Uncovering False Information: Techniques for Detecting Fake News

CSD358 Project

Aditya Y
Computer Science Of Engineering
Shiv Nadar University
Greater Noida, India
ay795@snu.edu.in

Kunisetty Teja Sai Srinivas
Computer Science Of Engineering
Shiv Nadar University
Greater Noida, India
ks165@snu.edu.in

Gautham Sreeramozu
Computer Science Of Engineering
Shiv Nadar University
Greater Noida, India
gs462@snu.edu.in

Geetham Surapaneni
Computer Science Of Engineering
Shiv Nadar University
Greater Noida , Indiay
gs272@snu.edu.in

Abstract*—The growth of social media has resulted in an increase in fake news, and this fake news has harmful repercussions. Because false news is such a big problem, attempts should be put in place to recognize all kind of fake news, but it is not that easy. Since establishing the accuracy of the facts in a story is complicated and challenging to do, even for professionals, it may be very subjective to spot fake news manually. Therefore, early automatic detection of fake news on the internet is crucial. In detecting fake news, machine learning algorithms, especially text classification algorithms, are used as a solution for this problem. Searching for the best method with high accuracy needs to be done continuously. This paper compares four main algorithms, namely Random Forest, Decision Tree, Rocchio Classifier, Gradient Boosting, k-Neighbours classifier, Adaboost Classifier, Logistic Regression (LR), and Naïve Bayes. The experiment was carried out using a dataset, consisting of fake news and real news. The performance matrix of each algorithm was evaluated with accuracy, recall, precision, and F1-score as the harmonic mean. The results showed that KNeighborsClassifier was able to separate fake news and real news with the highest F1-score reaching 91,5%. This paper also proposes a framework of detecting fake news that can be implemented on public websites.

***Keywords*—fake news detection, text classification, Random Forest, Decision Tree, Adaboost, k-Neighbour, Gradient Boosting, Rocchio, Logistic Regression, Naïve Bayes**

I. Introduction

The current technological era has allowed for the rapid and often uncontrollable dissemination of misleading information, even news that should be believed.

Fake news disseminates inaccurate or slanted information with the intent to deceive us and unquestionably has a detrimental effect on society. For example, often fake news is information about job openings that encourages fraud to defraud potential candidates of their money.

The detection of fake news on public and government websites is crucial, particularly in Indonesia where the characteristics of Indonesian news differ from other languages. Previous studies have compared different methods, such as SVM and SGD, with better results found using SGD. Research using SVM with TF-IDF to form word vectors found an increase in recall and precision, emphasizing the importance of balanced training data. Other studies have used Naïve Bayes and a combination of six methods, including NB, SVM, PA, RF, SGD, and LR, achieving varying levels of accuracy. Logistic regression was suggested as a simpler model with an average accuracy of more than 90% in classifying fake news. Overall, finding the best method for classifying fake news in Indonesian news is important in implementing automatic fake news detection.

II. Literature

Stahl (1) conducted a study with the objective of examining the existing and current techniques used for detecting fake news in textual formats. In the study, Stahl proposed a three-part methodology that involves using Naïve Bayes (NB), as a means of detecting fake news on social media. The goal of the study was to provide guidance to researchers on which combination of methods to use for accurate and dependable detection of fake news on social media platforms.

In a study referenced as [4], the researchers employed Logistic Regression to identify and detect fake news. The main objective of their research was to develop a model for detecting false news, and the efficacy of Logistic Regression was evaluated for this purpose. Subsequently, a web application was developed that enabled users to input the news text or URLs for analysis. The study findings indicated that the approach used in this model, which involved stance detection, resulted in a more accurate detection of fake news compared to using TF-IDF vectorization in constructing the model for detecting fake news.

Researchers in [3] employed a machine learning algorithm, specifically the Decision Tree algorithm, for detecting false news. The dataset was split into smaller subsets, and the Decision Tree was among the machine learning classifiers utilized for this task. The researchers used these classifiers as a means of detecting fake news.

The goal of fake news identification, according to the author Monther Aldwahedi et al. [4], is to develop a method that users may use to identify and filter out websites that contain inaccurate and misleading information. A simple and carefully selected feature of the title and post to accurately identify fake posts. The experimental results showed a 91.5% accuracy using the k-Neighbouring (kNn) classifier.

The majority of the articles that were reviewed did not place emphasis on the aspect of feature extraction. The identification of words that are significantly associated with fake news is crucial for detecting fake news, and this objective can be attained through the implementation of various feature extraction techniques. The present study delved into several techniques for feature extraction and made use of a comprehensive dataset that amalgamates numerous datasets on fake news.

III. METHODOLOGY:

The primary objective of this study is to develop machine learning classifiers that can accurately distinguish between real and fake news articles.

A. Dataset Usage:

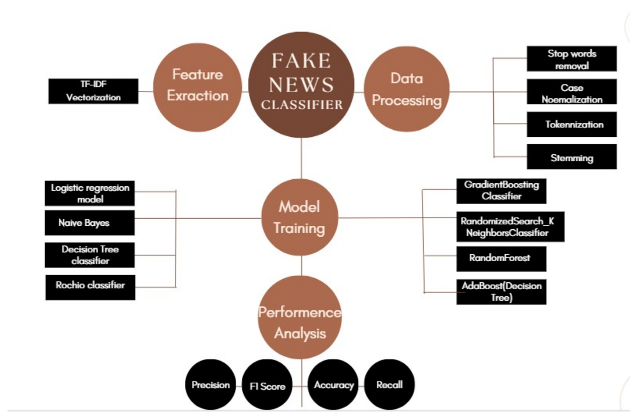
The dataset provided includes two types of news articles: real news and fake news. The real news articles were collected by crawling Reuters.com, while the fake news articles were obtained from a variety of sources, including Wikipedia and Politifact. The collection covers a wide range of topics, although the majority of the articles are related to politics and foreign events

B. Data Pre-Processing:

The first step in text processing classification is preprocessing, which involves transforming unstructured text into a word representation that can be modeled by the classifier algorithm. This process consists of four phases, namely case folding, cleaning, tokenizing, and stop word removal. Case folding changes capital letters to lowercase or vice versa. Data cleaning involves removing irrelevant data such as certain punctuation marks (,:'?/;"). Tokenizing is the process of splitting serialized words in sentences into tokens. Stop word removal is a crucial step in which commonly used words without significant meaning or information, such as "which," "in," "to," "is," "at the end," "if," and "from," are eliminated.

In order to use text data for Natural Language Processing (NLP), the text undergoes cleaning and preparation, known as data preprocessing. This process involves various steps, such as text normalization, stop word removal, tokenization, and stemming.

- ➢ Text normalization involves cleaning the text and converting it into a consistent format. This includes removing special characters such as '@', '$', '*', and URLs, and converting all the letters to lowercase.
- ➢ Stop-word removal is the process of eliminating highly repetitive words that do not add significant meaning to the text, such as 'a', 'the', 'is', and 'are'.
- ➢ Tokenization involves splitting sentences into individual tokens or words. For instance, the sentence 'no work today' would be tokenized into 'no', 'work', and 'today'.
- ➢ Stemming is the process of stripping the suffixes and prefixes of words to extract the base or root verb. For instance, the word 'studying' would be stemmed to 'study'.



C. *Features Extraction*

Before we can train our machine learning models, we must identify the most important words in the data that will yield the best results. This process is called feature extraction because we use words as the features to train our models. In this study, we utilized Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer feature extraction technique. Once we cleaned the data, we applied this technique to extract features from the data.

TF-IDF Vectorizer:

TF-IDF is a technique used to represent words as vectors based on their frequency and relevance to a particular document. It involves calculating the term frequency (TF) of each word in the document, and then multiplying it by the inverse document frequency (IDF). This is expressed in the below equations.

$TF - IDF(t, d) = TF(t, d) * IDF(t)$

$IDF(t) = \log (n\ DF(t)) + 1$

where TF-IDF(t,d) calculates the frequency of each word in a document (TF) and IDF considers how often the word appears in other documents.

By doing so, it highlights words that are highly relevant to a particular document by assigning them high values, while common words that appear in all documents are given low values even if they are frequently used.

This technique allows us to identify the most meaningful words in a document while filtering out commonly used words that are not as relevant to the document's content.

D. *Proposed Models:*

- Logistic Regression is a technique that can estimate the likelihood of a binary outcome using one or more independent variables. This approach is frequently employed in classification problems that involve binary or multiple classes. Contrary to standard linear regression, logistic regression does not rely on a linear relationship between the dependent variable and independent variables. Instead, the results are expressed as binary values of 0 or 1, signifying the chance of an event taking place. This unique aspect of logistic regression allows it to be effective for predicting categorical variables.
- Random Forest (RF) is a machine learning algorithm that is used in supervised learning contexts, such as classification and regression problems. The RF algorithm comprises a group of decision trees that work together to make predictions. These predictions are then combined and weighted to generate a more accurate and reliable prediction. The goal of using RF is to improve the performance of the algorithm by combining multiple decision trees instead of relying on a single decision tree for predictions. By using a collection of decision trees, RF is able to generate more accurate and reliable predictions than a single decision tree.
- The Gradient Boosting Classifier is a machine learning method that builds an ensemble of weak prediction models, typically decision trees, to create a strong predictor. When decision trees are used as the weak learner, the resulting model is known as gradient boosted trees, which generally outperforms random forest. This approach constructs the model in stages, similarly to other boosting techniques, but it allows the optimization of any differentiable loss function. After fitting the model, we will assess its performance by comparing the score and examining the confusion matrix. Once all classifiers have been fitted, we will discard the models and vectorization models, which will be utilized to connect the model with the webserver
- The K-Nearest Neighbors (KNN) algorithm is a supervised machine learning technique used for classification and regression problems. The algorithm is provided with a set of input data, labeled with their corresponding output values, and it uses this information to create a function that can predict the output label of a new, unlabeled input. The algorithm works by identifying the 'nearest neighbors' of the input data point. The value of K represents the number of nearest neighbors that the algorithm considers.
- Rocchio Classification is a form of relevance feedback that employs the centroid of the relevant document class as its primary component. This centroid corresponds to the most significant element of the Rocchio vector in relevance feedback. To create good class boundaries, Rocchio classification utilizes centroids as defining points. It calculates the centroid for each class and computes the distance from each centroid to a new text data. The data point is then assigned to the nearest centroid.
- The Decision Tree algorithm mimics the process of decision-making in our daily lives, where decisions are made based on available resources. The algorithm constructs a tree structure in which each internal node represents a decision, and the leaf nodes represent the outcome of the decision. The prediction is made by traversing the tree from the root to the leaf node that best matches the input data.

E. Evaluation Metrics

1) Accuracy

The ratio of correct predictions is calculated using Equation (1) by dividing the total number of correct predictions by the total number of predictions made.
$Accuracy = TP + TN\ TP{+}FN{+}FP{+}TN$

2) Precision

A measure of the number of positive, “fake”, instances out of all the instances that were predicted as positive which is calculated by.

$Precision\ (P) = TP/(\ TP + FP\ ).$

where TP is the number of truly predicted samples of the positive class, over the total number of samples of the positive class.

3) Recall

It predicts the positive class predictions over the number of positive instances in the dataset itself regardless of whether they were correctly predicted as positive or not. This indicates how successful the model is at predicting the true positives which is calculated by:

$$Recall\ (R) = TP\ /(\ TP + FN\ )$$

,where TP is the number of truly predicted samples of the positive class, over the total number of samples that were predicted as a positive class by the model.

4) F1-score

A measure that combines the precision and recall values in a single number which is calculated using Equation (4) [20].

$$\text{F-Measure} = 2 \times \frac{P * R}{P+R}$$

IV. *Experimentation and Observations*

To build the proposed models, Python 3.9 programming language was used. The experiments had been performed online using Jupyter Notebook. Nine different experiments were performed to compare the results.

A. Dataset brief:

The dataset used for training and testing was obtained by scraping news articles from various websites. The articles were then manually categorized as either 'fake' or 'real'. The dataset contains 10387 articles of fake news and 10303 real news articles.

Ratio of real and fake news:

fake 0.50203

real 0.49797

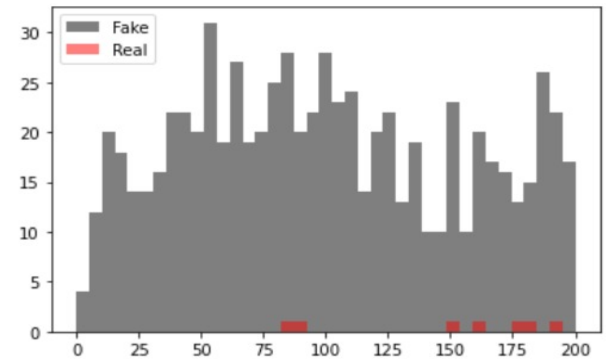
Name: label, dtype: float64

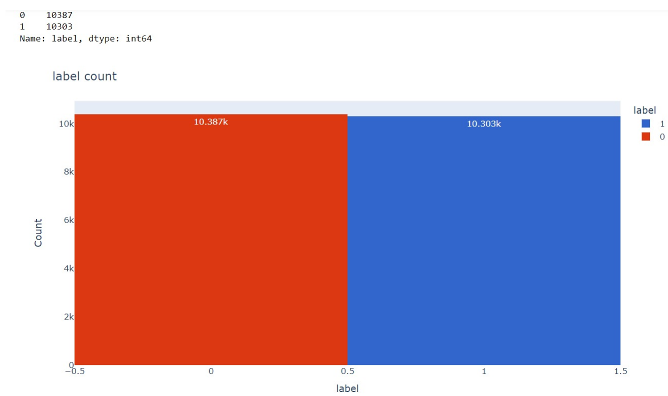
Fake : 10387

Real :10303

((20690, 3) => depicts the rows and columns.

The Histogram shown below shows that fake news text length is much larger that real news text length



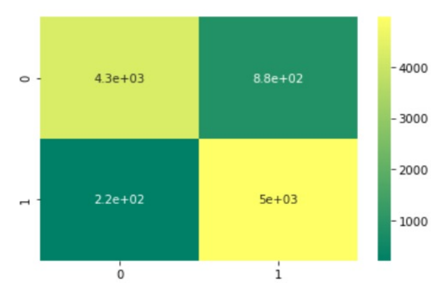


B. Results:

Performances results of each model are presented below in the form of graphs:

1.) VotingClassifiers with Naive Bayes:

	precision	recall	f1-score	support
0	0.95	0.83	0.89	5154
1	0.85	0.96	0.90	5191
accuracy			0.89	10345
macro avg	0.90	0.89	0.89	10345
weighted avg	0.90	0.89	0.89	10345



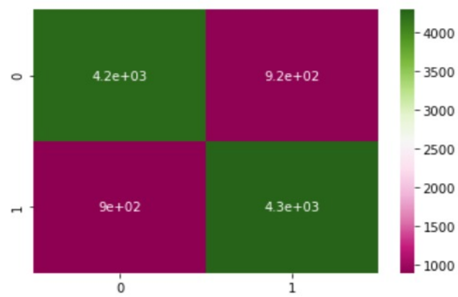
This model combines multiple models using a voting strategy, with one of the models being a Naive Bayes classifier. It has a high accuracy score on the test set of 0.893475

.

2.)Decision Tree

i) Basic: This is a basic decision tree model with a training score of 0.819333 and a test score of 0.824360. It may be prone to overfitting on the training set.

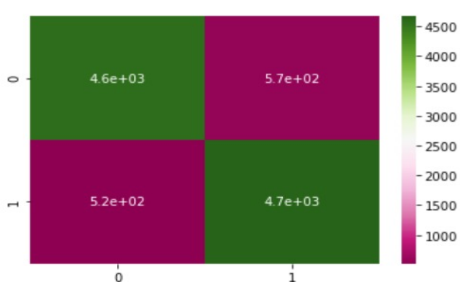
	precision	recall	f1-score	support
0	0.82	0.82	0.82	5154
1	0.82	0.83	0.83	5191
accuracy			0.82	10345
macro avg	0.82	0.82	0.82	10345
weighted avg	0.82	0.82	0.82	10345



This is a basic decision tree model with a training score of 0.819333 and a test score of 0.824360. It may be prone to overfitting on the training set.

ii.) Addaboost with Decision Tree:

	precision	recall	f1-score	support
0	0.90	0.89	0.89	5154
1	0.89	0.90	0.90	5191
accuracy			0.89	10345
macro avg	0.89	0.89	0.89	10345
weighted avg	0.89	0.89	0.89	10345

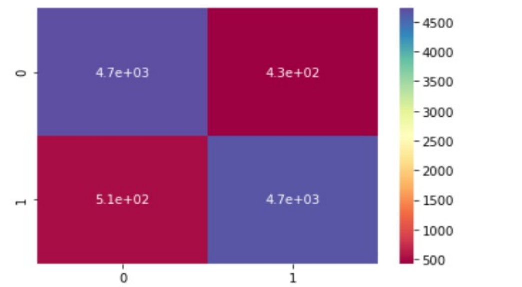


This is an AdaBoost model that uses decision trees as the base classifier. It has a high training score of 0.993620 and a test score of 0.894442.

3.) RandomForest:

i) RandomizedSearchCV:

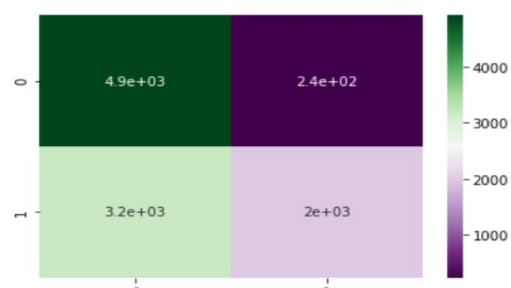
	precision	recall	f1-score	support
0	0.90	0.92	0.91	5154
1	0.92	0.90	0.91	5191
accuracy			0.91	10345
macro avg	0.91	0.91	0.91	10345
weighted avg	0.91	0.91	0.91	10345



Random_forest with RandomSearchCV: This is a random forest model that uses randomized search for hyperparameter tuning. It has a training score of 0.910875 and a test score of 0.909135, indicating good performance on the test set.

ii.) AddaBoost with Random Forest:

	precision	recall	f1-score	support
0	0.61	0.95	0.74	5154
1	0.89	0.38	0.54	5191
accuracy			0.67	10345
macro avg	0.75	0.67	0.64	10345
weighted avg	0.75	0.67	0.64	10345

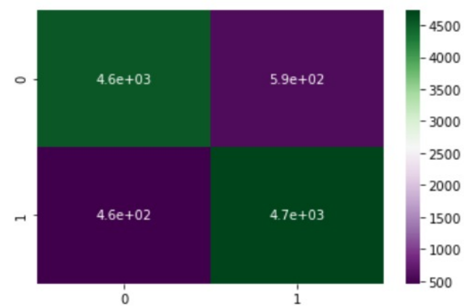


This is an AdaBoost model that uses random forests as the base classifier. It has a lower training score of 0.704688 and a lower test score of 0.668246, indicating poor performance on the test set.

Note: Adaboost can be used for fake news classification by training the algorithm on labeled news articles to identify patterns that distinguish between real and fake news. The classifier can then be used to analyze new articles and determine whether they are real or fake. However, its accuracy depends on the quality of the training data and the features used, and it may not be able to account for all nuances and biases.

4.) Gradient_boost with RandomSearchCV:

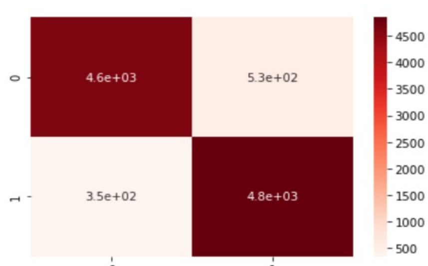
	precision	recall	f1-score	support
0	0.91	0.89	0.90	5154
1	0.89	0.91	0.90	5191
accuracy			0.90	10345
macro avg	0.90	0.90	0.90	10345
weighted avg	0.90	0.90	0.90	10345



This is a gradient boosting model that also uses randomized search for hyperparameter tuning. It has a high training score of 0.993620 but a lower test score of 0.898695, suggesting some overfitting on the training set.

5.) RandomizedSearch_KNeighborsClassifier:

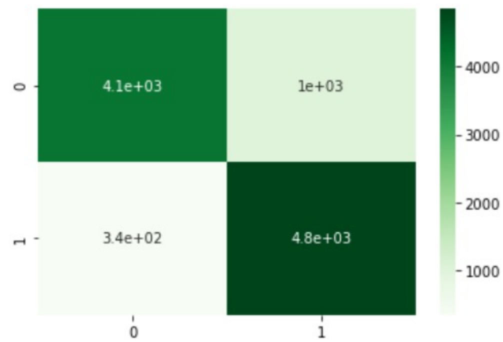
	precision	recall	f1-score	support
0	0.93	0.90	0.91	5154
1	0.90	0.93	0.92	5191
accuracy			0.92	10345
macro avg	0.92	0.92	0.92	10345
weighted avg	0.92	0.92	0.92	10345



This is a k-nearest neighbors model that uses randomized search for hyperparameter tuning. It has a high training score of 0.922571 and a high test score of 0.915321, indicating good performance on the test set.

6.) Logistic Regression Model:

	precision	recall	f1-score	support
0	0.92	0.80	0.86	5154
1	0.83	0.93	0.88	5191
accuracy			0.87	10345
macro avg	0.88	0.87	0.87	10345
weighted avg	0.88	0.87	0.87	10345



7.) Rocchio Classifier:

	precision	recall	f1-score	support
0	0.94	0.48	0.64	5154
1	0.65	0.97	0.78	5191
accuracy			0.73	10345
macro avg	0.80	0.73	0.71	10345
weighted avg	0.80	0.73	0.71	10345

Rochio classifier,Accuracy Score: 0.7280811986466892

V. Conclusion and Limitations

After computing the evaluation metrics for each models, we compare the training and testing scores of each model and represent it in the below table:

	Train score	Test score
votingclassifier_NB	0.895795	0.893475
DecisionTreeClassifier	0.819333	0.824360
Random_forest RandomSearchCV	0.910875	0.909135
Graident_boost RandomSearchCV	0.993620	0.898695
RandomizedSearch_KNeighborsClassifier	0.922571	0.915321
Addaboost(Decision Tree)	0.993620	0.894442
AddaBoost(Random Forest)	0.704688	0.668246
Rochio classifier	0.866409	0.868826
LogisticRegression	0.866409	0.868826

Based on the training and test scores you have provided, it appears that the Random_forest RandomSearchCV and RandomizedSearch_KNeighborsClassifier models have the best performance on the test set, with test scores of 0.909135 and 0.915321, respectively. The Gradient_boost RandomSearchCV model also has a high test score of 0.898695. However, it's important to keep in mind that other factors, such as the specific task and dataset, may also impact which model is the "best" choice for a particular problem

There is a demand for a system to detect fake news to protect society from scams and all that this type of news may lead to from public disturbance. By collecting the needed data and using ML algorithms, models can be built to solve these problems and identify anomalies. Still, this issue has not been completely resolved. Even though there have been several attempts to address the issue, existing techniques need modification and improvements.

In this work, we sought to address this issue by developing ML models that can detect different types of fake news. We applied Nine different features extraction techniques to improve our models' performance.

For future work, we aim to implement deep learning techniques to handle fake news. In addition, we would attempt to generate new datasets related to fake news with new features and study the effect of the different features on the model's detection's performance.

VI. References:

[1] K. Stahl, "Fake news detection in social media."

[2]. Muhammad Syahmi Mokhtar, Yusmadi Yah Jusoh, Novia Admodisastro, Noraini Che Pa, Amru Yusrin Amruddin. Fakebuster: Fake News Detection System Using Logistic Regression Technique In Machine Learning. International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-9, Issue-1, October 2019

[3]. C. M. M. Kotteti, X. Dong, N. Li and L. Qian, "Fake News Detection Enhancement with Data Imputation," 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), 2018.

[4] Aldwairi, Monther, and Ali Alwahedi. "Detecting fake news in social media networks." Procedia Computer Science 141 (2018): 215-222.

[5]V. Sari, F. Firdausi, and Y. Azhar, "Comparison of Arabica Coffee Quality Predictions Using the SGD Algorithm, Random Forest and Naive Bayes, (In Bahasa Indonesia)" Edumatic J. Pendidik. Inform., vol. 4, no. 2, pp. 1–9, 2020, doi: 10.29408/edumatic.v4i2.2202.

[6] M. A. Iqbal, "Application of Regression Techniques with their Advantages and Disadvantages," Elektron Mag., no. September, 2021.

[7] I. Santoso, I. Yohansen, N. Nealson, H. L. H. S. Warnars, and K. Hashimoto, "Early investigation of proposed hoax detection for decreasing hoax in social media," 2017 IEEE Int. Conf. Cybern. Comput. Intell. Cybern. 2017 - Proc., vol. 2017-Novem, pp. 175– 179, 2018, doi: 10.1109/CYBERNETICSCOM.2017.8311705.