Comparative Study of Machine Learning Algorithms and Text Vectorization Methods for Fake News Detection

Andreas Kanavos*, Ioannis Karamitsos†, Alaa Mohasseb‡ and Vassilis C. Gerogiannis§
*Department of Informatics
Ionian University, Corfu, Greece
akanavos@ionio.gr
†Department of Graduate and Research
Rochester Institute of Technology, Dubai, UAE
ixkcad1@rit.edu
‡School of Computing
University of Portsmouth, Portsmouth, UK
alaa.mohasseb@port.ac.uk
§Department of Digital Systems
University of Thessaly, Larissa, Greece
vgerogian@uth.gr

***Abstract*—The detection of fake news is a crucial task in today's society, given the widespread use of social media and online platforms. In this study, we investigate the application of Machine Learning (ML) algorithms for the detection of fake news. We consider two different datasets of categorized news articles of various sizes and apply various ML algorithms, along with two methods of text vectorization. Specifically, w e e xamine B ag of Words and Tf-Idf, with the use of stemming and with different n-gram values. The resulting vectors are processed by Naive Bayes algorithms, Linear algorithms, Support Vector Machines, and Random Forest classifiers. F 1-score a nd c omputational t ime for each algorithm-vectorization combination were recorded. Our results have shown that Linear Algorithms and Support Vector Machines combined with Tf-Idf vectors and n-gram value of (1,2) produced the highest accuracies, with an F1-score up to 96.8%.**

***Index Terms*—Machine Learning, Text Mining, Information Retrieval, Fake News Detection**

I. Introduction

In recent years, there has been increasing global concern about the phenomenon of fake news spread. Fake news problems can be classified e ither a s d isinformation (i.e., false information that is purposely spread to deceive people) or as misinformation (i.e., false or misleading information). Fake news are also due to inherent biases in news produced by humans with human bias. Lazer et al. [17] define the fake news phenomenon as the "fabricated information that mimics news media content in form but not in organizational process or intent". This misinformation can be spread through traditional news media, social media, or other online platforms and can have serious consequences for individuals and society. Fake news is a term widely used in the online community to describe news that is fake or contains some form of false information disseminated through various communication channels, such as print media, TV channels, and in today's world, social media. The information that was taken from newspapers has actually crossed over to social media today. The media has created a platform for many organizations to disseminate their opinion-based values that appeal to the emotions and beliefs of a particular target audience, with other forms of enticement targeted to appeal to less tech-savvy users who may click on links with misinformation.

The proliferation of false news in the media is gradually increasing as it originates from single individuals, who could then make their own false opinions and unreliable facts to support those opinions a worldwide issue [19]. A representative platform that can be scrutinized to support this theory is Twitter. Twitter is an online social networking platform that allows people all over the world to share their opinions on all sorts of issues that concern them [2], [14], [13], [15]. The goal of social media platforms such as Twitter is to increase the argumentative nature of the news by providing a means for people to discuss their own beliefs on a particular topic, rather than focusing on the topic itself with an objective perspective.

Although the same problem has been widely discussed also in the past by several journalists, on the occasion of the publication of propaganda articles related to World War I and the development of corporate public relations in the 1920s [17], recent years have seen a rapid increase in the volume and spread of fake news. One of the main reasons for the growing phenomenon of fake news is the World Wide Web and social media, as more and more people are using these media to get information, but also because they offer a cheap, fast, and easy way to disseminate news [12], as ordinary users can directly

comment and share any publication publicly. There may be various motivations for spreading fake news, such as financial and ideological aims. Economic because widespread fake news brings profits to content providers through advertising or encourages readers to buy products, and ideological because organizations and ideologies are often promoted through fake news articles [1].

One event that generated a lot of interest and brought about the problem of disinformation was the 2016 U.S. presidential election. During this election, there was a large flow of fake news. A typical example is the "Pizzagate conspiracy theory," in which a conspiracy theory was spread through social media [22]. According to this conspiracy theory, a pizzeria in Washington DC was allegedly a "showcase" for a child abuse ring involving U.S. presidential candidate Hillary Clinton. The spread of this conspiracy theory led to the restaurant in question being attacked by a gunman on December 4, 2016, fortunately with no casualties. More recently, with the emergence of the COVID-19 pandemic, there has been a flood of fake news and misinformation around the world.

In the field of IT, one main interest is to automate the detection of fake news using data mining and machine learning methods [31]. Machine learning algorithms can be used to help detect fake news by analyzing patterns and language used in news articles, social media posts, and other types of content. Some machine learning approaches include natural language processing (NLP), text classification, and information verification [16], [33]. A good example, with quite common features, is Spam Detection [24]. An email text is categorized as malicious (spam) or not, with the help of a software system. There are several successful approaches to solving the problem with data mining and machine learning methods [11], and now well-known email providers use corresponding methods and technologies to protect their subscribers from such emails. Other important developments that contribute to the increased possibility of solving the problem through machine learning are Natural Language Processing methods, such as Word2Vec [23]. With these methods, it is now possible to represent texts not only as strings and word occurrence frequencies but also conceptually, i.e., based on the actual meaning of words or even entire texts.

We can conclude that there are four research questions associated with the spread of fake news:

- RQ1: What are the primary factors contributing to the spread of fake news on social media platforms?
- RQ2: How does the consumption of fake news impact public opinion and political behavior?
- RQ3: What are the most effective methods for detecting and mitigating the spread of fake news?
- RQ4: What role do media literacy and public education play in combating the spread of fake news?

To address these research questions, the approach which is presented in this paper examines the application of Machine Learning methods for fake news detection. In particular, in the present paper, we introduce the Multinomial Naive Bayes method with N-gram models for the classification of fake news. The Multinomial Naive Bayes method is an extension of the Naive Bayes algorithm that is particularly suitable for text classification tasks, such as fake news detection. When combined with N-gram models, the Multinomial Naive Bayes classifier can effectively capture local patterns in text data and leverage contextual information for improved classification performance. It has been used to detect fake news based on textual features like word frequencies. N-gram models are a type of statistical language model that analyzes sequences of words, characters, or other elements in a given text. These models are used to predict the next item in a sequence, based on the frequency of N-1 preceding items.

More specifically, the main contributions of this paper are as follows: (1) Compare various text vectorization techniques, such as Bag-of-Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), and n-gram, to determine their effectiveness in capturing the relevant features of text for fake news detection. (2) Introduce different machine learning algorithms evaluating their interpretability of practical applications of fake news detection. (3) Evaluate the computational efficiency of different algorithms and text vectorization methods.

II. Related Work

Some authors have mainly been interested in the spread of fake news using social media platforms such as Twitter and Facebook. Vosoughi et al. [35] conducted a comprehensive study examining the spread of true and false news online, specifically on the Twitter platform. The authors collected and analyzed a dataset of news stories for the period between 2006 and 2017, and in their study they found that fake news spread significantly faster and further than true stories in all information categories. This study also highlighted that this difference in the spread of true and false news could not be attributed solely to the influence of bots, as they propagated both types of news at similar rates.

Lazer et al. [17] presented an overview of the science of fake news discussing its prevalence, impact, and potential countermeasures. The authors highlighted that the rapid rise of fake news has been facilitated by the digital revolution, which allows for the easy creation and dissemination of false information. The authors emphasized that the phenomenon of fake news is driven by a combination of human decision-making, social network structures, and online platforms' algorithms. This article also explored various factors that contribute to the spread of fake news, such as cognitive biases and social identity.

Pennycook et al. [28] investigated the "implied truth effect" in their study, which examines the impact of attaching warnings to a subset of fake news on the perceived accuracy of reports without warnings. In their study, they found that when a subset of fake news was labeled with a warning label, participants tended to perceive the unlabeled stories (both true and false) as more accurate. The authors also found that the implied truth effect was more pronounced among participants who were more likely to believe in conspiracy theories and among participants who were less able to distinguish between

true and false headlines. This study concluded that caution is needed when designing and implementing fact-checking and alerting initiatives.

Shao et al. [30] designed a platform called "Hoaxy" to track the spread of online misinformation. The objective of this system was to visualize the diffusion of fake news and to determine how misinformation spreads through social media networks. An interesting study about "rumor cascades" was conducted by Friggeri et al. [8] in 2014. The term rumor cascades were used as the chains of reshares of a piece of information on social media platforms, especially on Facebook, By measuring the dynamics of rumor spreading the authors analyzed the rumors collected from Facebook for the period between 2010 and 2012. To better understand the dynamics of rumor cascades and their effects, the authors concluded that is essential for developing effective strategies to measure the spread of misinformation on social platforms.

Bessi et al. [3] studied the role of social bots that infer the online discussion on Twitter during the 2016 US Presidential election. During the study, the authors identified that social bots select certain tweets and hashtags and create artificial popularity or consensus around specific topics. It has conclusively been shown that social bots contribute to the spread of misinformation and the distortion of online discussions during events such as elections.

Guess et al. [9] examined the impact of misinformation on fake news consumption during the 2016 US presidential election. Using multiple data sources (web data, survey responses, fake news database), the authors found that a small group of individuals consumed a high number of fake news stories. They found that there is a correlation between the group and higher fake news consumption, namely age, political ideology, and strong support for the candidate Donald Trump. Allcott et al. [1] also investigated the differential role of social media in the spread of fake news during the 2016 US presidential election, and analyzed various sources of data in terms of prevalence and potential impact. The authors identified Facebook as the common platform for the dissemination of fake news.

Several linguistic feature-based learning models for fake news detection and classification use various text features to identify and categorize fake news articles. These models use machine learning algorithms to learn patterns from linguistic features and effectively distinguish between real and fake news.

For example, Choudhary et. al. [5] presented a learning model based on linguistic features for detecting and classifying fake news. This model focuses on extracting a set of linguistic features from texts and applying machine learning techniques to distinguish between real and fake news articles.

Authors in [10] proposed a machine learning approach combined with effective feature extraction techniques to classify fake news. This study aimed to improve the performance of fake news detection and classification by using multiple machine learning algorithms (ensemble approach) and focusing on relevant features.

III. Material and Methods

A. Libraries

A number of libraries have been developed to facilitate the implementation of Machine Learning algorithms [6]:

- NumPy: It is a basic library used in Machine Learning and provides possibilities to create and edit multidimensional arrays and perform linear algebra calculations [26], [32]. Linear algebra is necessary to solve problems with multidimensional variables, which are usually presented in problems that require their solution using Machine Learning.
- Pandas: It is based on the infrastructure offered by NumPy and provides the ability to use DataFrames, which are a table-like data type with the ability to perform calculations, resize elements of the table, and the possibility of inhomogeneity of the data per column [21]. Apart from this, it can be used as a database and falls into the NoSQL category.
- NLTK: It is an important library in the field of text processing and provides text processing and natural language processing capabilities, such as recognizing the type of words, splitting words, removing Stop Words, identifying grammar elements, etc. [4].
- Gensim: It contains implementations of several algorithms such as LDA, Word2Vec, Doc2Vec and Text Summarization [29]. It is used by several agencies, and is a very important tool for text analysis and information extraction from texts.
- Scikit-Learn: It is probably the most widely used library in the Python language for executing Machine Learning algorithms [27]. It contains applications of many Machine Learning algorithms, such as Decision Trees, Support Vector Machines, Linear and Logistic Regression, but also data preparation algorithms, such as Normalization, Bag of Words and Tf-Idf, and many other applications.
- SciPy: It is a collection of libraries, such as NumPy, Pandas, matplotlib, and others [34]. At the same time, as a library, it provides some possibilities for scientific calculations, such as the possibilities of performing linear algebra operations, Fourier transforms, Signal Processing, but also data structures for sparse tables.

B. Algorithms

1) Naive Bayes: Naive Bayes algorithms belong to a group of classification algorithms based on probabilities, which utilize Bayes' theorem. These algorithms assume that each parameter of an element to be categorized is independent of the value of any other parameter. Several variants of Naive Bayes algorithms are commonly employed in solving classification problems, including:

Multinomial Naive Bayes: This algorithm is well-suited for discrete features, such as word counts in text classification tasks [20]. It models the likelihood of each feature belonging to a specific class as a multinomial distribution. We define the probability of document (d) being in class (c) as follows:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c) \quad (1)$$

where $P(t_k|c)$ is the conditional probability of term t_k occurring in a document of class c. The term $P(t_k|c)$ is interpreted as a measure of how much evidence t_k contributes that class c is the correct class. $P(c)$ is the prior probability of a document occurring in class c. $< t_1, t_2, \ldots, t_k d >$ are the tokens in d that are part of the vocabulary we use for classification and n_d is the number of such tokens in d. For example, $< t_1, t_2, \ldots, t_k d >$ for the one-sentence document "Covid is a fake news" might be $< Covid, fake, news >$, with $n_d = 3$, if we treat the terms 'is' and 'a' as common stop words.

The relation to multinomial unigram language model is formulated as follows:

$$P(d|q) \propto P(d) \prod_{t \in q} P(t|M_d) \quad (2)$$

Comparing the two equations, the document term d in text classification (equation 1) takes the role of the query in language modeling (equation 2) and the class c in text classification takes the role of the document d in language modeling. We use equation (2) to rank documents according to the probability that they are relevant to the query (q).

Bernoulli Naive Bayes: Designed specifically for binary feature variables, this algorithm assumes that each feature is a binary variable [20]. It models the likelihood of each feature given a class as a Bernoulli distribution.

Complement Naive Bayes: A variant of the Naive Bayes algorithm, the Complement Naive Bayes seeks to address the issue of imbalanced datasets [20]. It aims to classify the minority class by considering the complement of the majority class, thereby increasing performance on imbalanced data.

2) Passive Aggressive: The Passive Aggressive classifier is a type of online learning algorithm that is particularly useful for handling streaming or dynamically changing data. It works by making updates to its model based on the new incoming data samples while trying to minimize the loss incurred for misclassified instances.

The algorithm is named "passive-aggressive" because of its two update strategies:

- Passive Strategy: If the current sample is classified correctly, the classifier remains passive and does not update its model.
- Aggressive Strategy: If the current sample is misclassified, the classifier becomes aggressive and updates its model in a way that tries to minimize the loss as much as possible. The update process adjusts the model's weights using a learning rate parameter and the misclassification error.

One advantage of the Passive Aggressive classifier is its ability to handle non-stationary data and adapt to changing patterns over time. It is suitable for scenarios where new data continuously arrives, and the model needs to be updated incrementally. Overall, the Passive Aggressive classifier is a valuable algorithm in scenarios with evolving data streams, where adaptive and incremental learning is essential. Its simplicity, efficiency, and ability to handle changing patterns make it a popular choice for online learning tasks.

3) Random Forest: The Random Forest algorithm combines the concepts of decision trees and bagging to create an ensemble of diverse decision trees and makes predictions based on their collective output. At its core, a decision tree is a flowchart-like model where internal nodes represent features or attributes, branches represent decisions or rules, and leaf nodes represent the class labels or predicted values. Each decision tree in a Random Forest is trained on a randomly sampled subset of the training data, as well as a random subset of the features. This randomness helps to introduce diversity and reduce the risk of overfitting.

The Random Forest algorithm constructs multiple decision trees independently and then combines their predictions through majority voting (for classification) or averaging (for regression). This ensemble approach improves the model's robustness, generalization, and overall predictive accuracy. It can handle both categorical and continuous features and automatically handles missing values and outliers.

The random sampling of data and features during tree construction introduces the concept of bagging (bootstrap aggregating). It helps to reduce the variance and decorrelate the individual decision trees, making them more independent and less likely to make the same errors. Consequently, the ensemble can capture a wide range of patterns and make more accurate predictions.

4) Ridge: The Ridge classifier is a linear classifier that belongs to the family of ridge regression methods. It is commonly used for binary classification tasks, where the goal is to assign input data points to one of two classes. The Ridge classifier is based on the concept of ridge regression, which is a regularized form of linear regression. Regularization is used to prevent overfitting and improve the generalization ability of the model. In ridge regression, a penalty term is added to the loss function, which encourages the model to have smaller and more balanced weights.

Similarly, the Ridge classifier introduces a regularization parameter, often denoted as alpha, that controls the amount of regularization applied to the model. A higher alpha value leads to stronger regularization, resulting in smaller weights and a simpler model. The regularization helps in handling multicollinearity and reducing the impact of noisy or irrelevant features.

The Ridge classifier employs the ridge loss function, which combines the logistic loss function with the L2 regularization term. The logistic loss function is used to measure the discrepancy between the predicted probabilities and the true class labels. The regularization term penalizes large weights and encourages them to be closer to zero.

One advantage of the Ridge classifier is its ability to handle situations where the number of features is larger than the number of samples, a scenario known as the "large p, small n"

problem. The regularization helps to stabilize the model and mitigate the issues arising from the limited number of samples.

5) Stochastic Gradient Descent (SGD): The Stochastic Gradient Descent (SGD) algorithm is an iterative optimization algorithm that aims to find the optimal parameters of a model by minimizing a given loss function. It operates on training data in small subsets, known as mini-batches, making it efficient for large datasets. The algorithm starts with an initial set of parameters and iteratively updates them based on the gradients of the loss function with respect to the current parameters. In each iteration, a mini-batch of training examples is randomly selected, and the gradients are calculated on that mini-batch. The parameters are then updated in the direction of the negative gradient, scaled by a learning rate. This process continues for a fixed number of iterations or until convergence criteria are met.

The stochastic nature of SGD, where each iteration only considers a subset of the training data, introduces randomness and noise into the parameter updates. While this noise can cause the algorithm to oscillate during training, it also allows SGD to escape local optima and explore different regions of the parameter space, potentially leading to better generalization and faster convergence.

SGD is known for its efficiency and scalability, as it can handle large datasets with millions or even billions of training examples. It is memory-friendly since it only requires a small subset of the data to be loaded into memory at a time. This property makes SGD suitable for online learning scenarios, where data arrives in a streaming fashion.

6) Support Vector Machine: Liblinear is a library for large-scale linear classification that provides efficient implementations of linear SVM (Support Vector Machine) models. It is designed to handle large datasets with a high number of features and is known for its fast training and prediction times.

LinearSVC, on the other hand, is a class within the Scikit-Learn library that provides an interface to the Liblinear library for linear SVM classification. [7] The LinearSVC class in Scikit-Learn is a wrapper around the Liblinear implementation, offering a user-friendly API and integration with other Scikit-Learn functionalities. LinearSVC is particularly useful for binary classification tasks, where it seeks to find an optimal hyperplane that separates the two classes with the largest margin. It employs a one-vs-rest strategy for multiclass problems, transforming them into a set of binary classification problems. LinearSVC supports various loss functions and regularization methods, allowing users to fine-tune the model's behavior based on their specific requirements.

C. Methodology

The process of applying each algorithm with N-gram models typically has five phases:

1) Preprocessing Phase: The text data is cleaned (removing stop words, stemming, lemmatizing and lowercasing) and then tokenize the text.
2) N-gram extraction: Generate N-grams (e.g., unigrams or bigrams) from the preprocessed text.
3) Vectorization Phase: Convert the N-grams into numerical feature vectors using methods like Term Frequency-Inverse Document Frequency (TF-IDF) or Bag of Words (BoW).
4) Train and evaluate the model phase: Use the feature vectors to train the classifiers, associating each document with its corresponding class ('fake' or 'real' news). A labeled dataset containing genuine and fake news articles is divided into training and testing sets. The learning model is trained on the training set and then evaluated on the testing set using performance metrics like precision, recall, F1-score, and accuracy.
5) Prediction Phase: Select new unseen documents using the testing model to predict their class.

IV. Vectorization Techniques

In this study, the vectorization techniques used are Bag of Words (BoW), Term Frequency-Inverse Document Frequency(Tf-Idf) and the n-gram vector 1 and 2. All these methods are widely used in natural language processing and computational linguistics when converting text into numerical features. In the BoW model, each word in the text is considered a feature without taken consideration the order of the words. The n-gram resolves the ordering issue and the n-gram model considers a sequence of 'n' words together. For example, a 2-gram (or bigram) model considers two words together. This can capture some context, which can be important in many text problems. For the n-gram models, we have two common algorithms

CountVectorizer: This is a simple method for converting textual data into a numerical representation. A matrix is created in which each word is a feature (column) and each document (or piece of text) is a row. The value in each cell can be a count value indicating how many times a word occurs in a document, or a binary value indicating the presence or absence of the word in the document. This can be extended to n-gram by setting the n-gram range parameter to (n,n), where n is the desired n-gram size. In our study the values of $n = 1, 2$.

Tf-Idf Vectorizer: This is a more advanced method that not only counts the occurrence of words but also weights them according to their importance to a document compared to all other documents. The idea is that a word that occurs in many documents may not be a good discriminator and therefore should be down-weighted. Similar to CountVectorizer, Tf-idf Vectorizer can also be used with n-gram by setting the n-gram range parameter accordingly.

A. Implementation

The two most frequent and basic measures for information retrieval are precision and recall. F1-score was computed for each class in a weighted average manner to evaluate the performance of each classification model.

Precision can be written as:

$$\textit{Precision} = \frac{TP}{TP + FP} \tag{3}$$

Recall can be rewritten as:

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

F1-score can be rewritten as:

$$F1\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5}$$

B. Datasets

In our study we utilized two datasets to investigate the performance of machine learning algorithms in detecting fake news: the Fake News dataset and the InClass competition dataset posted by the University of Tennessee Machine Learning Club.

One of the datasets used in this study is the Fake News Dataset[1], which contains a total of 6,355 articles and includes the article's title, text, and category, with articles classified as true or false. The distribution of true and false articles is roughly equal. The second dataset used is an InClass competition hosted on the Kaggle website by the University of Tennessee Machine Learning Club[2]. This dataset contains 26,000 articles, including the article's ID, title, author, text, and category, with a roughly equal split between fake and real news articles.

V. Experimental Evaluation

Table I displays the results of several algorithms applied to the first dataset, along with the vectorization and n-gram techniques used, and the F1 score obtained for each combination. The algorithms include BernoulliNB, ComplementNB, LinearSVC, MultinomialNB, PassiveAggressive, RandomForest, Ridge, and SGD. The vectorization techniques used are Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (Tf-Idf), and the n-gram values tested are 1, 2, and 1-2. The F1 scores range from 0.851 to 0.939.

LinearSVC with BoW and n-grams of 1 and 2 achieve the highest F1 score of 0.901, followed by Ridge with Tf-Idf and n-gram of 1 with an F1 score of 0.939, and PassiveAggressive with Tf-Idf and n-gram of 1 with an F1 score of 0.936. The worst-performing algorithm is BernoulliNB with BoW and n-gram of 1 and 2, with an F1 score of 0.851. Furthermore, the table shows that the best-performing algorithm for the first dataset using Bag of Words (BoW) is LinearSVC with n-gram 1 and 2 and an F1 score equals to 0.901. The best-performing algorithm using TF-IDF vectorization is Ridge with n-gram 1 and an F1 score of 0.939. However, it's worth noting that several algorithms perform similarly well, and the choice between BoW and TF-IDF may depend on the specific task and dataset.

Generally, algorithms that use Tf-Idf vectorization tend to outperform those that use Bag-of-Words (BoW) vectorization, with a few exceptions. The results indicate that using higher n-grams does not necessarily improve performance for all algorithms. Overall, the table provides a helpful comparison of the performance of several classification algorithms on the first dataset using different vectorization and n-gram techniques.

TABLE I
Results from several Algorithms regarding the first Dataset

Algorithm	Vectorization	N-Gram	F1-Score
Bernoulli NB	BoW	1,2	0,851
Bernoulli NB	Tf-Idf	1,2	0,851
Complement NB	BoW	1,2	0,873
Complement NB	Tf-Idf	1,2	0,898
Multinomial NB	BoW	1,2	0,872
Multinomial NB	Tf-Idf	1,2	0,905
Passive Aggressive	BoW	1,2	0.926
Passive Aggressive	Tf-Idf	1,2	0.935
Random Forest	BoW	1	0,888
Random Forest	BoW	1,2	0,874
Random Forest	BoW	2	0,872
Random Forest	Tf-Idf	1	0,890
Random Forest	Tf-Idf	1,2	0,886
Random Forest	Tf-Idf	2	0,872
Ridge	BoW	1,2	0.924
Ridge	Tf-Idf	1,2	0.939
SGD	BoW	1,2	0.918
SGD	Tf-Idf	1,2	0.934
SVC	BoW	1	0,878
SVC	BoW	1,2	0,901
SVC	BoW	2	0,900
SVC	Tf-Idf	1	0,936
SVC	Tf-Idf	1,2	0,929
SVC	Tf-Idf	2	0,916

Similarly, Table II displays the results of several algorithms applied to the second dataset. The best performing algorithm in terms of F1 score is PassiveAggressive, with a score of 0.964 when using Tf-Idf vectorization with 1,2 n-grams. LinearSVC also performs well, particularly with Tf-Idf vectorization, achieving a score of 0.962 with 1 n-gram and 0.959 with 1,2 n-grams. ComplementNB and MultinomialNB have similar performance, achieving a score of around 0.908-0.917 depending on the vectorization and n-gram used. RandomForest generally performs worse than the other algorithms, with the best F1 score being 0.919 when using Tf-Idf vectorization with 1 n-gram.

Looking at the results for the second dataset, we can see that the best performing algorithm for both Bag-of-Words (BoW) and Tf-Idf vectorization is LinearSVC. For BoW, the best performing n-gram is 1, with a F1 score of 0.942 for LinearSVC. For n-grams of 1,2 and 2, LinearSVC still performed the best, but with slightly lower F1 scores of 0.947 and 0.917, respectively. For Tf-Idf, the best performing n-gram is 1, with a F1 score of 0.962 for LinearSVC. For n-grams of 1,2 and 2, the F1 scores were slightly lower at 0.959 and 0.955, respectively. It is interesting to note that for this dataset, PassiveAggressive and Ridge also performed well with Tf-Idf vectorization, achieving F1 scores of 0.959 and 0.962, respectively. On the other hand, RandomForest performed relatively poorly for this dataset, with the highest F1 score of 0.919 achieved using Tf-Idf vectorization and an n-gram of 1.

[1]https://github.com/GeorgeMcIntire/fake_real_news_dataset

[2]https://www.kaggle.com/c/fake-news/data

TABLE II
RESULTS FROM SEVERAL ALGORITHMS REGARDING THE SECOND DATASET

Algorithm	Vectorization	N-Gram	F1-Score
Bernoulli NB	BoW	1,2	0,889
Bernoulli NB	Tf-Idf	1,2	0,903
Complement NB	BoW	1,2	0.909
Complement NB	Tf-Idf	1,2	0.923
Multinomial NB	BoW	1,2	0.909
Multinomial NB	Tf-Idf	1,2	0.925
Passive Aggressive	BoW	1,2	0.959
Passive Aggressive	Tf-Idf	1,2	0.964
Random Forest	BoW	1	0,915
Random Forest	BoW	1,2	0,888
Random Forest	BoW	2	0,918
Random Forest	Tf-Idf	1	0,919
Random Forest	Tf-Idf	1,2	0,909
Random Forest	Tf-Idf	2	0,917
Ridge	BoW	1,2	0.951
Ridge	Tf-Idf	1,2	0.959
SGD	BoW	1,2	0.945
SGD	Tf-Idf	1,2	0.957
SVC	BoW	1	0,942
SVC	BoW	1,2	0,947
SVC	BoW	2	0,917
SVC	Tf-Idf	1	0,962
SVC	Tf-Idf	1,2	0,959
SVC	Tf-Idf	2	0,955

A. Execution Times

During the execution of the classification algorithms, we recorded the training time for the training data as well as the processing time for the classification of the validation data. Table III presents the execution times (in seconds) of the above algorithms when running them with Tf-Idf and N-Gram (1,2) vectors, which yielded the best results across all datasets. The Random Decision Tree algorithm was not included due to its long processing time, while the Multinomial Naive Bayes algorithm was included despite its shorter execution time.

TABLE III
EXECUTION TIMES FOR SEVERAL ALGORITHMS WITH TF-IDF AND N-GRAM (1,2)

Algorithm	Dataset 1	Dataset 2
Bernoulli NB	0,13	0,44
Complement NB	0,16	0,48
Multinomial NB	0,15	0,45
Passive Aggressive	0,3	1,4
Ridge	0,9	3,8
SGD	2,5	18,5
SVM	1,7	4,9

B. Discussion

After conducting initial tests and optimizing the models, our proposed methods achieved F1 scores of up to 96.8% on the second dataset, which contained 26,000 articles, and up to 93.9% on the smaller first dataset, which contained 6,335 articles. Our findings indicate that the best choices for vectorization modes and classification algorithms, based on the performed tests, are Tf-Idf N-Gram (1,2) vectors, as well as Support Vector Machines (SVM) and Stochastic Gradient Descent (SGD) algorithms.

The linear algorithms and SVM, in particular, exhibited exceptional performance, demonstrating high accuracy in classification tasks while maintaining low learning and prediction times. Notably, the Multinomial Naive Bayes algorithm required the least computing power among the tested algorithms. Although its classification accuracy was not as high as some others, it remained acceptable, achieving an F1 score of 93.4% on the second dataset and 90% on the first dataset.

Several important observations emerged from our analysis:

- Among the vectorization methods tested, Tf-Idf consistently yielded the highest accuracy. Its ability to capture the importance of terms in the corpus appears to be well-suited for discriminating between fake and real news articles.
- Most models with the highest accuracy had N-Gram values of (1,2), except for the test using the Random Forest classifier algorithm, where the optimal N-Gram value was 1. This suggests that considering both unigrams and bigrams in the text representation can enhance the models' ability to capture relevant patterns and improve classification performance.
- In terms of processing time, Random Forest classifier exhibited the longest execution times, while Multinomial Naive Bayes classifier stood out as the best algorithm in this regard.
- The algorithms that consistently demonstrated the best results across the datasets were Support Vector Machines (SVM) and Stochastic Gradient Descent (SGD). Both algorithms exhibited short learning and prediction times. Notably, SVM exhibited slightly increased recall in both datasets, indicating its effectiveness in correctly identifying fake news instances.
- In conclusion, our proposed research highlights the effectiveness of linear algorithms, particularly SVM and SGD classifiers, in accurately detecting fake news. The choice of Tf-Idf N-Gram (1,2) vectors as the text representation method further enhances classification performance. While Multinomial Naive Bayes offers the advantage of low computational requirements, it sacrifices the accuracy metric. Moving forward, researchers in the field of fake news detection can leverage our findings to develop robust and efficient systems for combating misinformation, contributing to the ongoing efforts in ensuring the integrity of information dissemination in the digital age.

VI. CONCLUSIONS AND FUTURE WORK

In conclusion, this research paper explored the application of various machine learning algorithms and text vectorization methods for fake news detection. By conducting extensive experiments on multiple datasets, we obtained valuable insights and achieved high accuracy in classifying fake news articles. The results highlighted the effectiveness of several

algorithms, including linear algorithms and support vector machines, particularly when combined with Tf-Idf N-Gram (1,2) vectorization. We observed that the Naive Bayes algorithm exhibited the least computational power requirements. Stemming had a minimal impact on accuracy, and Tf-Idf vectorization consistently outperformed other methods. Additionally, the N-Gram value (1,2) proved to be optimal in most cases. Overall, we achieved F1-scores of up to 93.4% and 96.4% for the two datasets, respectively, demonstrating the effectiveness of our approach in detecting fake news articles.

Further research in the field of fake news detection can build upon the findings of this study and explore several avenues for future work. Firstly, investigating the effectiveness of deep learning models, such as recurrent neural networks or transformer-based architectures like BERT, could provide valuable insights into their potential for improving classification accuracy. Additionally, exploring ensemble learning techniques that combine multiple classifiers or leveraging semi-supervised and active learning approaches can enhance the robustness and efficiency of fake news detection systems [18], [25]. Moreover, incorporating more diverse features beyond textual content, such as user metadata, social network structures, or fact-checking data, can contribute to a more comprehensive understanding of fake news propagation and aid in more accurate detection. Finally, conducting studies to assess the generalization of the proposed models across different domains, languages, and cultural contexts would be beneficial for their practical deployment in diverse settings.

REFERENCES

[1] H. Allcott and M. Gentzkow. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2):211–236, 2017.

[2] F. T. Asr and M. Taboada. Big data and quality data for fake news and misinformation detection. *Big Data & Society*, 6(1):2053951719843310, 2019.

[3] A. Bessi and E. Ferrara. Social bots distort the 2016 U.S. presidential election online discussion. *First Monday*, 21(11), 2016.

[4] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc., 2009.

[5] A. Choudhary and A. Arora. Linguistic feature based learning model for fake news detection and classification. *Expert Systems with Applications*, 169:114171, 2021.

[6] E. Dritsas, G. Vonitsanos, I. E. Livieris, A. Kanavos, A. Ilias, C. Makris, and A. K. Tsakalidis. Pre-processing framework for twitter sentiment classification. In *15th International Conference on Artificial Intelligence Applications and Innovations (AIAI)*, volume 560, pages 138–149, 2019.

[7] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[8] A. Friggeri, L. Adamic, D. Eckles, and J. Cheng. Rumor cascades. In *8th International AAAI Conference on Weblogs and Social Media*, volume 8, pages 101–110, 2014.

[9] A. Guess, B. Nyhan, and J. Reifler. Selective exposure to misinformation: Evidence from the consumption of fake news during the 2016 u.s. presidential campaign. 2018.

[10] S. Hakak, M. Alazab, S. Khan, T. R. Gadekallu, P. K. R. Maddikunta, and W. Z. Khan. An ensemble machine larning approach through effective feature extraction to classify fake news. *Future Generation Computer Systems*, 117:47–58, 2021.

[11] N. Jindal and B. Liu. Review spam detection. In *16th International Conference on World Wide Web (WWW)*, pages 1189–1190, 2007.

[12] E. C. T. Jr, Z. W. Lim, and R. Ling. Defining "fake news" - a typology of scholarly definitions. *Digital Journalism*, 6(2):137–153, 2018.

[13] E. Kafeza, A. Kanavos, C. Makris, G. Pispirigos, and P. Vikatos. T-PCCE: twitter personality based communicative communities extraction system for big data. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1625–1638, 2020.

[14] E. Kafeza, A. Kanavos, C. Makris, and P. Vikatos. T-PICE: twitter personality based influential communities extraction system. In *2014 IEEE International Congress on Big Data*, pages 212–219, 2014.

[15] A. Kanavos, I. Perikos, P. Vikatos, I. Hatzilygeroudis, C. Makris, and A. K. Tsakalidis. Conversation emotional modeling in social networks. In *26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 478–484, 2014.

[16] A. Kanavos, E. Theodoridis, and A. K. Tsakalidis. Extracting knowledge from web search engine results. In *24th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 860–867, 2012.

[17] D. M. J. Lazer, M. A. Baum, Y. Benkler, A. J. Berinsky, K. M. Greenhill, F. Menczer, M. J. Metzger, B. Nyhan, G. Pennycook, D. Rothschild, M. Schudson, S. A. Sloman, C. R. Sunstein, E. A. Thorson, D. J. Watts, and J. L. Zittrain. The science of fake news. *Science*, 359(6380):1094–1096, 2018.

[18] I. E. Livieris, A. Kanavos, V. Tampakas, and P. E. Pintelas. A weighted voting ensemble self-labeled algorithm for the detection of lung abnormalities from x-rays. *Algorithms*, 12(3):64, 2019.

[19] A. Lyras, S. Vernikou, A. Kanavos, S. Sioutas, and P. Mylonas. Modeling credibility in social big data using LSTM neural networks. In *17th International Conference on Web Information Systems and Technologies (WEBIST)*, pages 599–606, 2021.

[20] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[21] W. McKinney. Data structures for statistical computing in python. In *9th Python in Science Conference*, volume 445, pages 51–56, 2010.

[22] P. Metaxas and S. Finn. The infamous# pizzagate conspiracy theory: Insight from a twittertrails investigation. 2017.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[24] A. Mohasseb, B. Aziz, and A. Kanavos. SMS spam identification and risk assessment evaluations. In *16th International Conference on Web Information Systems and Technologies (WEBIST)*, pages 417–424, 2020.

[25] A. Mohasseb and A. Kanavos. Factoid vs. non-factoid question identification: An ensemble learning approach. In *18th International Conference on Web Information Systems and Technologies (WEBIST)*, pages 265–271, 2022.

[26] T. E. Oliphant. *Guide to NumPy*, volume 1. Trelgol Publishing, 2006.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[28] G. Pennycook, T. D. Cannon, and D. G. Rand. Prior exposure increases perceived accuracy of fake news. *Journal of experimental psychology: general*, 147(12):1865, 2018.

[29] R. Řehuřek and P. Sojka. Software framework for topic modelling with large corpora. In *LREC Workshop New Challenges for NLP Frameworks*, 2010.

[30] C. Shao, G. L. Ciampaglia, A. Flammini, and F. Menczer. Hoaxy: A platform for tracking online misinformation. In *25th International Conference Companion on World Wide Web (WWW)*, pages 745–750, 2016.

[31] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu. Fake news detection on social media: A data mining perspective. *SIGKDD Explorations*, 19(1):22–36, 2017.

[32] S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, 13(2):22–30, 2011.

[33] S. Vernikou, A. Lyras, and A. Kanavos. Multiclass sentiment analysis on covid-19-related tweets using deep learning models. *Neural Computing and Applications*, 34(22):19615–19627, 2022.

[34] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, et al. Scipy 1.0-fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, 2020.

[35] S. Vosoughi, D. Roy, and S. Aral. The spread of true and false news online. *Science*, 359(6380):1146–1151, 2018.