Unmasking Disinformation: Detection of Fake News Online using Learning Techniques

***Abstract*—In an era where fake news proliferates effortlessly across social media platforms, the task of discerning reliable news sources has become increasingly challenging. Detecting fake news in the realm of social media presents unique characteristics and complexities that render traditional detection algorithms ineffective. Recent research endeavors have sought to address these challenges by leveraging a diverse array of methods, including machine learning, deep learning, feature engineering and graph mining. In our research, we investigate the detection of fake news by employing a combination of traditional machine learning algorithms and deep learning models for fake news classification. Furthermore, we extend our research by conducting real-time analysis for fake news detection using Twitter data and data from fact-checking sites. By combining the strengths of traditional machine learning and deep learning, our research offers a holistic approach to fake news detection, enhancing the accuracy and robustness of our results.**

***Index Terms*—Natural Language Processing, Text Classification, Fake News Detection**

I. Introduction

In today's era of the internet and social media, the phenomenon of fake news has emerged as an all-encompassing and powerful force, exerting a significant impact on society, politics, and public discourse. The rapid dissemination of fake and misleading information online has presented a crucial challenge that goes beyond the simple dissemination of facts. Fake news has the ability to create conflict, manipulate public opinion, and diminish trust in reliable sources of information. The issue of fake news is not a recent one, but the digital age has given it an unprecedented level of influence and reach. False or misleading narratives are shared through various online platforms, such as social media timelines, blog posts, and online newspapers, which makes it increasingly challenging to differentiate between trustworthy news sources and fabricated content [1]. The far-reaching consequences of this informational landscape have an impact not only on individuals' understanding of the world but also on their participation in civic processes. According to research findings, fake news is not limited to isolated incidents but rather permeates various domains, including politics, health, and even science [2]. False narratives surrounding political events have the potential to sway public opinion and influence elections [1], while medical misinformation can undermine public health efforts and lead to dangerous outcomes [3]. The detection and mitigation of fake news present unique challenges as the characteristics and methods used in creating deceptive content continue to evolve [4]. Traditional news media detection algorithms often struggle to combat the ingenuity of those spreading disinformation in the digital realm. Consequently, new and innovative approaches are necessary to address the multifaceted problem of online fake news.

This research work delves into the complexities of fake news detection, employing a variety of machine learning algorithms and deep learning models. By employing advanced detection techniques, we aim to enhance the accuracy and robustness of fake news detection. In this research, we aim to contribute to the ongoing effort to combat the spread of fake news in an increasingly digital world. By evaluating various learning models for the detection of fake news online and real time analysis, we can infer that the transformer based deep learning model, DistilBERT [5] is able to outperform traditional machine learning models.

II. Literature Survey

Kuai Xu, Feng Wang, and Haiyan Wang [6] carried out a research on the reputations of different domains and the characteristics of the content in both fake and genuine news. The research included a large dataset of popular real and fake news that were posted on Facebook, along with the comments, reactions, and shares. The authors examined the reputation of the domains by looking at factors such as registration behaviors, timing of registration, rankings of the domains, and their popularity. They also explored the characteristics of the content using techniques like term frequency-inverse document frequency (TF-IDF) analysis and Latent Dirichlet Allocation (LDA) topic modeling. The results of the research showed significant differences in the reputations and characteristics of the domains between real and fake news publishers. However, one limitation of this study was that it relied on a limited set of key words or terms that were identified through TF-IDF analysis. As a result, the research aimed to improve accuracy by investigating the Word2Vec algorithm, which works with complete vector representations of text. S. Sharma, M. Saraswat and A. K. Dubey [7] proposed a holistic solution for fake news detection. The user characteristics were classified using XGBoost algorithm and a sequential neural network and state-of-the-art (SoTA), BERT transformer is then used to classify the tweets. The authors compared results to baseline models and found their approach effective. S. H. Kong et. al. [8] proposed training four similar neural network models. These models are individually provided with distinct textual vectors representing the news title and news content, enabling a comparison of their respective performances. The training process involves the use of N-gram vectors for both news titles and content, as well as sequence vectors for news titles and content. Models trained with the N-gram and sequence

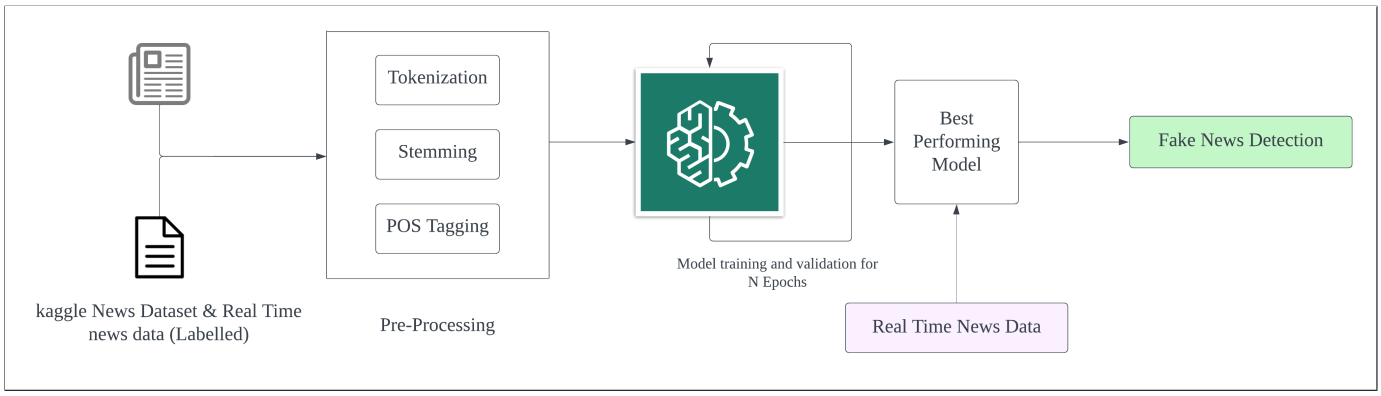

Fig. 1. Architecture Diagram for the detection of online fake news

vectors of news content were found to outperform those trained with news titles as well as baseline models. K. V. Vardhan et. al. [9] proposed a machine learning-based approach for fake news detection on social media. The authors use a variety of features, including textual, social, and temporal features, to train their models. They evaluated their method on a dataset of Twitter posts and achieved an accuracy of 95%. Petkar, P. B., and Sonawane, S. S. [10] put forth three primary methodologies for the detection of fake news: knowledge-based, style-based, and hybrid approaches. Knowledge-based approaches involve the comparison of fake news with a knowledge base consisting of verified facts, while style-based approaches focus on the analysis of the writing style employed in the news article. Hybrid approaches, on the other hand, combine the use of both methods. The authors also delve into the challenges associated with the detection of fake news and explore alternative approaches to tackle this issue. I. Ahmad, M. Yousaf, S. Yousaf, and M. O. Ahmad [11] proposed an approach for fake news detection that uses a combination of machine learning algorithms and linguistic features. The machine learning algorithms used in the study include Naive Bayes, Support Vector Machines (SVM), and Random Forest. While, the linguistic features used in the study include n-grams, sentiment scores, and readability scores. The authors evaluate their approach on four real-world datasets and show that it outperforms other methods. the paper evaluates various advanced techniques, including convolutional neural networks (CNNs), long short-term memories (LSTMs), ensemble methods, and attention mechanisms. S. Kumar, R. Asthana, S. Upadhyay, N. Upreti, M. Akbar in their research, combined CNN with bidirectional LSTM networks, while incorporating an attention mechanism that yielded an accuracy of 88.78%. In comparison, H. Ko et. al. [12] addressed the issue of identifying fake news and achieved a detection rate of 85%.

III. Proposed Methodology

Our model, as illustrated in figure 1, was employed for both training and testing the machine learning and deep learning models, as well as for subsequent real-time detection of fake news.

1) Data Collection
2) Pre-Processing
3) Model Training
4) Real-time fake news detection

A. Data Collection

This research work proposed has been trained and evaluated by utilizing the Kaggle dataset[1] for fake news. The primary goal of this dataset was to determine the classification of the quantity of instances that are fake and real.

For real-time prediction of fake news, we have used real-time tweets from twitter API v2[2] and Google news.

B. Pre-Processing

We perform various preliminary pre-processing activities on the training data. These activities encompass converting the strings to lowercase, removing leading/trailing white spaces, and eliminating duplicate spaces. Also, we perform the following pre-processing steps to further clean the data

1) Tokenization
2) Stemming
3) Lemmatization
4) POS Tagging

a) Tokenization: It is the process of dividing a string or text into a collection of tokens. A token can be considered as a component, such as a word in a sentence or a sentence in a paragraph.

b) Stemming: Stemming is the process of eliminating suffixes or prefixes from words in order to derive the central or base word. The objective of stemming is to categorize words with comparable meanings, thus streamlining the analysis of text.

c) Lemmatization: Lemmatization converts a word into their base or dictionary form, which is referred to as a *lemma*. Unlike stemming, which trims words to their root form using a set of rules, lemmatization takes into account a word's meaning and context in order to generate a more meaningful and linguistically accurate representation. Lemmatization is

[1]Kaggle Fake News Dataset (Labelled)
[2]Twitter API v2

employed to preserve grammatical accuracy and to cluster words with similar meanings together, thereby enabling a more precise understanding of natural language.

d) POS Tagging: POS tagging, also known as Part-of-Speech tagging, is a task that entails assigning grammatical categories or parts of speech to every word in a text or sentence. These categories generally consist of nouns, verbs, adjectives, adverbs, pronouns, prepositions, and conjunctions.

C. Model Training

To classify and detect fake news, we utilize and compare various models including Random Forest Classifier [13], Support Vector Machines (SVM) [14], and XGBoost [15]. A critical step in preparing the data for training these models is vectorization. Vectorization involves converting textual data into numerical representations, which can be processed by these algorithms. In our approach, we utilized the TF-IDF [16] vectorizer for this task.

a) Random Forest Classifier: Random Forest is an extension of decision tree algorithms, designed to mitigate issues such as over fitting and instability commonly associated with individual decision trees. Random Forest operates by constructing multiple decision trees during the training phase, each of which is trained on a different subset of the data and with a subset of features. This ensemble of trees, collectively referred to as a "forest," collaboratively contributes to making predictions. During the training phase, the Random Forest builds a "forest" of decision trees. The number of trees in the forest is a hyperparameter that can be adjusted based on the problem's requirements. The advantage of this is such that, every decision tree within the forest is constructed using a random subset of the training data (bootstrapped samples) and a random subset of the features. This introduces randomness and diversity into the model, which helps reduce overfitting. The decision trees are trained to classify data points based on the features and labels in the training set. When classifying new, unseen data points (inference phase), each tree in the forest independently predicts the class label or outcome for the input data. For classification tasks, the model employs a voting mechanism, where each tree votes for a class label. The class that receives the majority of votes becomes the predicted class for the input data point. This is known as majority voting. For regression tasks, the model averages the individual predictions from all trees to produce the final predicted value.

b) Support Vector Machines (SVM) Classifier: SVMs classify data by finding an optimal hyperplane that maximally separates data points of different classes in a feature space. SVM operates in a feature space where each data point is represented by a set of features or attributes. The number of dimensions in the feature space corresponds to the number of features used to describe the data. In the training phase, the SVM algorithm analyzes the labeled training data to find the optimal hyperplane. The hyperplane functions as the decision boundary that segregates the data points into their respective classes. The objective of the Support Vector Machine (SVM) is to identify the hyperplane that maximizes the margin, which represents the perpendicular distance between the hyperplane and the closest data points from each class. These closest data points are commonly referred to as support vectors. Moreover, the SVM accommodates non-linearly separable data by employing kernel functions such as polynomial or radial basis function kernels, to transform the feature space. This transformation allows for projecting the data into a higher-dimensional space where linear separation becomes feasible.

c) XGBoost: Gradient Boosting is a technique for ensemble learning that combines the predictions made by multiple base models, often decision trees, in order to construct a more robust predictive model. Its methodology involves training fresh models to rectify the inaccuracies of previous models. The fundamental concept behind this approach is to systematically refine the model's performance by adjusting its forecasts based on the gradients (derivatives) of the loss function, which quantifies the disparity between the predicted values and the actual values. XGBoost (Extreme Gradient Boosting) is an optimal implementation of Gradient Boosting. It optimizes the loss function for classification tasks, such as log loss (cross-entropy), and iteratively refines the model to minimize the loss while making predictions. By using a combination of multiple decision trees and optimizing their parameters, XGBoost achieves highly accurate classification.

In addition, we have attempted to incorporate SoTA deep learning models such as Recurrent Neural Networks (RNNs) [17], Long Short-Term Memory (LSTM) [18], and Bidirectional Encoder Representations from Transformers (BERT) [19], specifically DistilBERT [5] for fake news classification.

d) Recurrent Neural Networks (RNN): RNNs classify data by processing sequential input, such as time series or text, one step at a time, maintaining an internal state that encodes information from previous steps. RNNs use this context to produce an output sequence. For classification, an RNN typically employs a final layer, like a dense layer with softmax activation, to map the internal state or the final output of the sequence to class labels. The network is trained by minimizing a loss function, and backpropagation through time updates the model's parameters. This can be represented mathematically as,

$$h_t = \sigma(W \cdot x_t + U \cdot h_{t-1}) \tag{1}$$

where, h_t is the hidden state at time t, and x_t is the input data at time t. W and U are weight matrices. σ represents the activation function. To classify data, we use the final hidden state h_T to make predictions. A final transformation and an activation function is applied to produce the output y

$$y = \sigma(V \cdot h_T) \tag{2}$$

For the activation function, we make use of the sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

For optimization of the RNN, we use the Adaptive Moment Estimation (Adam) [20] optimiser. The loss function used is

the Binary Cross-Entropy, explained in equation 4, where y is the predicted output and y_{true} is the correct output.

$$L = -\sum(y_{true} * log(y) + (1 - y_{true}) * log(1 - y)) \quad (4)$$

For this classification objective, we have created an embedding layer that converts the input data into dense vectors of dimension 100, and a SimpleRNN layer with 100 units (neurons) that only returns the output at the last time step. We also create a dense layer with a single neuron and a sigmoid activation function to achieve our binary classification objective. We compile the model with the Adam optimizer and binary cross-entropy loss function for training. We have trained the RNN model for 20 epochs with a batch size of 32.

e) Long Short-Term Memory (LSTM): LSTM is a type of RNN designed to capture long-range dependencies in sequential data. Unlike traditional RNNs, LSTMs are equipped with specialized memory cells and gating mechanisms, making them effective at retaining and updating information over extended sequences. The input gate governs the influx of fresh information into the memory cell. The forget gate oversees the selection of relevant information from the preceding memory cell state. Lastly, the output gate, determines what information to pass on to the hidden state. The memory cell state is updated based on these gates and the candidate values. After processing the entire input sequence, the LSTM's final hidden state, denoted as h_t contains the information relevant to the classification task. To perform classification, we add a dense layer on top of the LSTM. The output of this layer is represented as y which is a vector of scores for each possible class. Let N be the number of classes and h_t be the final hidden state, we calculate the class probabilities for each class. To calculate the class probabilities, we apply the softmax activation given in equation 5.

$$P_{y_i} = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \quad (5)$$

where, z_i score for class i, the scores represent the model's confidence in each class for the given input. Subsequently, the class with the highest calculated probability is selected as the predicted class for the input sequence.

$$y_{predicted} = \arg\max_i P(y_i) \quad (6)$$

f) Bidirectional Encoder Representations from Transformers (BERT) and DistilBERT: BERT is a pre-trained model, trained on a massive amount of text data, and had the ability to understand the contextual relationships between words in sentences. BERT operates by tokenizing input text into subword units, embedding them into dense vectors, and processing them through a multi-layer bidirectional Transformer architecture. For the purpose of classification, we fine-tune DistilBERT, a light-weight version of BERT. DistilBERT employs a distillation process in which a pre-trained BERT model is used as a teacher to train the smaller DistilBERT. During training, DistilBERT learns not only to predict masked words in sentences but also to predict the teacher's predictions. This process encourages DistilBERT to capture the same knowledge and representations as BERT but with fewer parameters. DistilBERT employs knowledge compression which involves quantizing the model's weights, essentially representing them with fewer bits.

To use BERT based models for classification, we tokenize the input text into subword tokens using the same subword vocabulary that was used during pre-training, this representing each token as an embedding. The CLS embedding of is extracted from the BERT model, and fed through a classification layer, which consists of a weight matrix W and bias vector b. Class probabilities are then calculated by using a softmax function. This can be mathematically represented in equation 7.

$$P(y_i) = \frac{e^{(W_i \cdot [\text{CLS}]) + b_i}}{\sum_j e^{(W_j \cdot [\text{CLS}]) + b_j}} \quad (7)$$

where, W_i is the weight vector for class i, b_i is the bias for class i, $[CLS]$ represents the embedding of the [CLS] token.

We compile the model using the Adam optimizer, with the learning rate parameter is set to 5e-5. The number of epochs used during training is set to 5 with the batch size set to 32. The loss function used is the Sparse Categorical Cross Entropy (SCCE), represented by equation 8.

$$\text{SCCE} = -\log\left(\frac{e^{z_y}}{\sum_{j=1}^{C} e^{z_j}}\right) \quad (8)$$

where, e^{z_y} represents the exponential of the logit z corresponding to the true class label y.

IV. Results and Analysis

A. About the Dataset

The dataset[1] contains 20800 Rows and 5 columns. The columns present in the dataset are id, Article Title, Author Name, Article Text, and Label. The Label column is the target column Y and for the purpose of this research, we will use Article Text as the independent column X. We use the independent columns to predict the news as Reliable (0) and Unreliable (1).

B. Results

TABLE I
Traing and Testing accuracies of SVM, Random Forest and XGBoost Classifiers

Model	Training Accuracy	Testing Accuracy
Support Vector Machine	0.9996	0.9691
Random Forest Classifier	1.0000	0.9067
XG Boost Classifier	0.9997	0.9509

Table I illustrates the similarity between the accuracy achieved during the training and testing phases for the SVM, Random Forest, and XGBoost classifiers. On the other hand, Table II provides a concise representation of the training epochs for RNN, LSTM, and DistilBERT models. Additionally, the table presents a comparison of the average training

TABLE II
SUMMARY OF TRAINING EPOCHS, FOR RNN, LSTM, AND DISTILBERT

Model	Loss Function	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
RNN	Binary Cross-Entropy	0.3931	0.8084	0.4998	0.7465
LSTM	Binary Cross-Entropy	0.1832	0.9244	0.2718	0.9048
DiltilBERT	Sparse Categorical Cross Entropy	0.0238	0.9912	0.0210	0.9954

Number of Epochs for training was set to 5 for all of the models mentioned in the table above.

loss, training accuracy, validation loss, and validation accuracy across a span of **5** epochs. The observed similarities in the training accuracies and validation accuracies across these tables suggest that the aforementioned models do not demonstrate overfitting tendencies to the given dataset.

TABLE III
COMPARISON OF MODEL PERFORMANCE

Model	Accuracy	Precision	Recall	F-1 Score
SVM	0.9629	0.9672	0.9587	0.9639
Random Forest	0.8913	0.9632	0.8087	0.8785
XGBoost	0.9664	0.9671	0.9671	0.9671
RNN	0.8055	0.7607	0.7751	0.7678
LSTM	0.9767	0.9753	0.9758	0.9756
DiltilBERT	0.9923	1.0	0.9883	0.9941

Table III and figure 3 present a comparison of the model performance in the identification of fabricated information on the Kaggle fake news dataset. Based on the available data, it can be deduced that DistilBERT exhibits the best performance amongst the models.

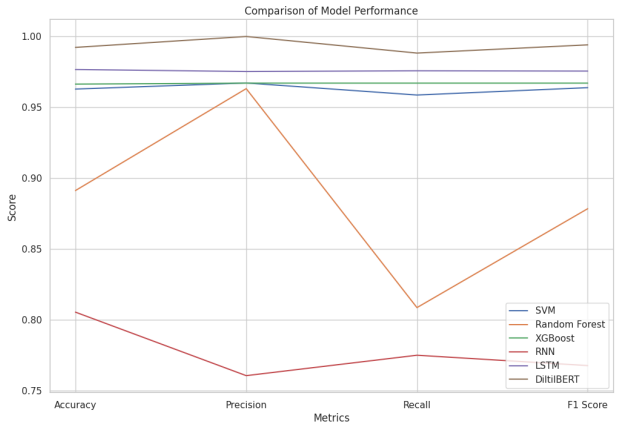

Fig. 2. Comparison of Model Performance

The ROC Curve for Fake News Detection is depicted in 4. The model's performance at all potential classification thresholds is quantified by the area under the ROC curve (AUC). This data suggests that the models excel at distinguishing between true negatives and false positives. The high AUC value indicates that the models can accurately classify the majority of the observations.

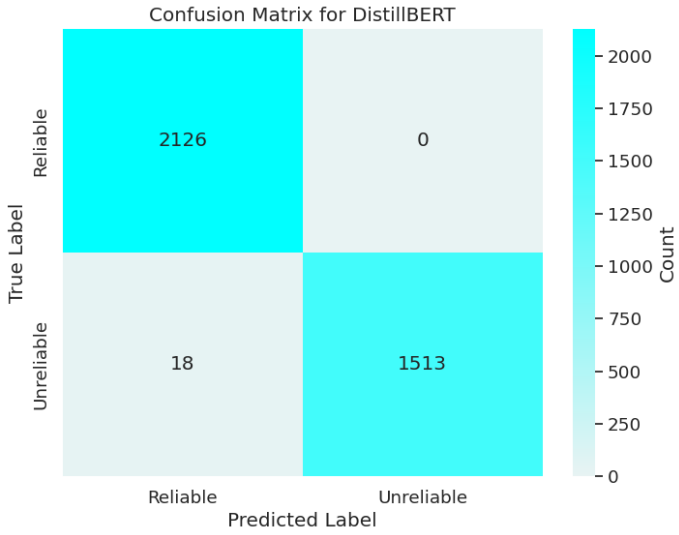

Fig. 3. Confusion Matrix for DistilBERT

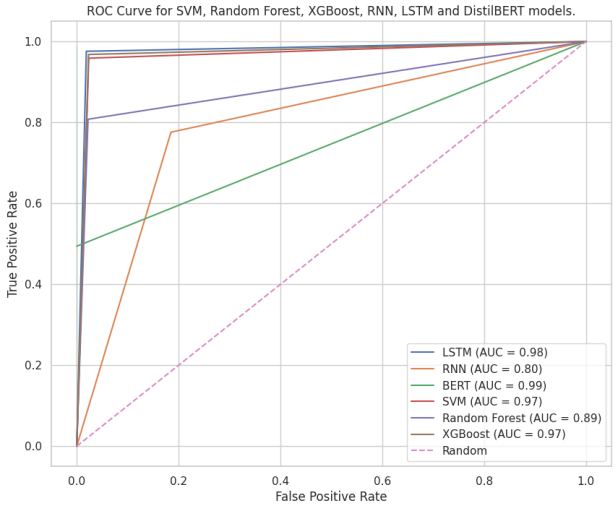

Fig. 4. ROC Curve

C. Results for Real-Time Detection

We use the best performing model, DistilBERT, for real-time detection of fake news utilizing Twitter data and information from fact-checking websites.

TABLE IV
EVALUATION OF DISTILBERT ON REAL-TIME DATA

Source	Reliable Articles	Fake Articles	Articles Predicted Correctly as Reliable	Articles Predicted Correctly as Fake	Accuracy
Twitter	25	12	23	9	0.8648
Google News	16	3	16	1	0.8947
Fact-Checkers	12	9	9	7	0.7619

a) About the Dataset: We gather tweets utilizing the Twitter API v2[3] and Google news. The tweets obtained are those that include community annotations identifying the tweet as false information. To identify false information, we compile data from Google News and manually categorize them as dependable or undependable. To ensure an impartial analysis, we assess the model by collecting news articles from well-known fact-checking websites [4][5].

b) Results: Table IV assesses the most effective model, namely DistilBERT, on three different sources, namely twitter, Google news, and Fact Checking Websites. DistilBERT achieved a reasonably high level of accuracy, specifically 86.48% on twitter data, 89.47% on Google news articles, and 76.19% on articles from fact-checking websites. Consequently, the model achieved an overall accuracy of 84.04% when predicting real-time data.

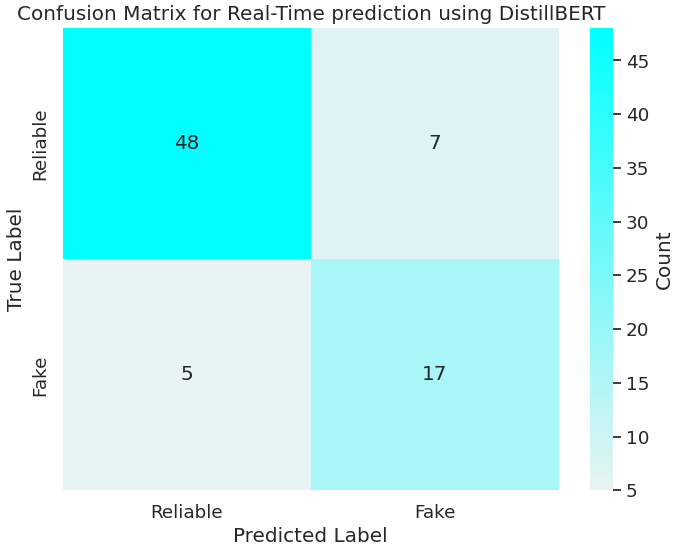

Fig. 5. Confusion Matrix for Real-Time Fake News prediction using DistillBERT

V. CONCLUSION

In conclusion, our extensive examination of machine learning and deep learning models utilized for the fake news dataset from Kaggle has uncovered noteworthy insights regarding the effectiveness of various algorithms. Among the array of models investigated, the BERT-based architecture, specifically DiltilBERT, emerged as the best performer for the identification of fake news. The model's performance, particularly following the process of fine-tuning on the dataset, exhibited an ability to achieve near-perfect accuracy, thereby showcasing its potential for accurately distinguishing between genuine and fabricated news.

To broaden the scope of our findings, we carried out real-time evaluations utilizing DistilBERT on present-day news and tweet data. The outcomes derived from this real-world assessment further reinforced the robustness of the model, providing substantial evidence of its expertise in detecting fake news across dynamic and evolving information sources.

These findings not only emphasize the significance of leveraging advanced language models for fake news detection but also demonstrates the practical value of these models in addressing the obstacles presented by the rapid spread of misinformation in real-time situations.

[3]Twitter API v2
[4]https://www.factchecker.in/
[5]https://www.factcheck.org/

REFERENCES

[1] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science*, vol. 359, no. 6380, pp. 1146–1151, 2018. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.aap9559

[2] G. Pennycook, A. Bear, E. Collins, and D. G. Rand, "The implied truth effect: Attaching warnings to a subset of fake news headlines increases perceived accuracy of headlines without warnings," *Management Science*, 2019, forthcoming. [Online]. Available: https://ssrn.com/abstract=3035384

[3] D. A. Broniatowski, A. M. Jamison, S. Qi, L. AlKulaib, T. Chen, A. Benton, S. C. Quinn, and M. Dredze, "Weaponized health communication: Twitter bots and russian trolls amplify the vaccine debate," *American Journal of Public Health*, vol. 108, pp. 1378–1384, 2018. [Online]. Available: https://ajph.aphapublications.org/doi/10.2105/AJPH.2018.304567

[4] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu, "Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media," *Big Data*, vol. 8, no. 3, pp. 171–188, Jun. 2020, funding Information: This material is in part supported by the National Science Foundation awards 1909555, 1614576, 1742702, 1820609, and 1915801. Publisher Copyright: © Copyright 2020, Mary Ann Liebert, Inc., publishers 2020.

[5] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, vol. abs/1910.01108, 2019. [Online]. Available: http://arxiv.org/abs/1910.01108

[6] H. Wang, F. Wang, and K. Xu, *Modeling Information Diffusion in Online Social Networks with Partial Differential Equations*, ser. Surveys and Tutorials in the Applied Mathematical Sciences. Springer-Verlag, 2015, vol. 7. [Online]. Available: https://link.springer.com/book/10.1007/978-3-030-38852-2

[7] S. Sharma, M. Saraswat, and A. K. Dubey, "Fake news detection using deep learning," in *Knowledge Graphs and Semantic Web*, B. Villazón-Terrazas, F. Ortiz-Rodríguez, S. Tiwari, A. Goyal, and M. Jabbar, Eds. Cham: Springer International Publishing, 2021, pp. 249–259. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-91305-2_19

[8] S. H. Kong, L. M. Tan, K. H. Gan, and N. H. Samsudin, "Fake news detection using deep learning," in *2020 IEEE 10th Symposium on Computer Applications Industrial Electronics (ISCAIE)*, 2020, pp. 102–107. [Online]. Available: https://ieeexplore.ieee.org/document/9108841

[9] K. V. Vardhan, B. Josephine, and K. R. Rao, "Fake news detection in social media using supervised learning techniques," in *2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, 2022, pp. 695–698. [Online]. Available: https://ieeexplore.ieee.org/document/9760961

[10] P. B. Petkar* and D. S. S. Sonawane, "Fake news detection: A survey of techniques," p. 383–386, Jul. 2020. [Online]. Available: http://dx.doi.org/10.35940/ijitee.I7098.079920

[11] I. Ahmad, M. Yousaf, S. Yousaf, M. O. Ahmad, and M. I. Uddin, "Fake news detection using machine learning ensemble methods," *Complexity*, vol. 2020, p. 8885861, 2020. [Online]. Available: https://doi.org/10.1155/2020/8885861

[12] H. Ko, J. Y. Hong, S. Kim, L. Mesicek, and I. S. Na, "Human-machine interaction: A case study on fake news detection using a backtracking based on a cognitive system," *Cognitive Systems Research*, vol. 55, pp. 77–81, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389041718310647

[13] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 10 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[14] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 09 1995. [Online]. Available: https://doi.org/10.1007/BF00994018

[15] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[16] J. E. Ramos, "Using tf-idf to determine word relevance in document queries," 2003. [Online]. Available: https://api.semanticscholar.org/CorpusID:14638345

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. [Online]. Available: https://www.nature.com/articles/323533a0

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6795963

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[20] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec. 2014.