have similar writing styles of authors, which led to 97% accuracy of logistic regression model. On DS4, which is the combination of all the three datasets (and thereby includes more versatile writing styles as well), the accuracy of logistic regression drops to 87%.

4. Conclusion

The task of classifying news manually requires in-depth knowledge of the domain and expertise to identify anomalies in the text. In this research, we discussed the problem of classifying fake news articles using machine learning models and ensemble techniques. The data we used in our work is collected from the World Wide Web and contains news articles from various domains to cover most of the news rather than specifically classifying political news. The primary aim of the research is to identify patterns in text that differentiate fake articles from true news. We extracted different textual features from the articles using an LIWC tool and used the feature set as an input to the models. The learning models were trained and parameter-tuned to obtain optimal accuracy. Some models have achieved comparatively higher accuracy than others. We used multiple performance metrics to compare the results for each algorithm. The ensemble learners have shown an overall better score on all performance metrics as compared to the individual learners.

Fake news detection has many open issues that require attention of researchers. For instance, in order to reduce the spread of fake news, identifying key elements involved in the spread of news is an important step. Graph theory and machine learning techniques can be employed to identify the key sources involved in spread of fake news. Likewise, real time fake news identification in videos can be another possible future direction.

Data Availability

Previously reported data were used to support this study and are available at https://www.kaggle.com/c/fake-news and https://www.kaggle.com/jruvika/fake-news-detection.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] A. Douglas, "News consumption and the new electronic media," *The International Journal of Press/Politics*, vol. 11, no. 1, pp. 29–52, 2006.

[2] J. Wong, "Almost all the traffic to fake news sites is from facebook, new data show," 2016.

[3] D. M. J. Lazer, M. A. Baum, Y. Benkler et al., "The science of fake news," *Science*, vol. 359, no. 6380, pp. 1094–1096, 2018.

[4] S. A. García, G. G. García, M. S. Prieto, A. J. M. Guerrero, and C. R. Jiménez, "The impact of term fake news on the scientific community scientific performance and mapping in web of science," *Social Sciences*, vol. 9, no. 5, 2020.

[5] A. D. Holan, *2016 Lie of the Year: Fake News*, Politifact, Washington, DC, USA, 2016.

[6] S. Kogan, T. J. Moskowitz, and M. Niessner, "Fake News: Evidence from Financial Markets," 2019, https://ssrn.com/abstract=3237763.

[7] A. Robb, "Anatomy of a fake news scandal," *Rolling Stone*, vol. 1301, pp. 28–33, 2017.

[8] J. Soll, "The long and brutal history of fake news," *Politico Magazine*, vol. 18, no. 12, 2016.

[9] J. Hua and R. Shaw, "Corona virus (covid-19) "infodemic" and emerging issues through a data lens: the case of China," *International Journal of Environmental Research and Public Health*, vol. 17, no. 7, p. 2309, 2020.

[10] N. K. Conroy, V. L. Rubin, and Y. Chen, "Automatic deception detection: methods for finding fake news," *Proceedings of the Association for Information Science and Technology*, vol. 52, no. 1, pp. 1–4, 2015.

[11] F. T. Asr and M. Taboada, "Misinfotext: a collection of news articles, with false and true labels," 2019.

[12] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media," *ACM SIGKDD Explorations Newsletter*, vol. 19, no. 1, pp. 22–36, 2017.

[13] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science*, vol. 359, no. 6380, pp. 1146–1151, 2018.

[14] H. Allcott and M. Gentzkow, "Social media and fake news in the 2016 election," *Journal of Economic Perspectives*, vol. 31, no. 2, pp. 211–236, 2017.

[15] V. L. Rubin, N. Conroy, Y. Chen, and S. Cornwell, "Fake news or truth? using satirical cues to detect potentially misleading news," in *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pp. 7–17, San Diego, CA, USA, 2016.

[16] H. Jwa, D. Oh, K. Park, J. M. Kang, and H. Lim, "exBAKE: automatic fake news detection model based on bidirectional encoder representations from transformers (bert)," *Applied Sciences*, vol. 9, no. 19, 2019.

[17] H. Ahmed, I. Traore, and S. Saad, "Detection of online fake news using n-gram analysis and machine learning techniques," in *Proceedings of the International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, pp. 127–138, Springer, Vancouver, Canada, 2017.

[18] W. Y. Wang, *Liar, Liar Pants on Fire: A New Benchmark Dataset for Fake News Detection*, Association for Computational Linguistics, Stroudsburg, PA, USA, 2017.

[19] B. Riedel, I. Augenstein, G. P. Spithourakis, and S. Riedel, "A simple but tough-to-beat baseline for the fake news challenge stance detection task," 2017, https://arxiv.org/abs/1707.03264.

[20] N. Ruchansky, S. Seo, and Y. Liu, "Csi: a hybrid deep model for fake news detection," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 797–806, Singapore, 2017.

[21] V. Pérez-Rosas, B. Kleinberg, A. Lefevre, and R. Mihalcea, "Automatic detection of fake news," 2017, https://arxiv.org/abs/1708.07104.

[22] P. Bühlmann, "Bagging, boosting and ensemble methods," in *Handbook of Computational Statistics*, pp. 985–1022, Springer, Berlin, Germany, 2012.

[23] H. Ahmed, I. Traore, and S. Saad, "Detecting opinion spams and fake news using text classification," *Security and Privacy*, vol. 1, no. 1, 2018.

[24] Kaggle, *Fake News*, Kaggle, San Francisco, CA, USA, 2018, https://www.kaggle.com/c/fake-news.

[25] Kaggle, *Fake News Detection*, Kaggle, San Francisco, CA, USA, 2018, https://www.kaggle.com/jruvika/fake-news-detection.

[26] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[27] T. M. Mitchell, *The Discipline of Machine Learning*, Carnegie Mellon University, Pittsburgh, PA, USA, 2006.

[28] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, Cambridge, UK, 2000.

[29] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The Annals of Statistics*, vol. 36, no. 3, pp. 1171–1220, 2008.

[30] V. Kecman, *Support Vector Machines-An Introduction in "Support Vector Machines: Theory and Applications"*, Springer, New York City, NY, USA, 2005.

[31] S. Akhtar, F. Hussain, F. R. Raja et al., "Improving mispronunciation detection of arabic words for non-native learners using deep convolutional neural network features," *Electronics*, vol. 9, no. 6, 2020.

[32] D. Ruta and B. Gabrys, "Classifier selection for majority voting," *Information Fusion*, vol. 6, no. 1, pp. 63–81, 2005.

[33] B. Gregorutti, B. Michel, and P. Saint-Pierre, "Correlation and variable importance in random forests," *Statistics and Computing*, vol. 27, no. 3, pp. 659–678, 2017.

[34] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Springer, Berlin, Germany, 1984.

[35] R. E. Schapire, "A brief introduction to boosting," *IJCAI*, vol. 99, pp. 1401–1406, 1999.

[36] E. M. Dos Santos, R. Sabourin, and P. Maupin, "Overfitting cautious selection of classifier ensembles with genetic algorithms," *Information Fusion*, vol. 10, no. 2, pp. 150–162, 2009.

[37] T. Chen and C. Guestrin, "Xgboost: a scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, San Francisco, CA, USA, 2016.

[38] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[39] L. Lam and S. Y. Suen, "Application of majority voting to pattern recognition: an analysis of its behavior and performance," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 27, no. 5, pp. 553–568, 1997.