Journal of Information Systems and Informatics
Vol. 5, No. 3, September 2023 e-ISSN: 2656-4882 p-ISSN: 2656-5935
DOI: 10.51519/journalisi.v5i3.548 Published By DRPM-UBD

Fake News Detection Using Optimized CNN and LSTM Techniques

Emmy D. Ajik[1], Georgina N. Obuandike[2], Faith O. Echobu[3]

[1, 2, 3]Faculty of Computing, Federal University Dutsinma, Katsina State, Nigeria
Email: adanny@fudutsinma.edu.ng[1], gobunadike@fudutsinma.edu.ng[2],
fadebiyi@fudutsinma.edu.ng[3]

Abstract

Concerns have been raised about the social consequences of fake news as it has spread rapidly on online platforms. It is critical to detect and mitigate the spread of fake news in order to maintain a healthy community conversation. There is a need to put more effort into the identification of fake news as more people use the internet, especially as more internet-enabled gadgets become more widely available and inexpensive. With the help of two Neural Network techniques: long-short-term memory (LSTM) and Convolutional Neural Network (CNN). This research proposes novel Deep Learning methods for identifying fake news using two datasets. These methods were considered for this research because they had proven to be successful in earlier studies that had been looked at. Finding the best-performing optimal models is the goal of this study. HyperOpt Technique was used for Neural Network model. The performance of the optimized models was compared with the performance of the models without optimization. The results obtained showed that for both datasets, CNN and LSTM performed better when training the models with the optimal values with an average difference of 12.7% for Accuracy, 11.9% for Precision, 12.3% for Recall and 15.4% for F1-Score.

Keywords: Optimization, HyperOpt Technique, Fake News, Convolutional Neural Network, Long Short-Term Memory

1. INTRODUCTION

Fake news is misleading information or manipulated news that contains misinformation and is communicated through both traditional and non-traditional media channels, such as print and television [1]. The prevalence of fake news on online platforms poses a significant problem, as it spreads misinformation and negatively impacts society. Machine learning models have shown promise in detecting fake news by analyzing textual features. However, to achieve optimal performance and accuracy, it is essential to effectively tune the hyperparameters of these models. The problem addressed in this study is the limited utilization and exploration of hyperparameter tuning techniques to optimize models for fake news detection. Existing research often focuses on developing machine learning models without adequately optimizing their hyperparameters, resulting in suboptimal performance and limited effectiveness in identifying fake news. Therefore, there is a need to investigate and develop optimized models for fake

news detection by leveraging hyperparameter tuning techniques. The purpose of the study is to contribute to the development of strong and effective strategies for combatting fake news while fostering an informed and trustworthy information environment.

Fake news detection on several platforms for social networking that employ the Naive Bayes classifier was conducted [2]. They gathered news from social media platforms such as Twitter, Facebook, and others. Because the material on this site is not entirely reputable, accuracy is quite poor. A study by [3] included online data mining in their solution approach. They developed deep learning models using LSTM and Glove Feed-Forward NN in conjunction with various word vector representations. The Glove Feed Forward Network model produced an accuracy of 83.3% and 84.3% without and with mined features respectively while the LSTM had an accuracy of 83.7% and 91.3% without and with mined features respectively. The outcome confirms their belief that tackling the fake news problem with a data mining component will provide better results than a solely NLP strategy.

Similarly, [4] goal was to use machine learning technologies in identifying fake news. Their study uses three prevalent methods: Naive Bayes (NB), Support Vector Machine (SVM), and Neural Network (NN). Normalization is an important step in cleaning up data before using machine learning to categorize the data. The results demonstrated that the Nave Bayes algorithm detects fake news with an accuracy of 96.08%. The neural network and the Support Vector Machine (SVM), two more advanced approaches, achieved an accuracy of 99.90%. Support Vector Machines (SVM), Naive Bayes' (NB), Decision Trees (DT), Logistic Regression (LR), and Artificial Neural Networks (ANN) were utilized to detect fake news with data gotten from kaggle.com [5]. They extracted features using the Count Vectorizer and the TF-IDF Vectorizer techniques. SVM outperforms TF IDF with an accuracy of 92.8%. With accuracies of 91.6 and 91.0, respectively, logistic regression outperforms both the count vectorizer and the TF IDF. [6] got data from Kaggle to detect fake news by utilizing CNN with text only, CNN with text and title, and CNN with text and author. For data preparation and feature extraction, they used convolution network layers. The combined CNN attained an accuracy of 96%. [7] created Deep Learning techniques for detecting fake news. They used dataset obtained from https://www.kaggle.com/c/fake-news/data to detect false news with a Feed forward neural network, CNN with one convolutional layer, CNN with multiple convolutional layers, and LSTM. They used the word embedding method to prepare the data. With the addition of convolutional layers, CNN achieves 97.15% accuracy.

A study by [8] Using the LIAR and PoliFact datasets, a hybrid technique was employed that included Natural Language Processing, Deep Learning, and Semantics. Multinomial Naive Bayes (MNB), Stochastic Gradient Boosting (SGD), Linear Regression (LR), Decision Tree (DT), and Support Vector Machine

were used to train the model in the study. Deep learning models such as CNN, Basic LSTM, Bi-LSTM, GRU, and CapsNet were also used to train the model. According to the study, CapsNet fared better than the other models when utilizing the LIAR dataset, with an accuracy rate of 64.7%. Another study using the LIAR dataset was conducted by [9], with an accuracy of 0.61, SVM and Bi-LSTM performed best. [10] in their research, they evaluated standard machine learning models to pick the best, to build a model using a supervised machine learning algorithm that can classify fake news as either true or false. They used Random Forests, XGBoost, K-Nearest Neighbours (KNN), Naive Bayes, Decision Tree, and SVM to analyze the LIAR-PLUS Master dataset. With more than 75% accuracy using the Count Vectorizer and Tiff Vectorize features, XGBOOST was the most accurate, followed by SVM along with Random Forest with roughly 73% accuracy.

Existing research frequently overlook the critical step of fine-tuning hyperparameters, resulting in suboptimal performance in detecting fake news. Furthermore, there has been little investigation into other hyperparameter tuning strategies other than grid search or random search. This limits our understanding of the most successful strategies for optimizing models in the context of fake news identification.

2. METHODS

The study will be implemented using the steps as depicted in Figure 1. Each of the steps is further discussed.

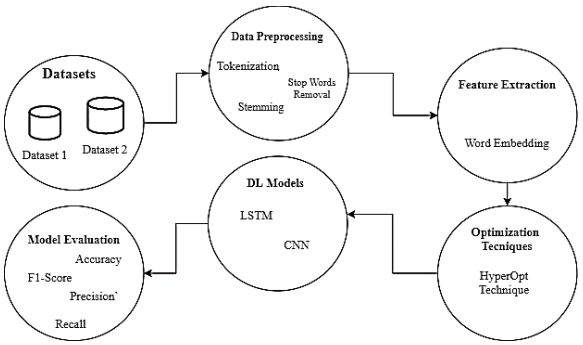

Figure 1: System Architecture

2.1. Data Collection

Data was obtained in CSV format from secondary sources available on the internet. Two datasets are selected, requested, and downloaded. The two datasets were collected with permission from the owners. Dataset 1 contains nearly 20,000 pieces of news about the American presidential election, 11,941 of which are

labelled fake news and 8,074 of which are designated authentic news. The dataset is freely available online and may be used with the author's permission [11]. Megan Risdal on Kaggle gathered text and metadata from over 240 websites to create fake news content, whereas real news is collected from popular authoritative news websites such as the Washington Post, New York Times, and others. Title, content, image, author, and website are just a few of the several pieces of information in the dataset. This research will focus on the text which is the news content and the type of column which specifies if the news is fake or real. Dataset 2 contains over 6300 news mainly about politics around America. The dataset contains 3171 news labelled Real and 3164 news labelled Fake. The dataset is available on Google Drive for free download. It was authored by [12]. Only the Text and Label columns will be used for this research.

2.2. Data Cleaning and Preprocessing

Before the dataset will be ready for the analysis phase, the dataset may contain missing values, inadequate attributes data types, valueless attributes, and any other problem that may affect the performance of the data during processing. The datasets will have to undergo:

a. **Dataset Cleaning:** This will remove all unwanted parts of the data. Capital letters and Urls are cleared from the text using **unescape** library imported from the **HTML package**. A part of the noisy data was manually removed. Most of the noisy data removed were data lying redundantly on other columns or extra comments made on new columns.
b. **Dataset Manipulation: Panda** DataFrame objects are used to read the datasets. The hold-out validation Method was used for splitting the data into train and test sets, with 70% and 30% allotted to the training set and test set respectively. This was done using the **train_test_split** library imported from the Scikit-Learn package. The random state for splitting was set to 20.
c. **Tokenization**: This involves transforming a document into tokens. A document can be separated into smaller parts of it, such as sentences or words, discarding characters from punctuation, thus generating tokens.
d. **Capitalization:** This function transforms all the text into a similar case either all uppercase or lowercase. This is to ensure that the same word is not placed into different tokens. This was achieved by transforming each word to a lambda property and using the function **x.lower()** to change the case.
e. **Lemmatization:** This is the process of putting the word in its base form, that is, ignoring tense, plural words, and their gender. It's the process of obtaining the base form of a word. For this research, the **WordNetLemmatizer** library was imported and used.
f. **Stop Words** Stop words are words that can be considered of little meaning, that is, it does not have much relevance in a text. They are words that usually

do not help in a search and thus can be excluded. In general, they are prepositions, conjunctions, determinants, and some connecting verbs. Examples are a, an, the, and, at, be, by, for, from, etc. For this research, **the NLTK toolkit** library was imported and used.

2.3. Feature Extraction

Word Embedding was used to extract and transform text data into vectors. By using the Label Encoding approach, category columns may be transformed into numerical ones so that they can be fitted by machine learning models, which primarily require numerical data.

2.4. Hyperparameter Tuning

Most machine learning models contain hyperparameters that must be modified to be customized to your dataset. Most of the time, some best practices for setting values for certain of these parameters are known. The combinations of interacting hyperparameters present the most difficult challenge. Many studies have been conducted to propose various rules for configuring hyperparameters. A better method would be to objectively examine multiple model hyperparameter values and choose the subset that leads to the best performance on a certain dataset [13]. This is referred to as Hyperparameter Optimization or Tuning. It is also known as a scenario where a set of best hyperparameters is chosen [14]. On a given independent data set, the objective of hyperparameter optimization is to identify the combination of hyperparameters that leads to the optimal model with a given loss function. [15].

The result of this process is a single set of parameters that performs best which will be used to train your model. Machine learning models also have parameters that are not to be confused with hyperparameters. Parameters are automatically learned while hyperparameters are set manually. There exist several optimization algorithms that exist and can be used. A few of these methods are Grid Search, Random Search, and HyperOpt Optimisation. This study used HyperOpt technique. James Bergstra created Hyperopt, a strong python package for hyperparameter optimization. For parameter tweaking, Hyperopt utilises a variant of Bayesian Optimisation that allows you to obtain the optimal parameters for a given model. On large scales, it can optimize a model with hundreds of parameters.

2.5. Models

Two deep learning models were used for this study, CNN and LSTM. Details of each architecture are as follow.

2.5.1 CNN

CNNs are created by modifying multilayer perceptron. All neurons in one layer have connections to all the neurons in the layer that follows in fully linked networks, which are also referred to as Multi-layered Perceptron. Because its neurons are completely connected, these networks are at risk of overfitting of the data. Methods for preventing overfitting or regularization often entail penalizing training parameters (such as weight decay) or reducing connectivity (skipped connections, dropout, and so on). CNNs employ a novel regularization method that takes advantage of the hierarchical layout of the data to generate patterns with increasing complexity utilizing simpler and smaller structures embedded in its filters. As a result, CNNs rank at the bottom of the complexity and connectivity scale. A convolutional neural network is made up of three layers, a layer of input, hidden layers, and an output layer. In a neural network made up of feed-forward neurons, hidden layers are those layers whose inputs and outputs are hidden by the final convolution and activation function. Convolutional layers are present in a convolutional neural network's hidden layers. A feature map is formed as the kernel for convolution advances into the input matrix of this layer, adding onto the input of the following layer. Following that, other layers are added, such as pooling layers, completely linked layers, and normalization layers [16]. The physical architecture of the proposed CNN models which was used in this study is seen from Figure 2.

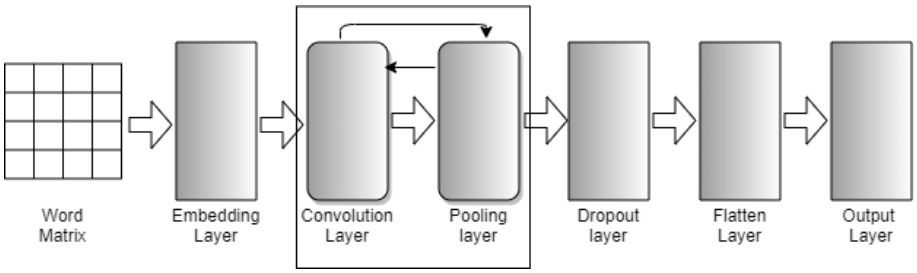

Figure 2. Proposed CNN Architecture

2.5.2 LSTM

Long Short-Term Memory (LSTM) is a Recurrent Neural Network (RNN) variation. RNNs can solely retain recent information, whereas LSTMs can handle long-term data. Furthermore, an RNN model experience an issue called the vanishing gradient problem when considering long sequence data; nevertheless, LSTM may avoid this challenge while training. This model can recall prior long-term time-series data and allows automatic control in the cell state for maintaining useful and discarding irrelevant properties. The three gates that regulate features in an LSTM model are the input gate, the forget gate, and the output gate. The input gate carries the function of allowing new data to enter into the cell state. The

forget gate removes irrelevant information from the cell state. The output gate, which controls the information taken from the cell state, then determines what will be the next hidden state. An LSTM model may automatically save or wipe recorded memory using these gates. The physical architecture of the proposed CNN models which was used in this study is seen from Figure 3.

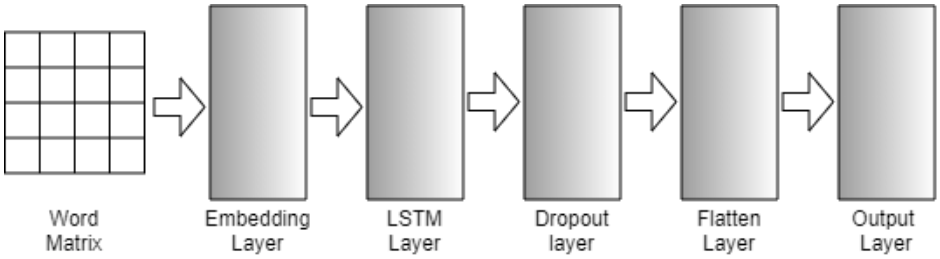

Figure 3. Proposed LSTM Architecture

2.6. Training Conditions

Google COLAB is used to implement the model. Python is used to write the pre-processing and classifier routines. Keras was utilized to implement deep learning. The Scikit-Learn package was used to analyse the data, evaluate the results, and create classifiers. The Matplotlib package was used to plot graphs, while Pandas and Numpy were utilized to read datasets and handle arrays, respectively. Hyperparameters used by [17] were used as default. This is shown in Table 1 and Table 2.

Table 1. CNN Model default parameters

Hyperparameter	Value
Number of epochs	10
Number of filters	200
Number of units in fully connected layer	30
Batch Size	50
Filter Size	[3,4,5]
Dropout rate	0.5
Learning rate	0.001

Table 2. LSTM model default parameters

Hyperparameter	Value
LSTM hidden state dimension	200
Number of units in fully connected layer	30
Learning rate	0.001
Number of epochs	10
Batch Size	50

During the training of the DL models, the following conditions were adopted due to system limitations available to the researchers and for a faster result:

a. **Early Stopping**: This is implemented to stop training when monitored metrics stop improving.
b. **Steps per Epoch**: This was set to 100.
c. **Validation Steps**: This was set to 10.

2.7. Model Evaluation

A model can be evaluated using various metrics. The evaluations were carried out using:

a. **Accuracy:** Accuracy is calculated by dividing the number of classifications correctly labelled by the total cases present.

$$Accuracy = \frac{TP+TN}{P+N} \quad (1)$$

b. **Precision:** This is a unit of measurement for exactness. It is the proportion of positive tuples that are accurately labeled as such.

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

c. **Recall:** This measures the number of positive class samples in the dataset that were correctly identified by the model.

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

d. **F1-Score:** This metric assesses the precision of a model. It combines the accuracy and recall scores of a model.

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

3. EXPERIMENTAL RESULTS

The model was trained as mentioned in the previous part, and the results are given in this section.

3.1 LSTM

Table 3 shows the performance of applying LSTM to Dataset 1 and Dataset 2. Without Optimisation, for Dataset 1 the result obtained is Train Accuracy 81.73%, Test Accuracy 77.16%, Precision 78.89%, Recall 77.16% and F1-Score 74.99%.

After the Hyperopt technique has been done, the optimal values obtained are summarized in Table 4 and the result obtained from training the model is Train Accuracy 95.6%, Test Accuracy 89.96%, Precision 89.61%, Recall 83.84% and F1-Score 86.63%. Without Optimisation, for Dataset 2 the result obtained is Train Accuracy 76.50%, Test Accuracy 73.01%, Precision 76.13%, Recall 73.01% and F1-Score 72.06%. After the Hyperopt technique has been done, the optimal values obtained are summarized in Table 4 and the result obtained from training the model is Train Accuracy 90.28%, Test Accuracy 86.48%, Precision 90.27%, Recall 90.31% and F1-Score 90.28.

Table 3. LSTM Result for Dataset 1 and Dataset 2

	Train Accuracy	Accuracy	Preci-sion	Recall	F1-Score
DATASET 1					
No Optimization	81.73	77.16	78.89	77.16	74.99
Optimization	95.6	89.96	89.61	83.84	86.63
DATASET 2					
No Optimization	76.50	73.01	76.13	73.01	72.03
Optimization	90.28	86.48	90.27	90.31	90.28

Table 4. LSTM Hyperparameter Values - Optimized for Dataset 1 and Dataset 2

Hyperparameters	CuDNNLSTM	Dropout	Batch_Size	Epoch
Optimal Values Dataset 1	64	0.16	50	14
Optimal Values Dataset 2	30	0.22	40	10

From the results obtained, it can be observed that the LSTM model with optimization performs better than the model without optimization.

3.2 CNN

Table 5 shows the performance of applying CNN to Dataset 1 and Dataset 2. Without Optimisation, for Dataset 1 the result obtained is Train Accuracy 70.86%, Test Accuracy 69.38%, Precision 74.06%, Recall 69.38% and F1-Score 63.94%. After the Hyperopt technique has been done, the optimal values obtained are summarized in Table 6.

The result in Table 5 obtained from training the model is Train Accuracy 98.15%, Test Accuracy 93.1%, Precision 95.56%, Recall 93.59%, and F1-Score 94.56%. Without Optimisation, for Dataset 2 the result obtained is Train Accuracy 100%, Test Accuracy 93.85%, Precision 93.85%, Recall 93.85%, and F1-Score 93.85%. After the Hyperopt technique has been done, the optimal values obtained are summarized in Table 6 and the result obtained from training the model is Train Accuracy 100%, Test Accuracy 95.16%, Precision 95.17%, Recall 95.16%, and F1-Score 95.16.

Table 5. CNN results for Dataset 1 and Dataset 2

	Train Accuracy	Accuracy	Precision	Recall	F1-Score
DATASET 1					
No Optimization	70.86	69.38	74.06	69.38	63.94
Optimization	98.15	93.10	95.56	93.59	94.56
DATASET 2					
No Optimization	100	93.85	93.85	93.85	93.85
Optimization	100	95.16	95.17	95.16	95.16

Table 6. CNN Hyperparameter Values – Optimized for Dataset 1

Hyperparameters	Dropout	Activation	Batch Size	Filters	Kernel Size	Epoch
Optimal Values Dataset 1	0.44	TanH	60	70	5	5
Optimal Values Dataset 2	0.26	TanH	60	60	3	9

The performance of the model with HyperOpt optimization was much better than the model without optimization.

3.3 Discussion

3.3.1. LSTM

Figure 4 illustrates the results obtained for LSTM for Dataset 1 and Dataset 2. The result obtained from the optimization of the LSTM Algorithm shows that the optimized model produces better performance than the unoptimized model. For Dataset 1, differences of 13.87% were obtained for Training Accuracy, 12.8% for Test Accuracy, 10.72% for Precision, 6.68% for Recall, and 11.64% for F1-Score. For Dataset 2, differences of 13.78% were obtained for Training Accuracy, 13.47% for Test Accuracy, 14.14% for Precision, 17.3% for Recall, and 18.22% for F1-Score.

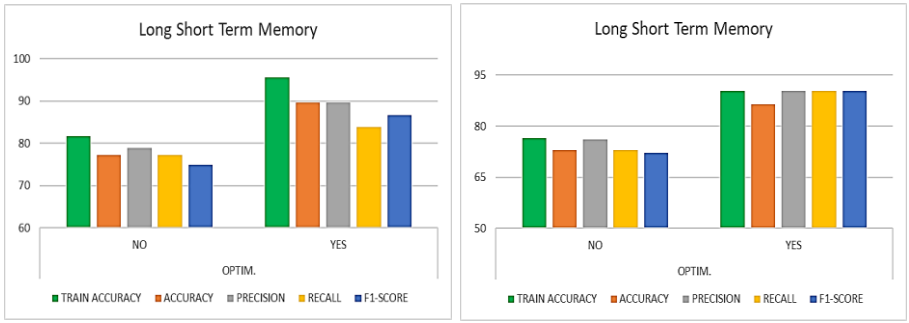

Figure 4. LSTM Performance result for Dataset 1 and Dataset 2

3.3.2. CNN

Figure 5 illustrates the results obtained for CNN for Dataset 1 and Dataset 2. The result obtained from the optimization of the CNN Algorithm shows that the optimized model performs better than the unoptimized model. For Dataset 1, differences of 27.29% were obtained for Training Accuracy, 23.72% for Test Accuracy, 21.50% for Precision, 24.21% for Recall, and 30.5% for F1-Score. For Dataset 2, differences of 0% were obtained for Training Accuracy, 1.31% for Test Accuracy, 1.32% for Precision, 1.31% for Recall, and 1.31% for F1-Score.

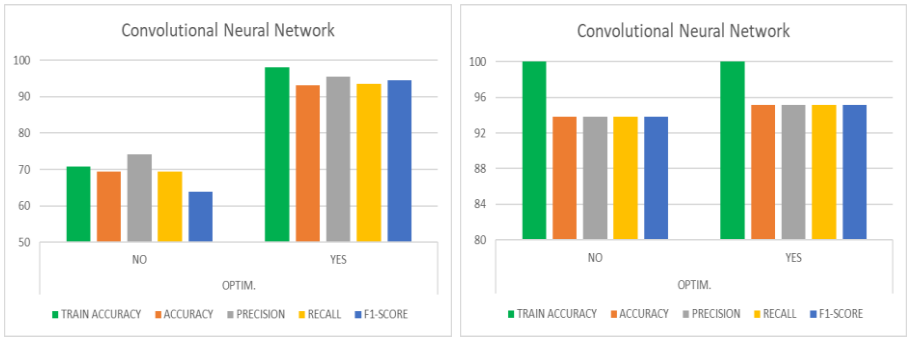

Figure 5. CNN Performance results for Dataset 1 anf Dataset 2

A comparison of the work of other authors and our work is presented in Table 7.

Table 7. Comparing our results with existing work.

Author	Dataset	Model	Performance
[18]	[11]	CNN	Accuracy = 92.77%, Precision = 92.20%, Recall = 92.27%, F1-Score = 92.1%
		LSTM	Accuracy = 87.04%, Precision = 91.46%, Recall = 87.04%, F1-Score = 89.2%

Author	Dataset	Model	Performance
[19]	Dataset from Kaggle	Hybrid CNN-LSTM	Accuracy =97.5%
[20]	ISOT dataset	Hybrid CNN-RNN	Accuracy =100% of training set
	FA-KES dataset		Accuracy = 60% of training set
Our Work	[11]	CNN	Accuracy = 93.1%, Precision = 95.56%, Recall = 93.59%, F1-Score = 94.56%
		LSTM	Accuracy = 89.69%, Precision = 89.61%, Recall = 83.84%, F1-Score = 86.63%
	[12]	CNN	Accuracy= 95.16%, Precision = 95.17%, Recall = 95.16%, F1-Score = 95.16%
		LSTM	Accuracy = 86.48%, Precision = 90.27, Recall = 90.31%, F1-Score = 90.28%

4. CONCLUSION

This study has focused on the development of optimized models for fake news detection through the use of a hyperparameter tuning technique. Machine learning models have shown promise as a way to solve the problem of the spread of false news in the digital era. The hyperparameters of these models, however, need to be carefully tuned in order for them to perform optimally and accurately. Through the systematic exploration and fine-tuning of hyperparameters using HyperOpt technique, the study has enhanced the effectiveness of fake news detection systems. By identifying the optimal configuration settings, the models better distinguished between fake and real news articles, improving their overall performance and reliability. The evaluation of the enhanced models using measures such as precision, recall, accuracy, and F1-Score revealed useful information about their performance. The findings show that hyperparameter tuning is crucial for enhancing the accuracy and capacity of false news detection systems. Further research can increase the number of hyperparameters being tuned so as to better improve the accuracy of the model. This research only used the statement or text for predictions. The proposed method can also be explored using authors, location, publisher etc. to predict whether a news is fake or real.

REFERENCES

[1] J. Zhang, B. Dong and P. S. Yu, "FAKEDETECTOR: Effective Fake News Detection with Deep Diffusive Neural Network," in 2020 IEEE 36th International Conference on Data Engineering(ICDE), Dallas, Texas, 2020.

[2] Q. Yumeng, "Predicting Future Rumours," Chinese Journal of Electronics Volume: 27, p. 514 – 520, 2018.

[3] S. Deepak and C. Bhadrachalam, "Deep neural approach to Fake-News identification," in International Conference on Computational Intelligence and Data Science (ICCIDS 2019),, 10.1016/j.procs.2020.03.276, 2019.

[4] P. Kaur, "Hybrid Text Classification Method for Fake News Detection.," International Journal of Engineering and Advanced Technology (IJEAT) , pp. 2388-2392, 2019.

[5] K. Poddar, D. G. Amali and K. S. Umadevi, "Comparison of Various Machine Learning Models for Accurate Detection of Fake News," Innovations in Power and Advanced Computing Technologies (i-PACT). doi:10.1109/i-pact44901.2019.8960044, 2019.

[6] M. A. Belhakimi, D. Ahlem and S. Giordano, "Merging deep learning model for fake news detection.," in International Conference on Advanced Electrical Engineering (ICAEE), 2019.

[7] M. Krešňáková, Viera, Sarnovsky, Martin and P. Butka, "Deep learning methods for Fake News detection.," 10.1109/CINTI-MACRo49179.2019.9105317., 2019.

[8] A. M. P. Braşoveanu and R. Andonie, "Integrating machine learning techniques in semantic fake news detection," Neural Processing Letters, vol. 52, no. 2., 2020.

[9] T. C. Truong, Q. B. Diep, I. Zelinka and R. Senkerik, "Supervised classification methods for fake news identification," in in Proceedings of the ICAISC, Zakopane, Poland, 2020.

[10] Z. Khanam, B. N. Alwasel, H. Sirafi and M. Rashid, "Fake News Detection Using Machine Learning Approaches," in IOP Conf. Ser.: Mater. Sci. Eng. 1099 012040, 2021.

[11] Y. Yang, 30 May 2017. [Online]. Available: https://drive.google.com/open?id=0B3e3qZpPtccsMFo5bk9Ib3VCc2c. [Accessed 26 February 2022].

[12] R. Patodi, 13 September 2019. [Online]. Available: https://drive.google.com/file/d/1er9NTLUA3qnRuyhfzuN0XUsolC4a-_1/view.

[13] A. Muzakir, K. Adi, R. Kusumaningrum, "Advancements in Semantic Expansion Techniques for Short Text Classification and Hate Speech Detectio," *Ingénierie des Systèmes d'Information*, Vol. 28, No. 3, June, pp. 545-556, 2023.

[14] M. Feurer and F. Hutter, "Hyperparameter Optimization. In: Hutter, F., Kotthoff, L., Vanschoren, J. (eds). Automated Machine Learning.," in The Springer Series on Challenges in Machine Learning., 2019.

[15] M. Claesen and B. Moor, "Hyperparameter Search in Machine Learning," arXiv:1502.02127, 2015.

[16] H. H. Aghdam and E. J. Heravi, "Guide to convolutional neural networks: A practical application to traffic-sign detection and classification," Cham, Switzerland: Springer, 2017.

[17] U. Ependi, A.F. Rochim, A. Wibowo, "A Hybrid Sampling Approach for Improving the Classification of Imbalanced Data Using ROS and NCL Methods," *International Journal of Intelligent Engineering and Systems*, vol. 16, no. 3, pp. 345-361, 2023.

[18] Y. Yang, L. Zheng, J. Zhang, Q. Cui, X. Zhang, Z. Li and P. S. Yu, "TI-CNN: Convolutional Neural Networks for Fake News Detection," Corr, 2018.

[19] A. Abdullah, M. Awan, M. Shehzad and M. Ashraf, "Fake news classification bimodal using convolutional neural network and long short-term memory," Int. J. Emerg. Technol. Learn, vol. 11, pp. 209-212, 2020.

[20] J. A. Nasir, O. S. Khan and I. Varlamis, "Fake news detection: A hybrid CNN-RNN based Deep Learning Approach," International Journal of Information Management Data Insights, vol. 1, no. 1, 2021.