# A Hybrid Rumor Detection Model Derived from a Comparative Study of Supervised Approaches

by

Mehzabin Sadat Aothoi
19101353
Samin Ahsan
19101497
Fardeen Ahmed
22241037

A Thesis submitted to the
Department of Computer Science & Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science & Engineering

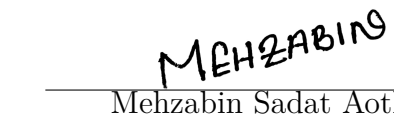
Department of Computer Science and Engineering
School of Data and Sciences
Brac University
January 2023

# Declaration

It is hereby declared that

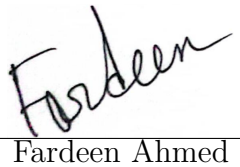
1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

**Student's Full Name & Signature:**



Mehzabin Sadat Aothoi
19101353

Samin Ahsan
19101497



Fardeen Ahmed
22241037

# Approval

The thesis titled "A Hybrid Rumor Detection Model Derived from a Comparative Study of Supervised Approaches" submitted by

1. Mehzabin Sadat Aothoi(19101353)
2. Samin Ahsan(19101497)
3. Fardeen Ahmed(22241037)

Of Fall, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science & Engineering on January 19, 2023.

**Examining Committee:**

Supervisor:
(Member)

Annajiat Alim Rasel
Digitally signed by Annajiat Alim Rasel
DN: cn=Annajiat Alim Rasel, o=Brac University, ou=CSE Department, email=annajiat@bracu.ac.bd, c=BD
Date: 2023.01.14 23:04:00 +06'00'

Mr. Annajiat Alim Rasel
Senior Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Ms. Najeefa Nikhat Choudhury
Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

# Abstract

In the current age of social media, information spreads like wildfire. Unfortunately, this also means that misinformation or rumors can spread easily. The spread of this misinformation can have negative consequences for society. This is especially true in recent years due to growing engagement in social media platforms for news. Hence, to prevent the spread of rumors, rumor detection is necessary. Bangladesh has been no exception to the spread of misinformation, causing countless propaganda over the years. Although a significant amount of work has already been conducted regarding rumor detection in English, Bangla rumor detection is still in its infancy. For our research, we first compared several Machine Learning (ML) models and Deep Learning (DL) models for rumor detection using both Bangla and English datasets. Comparing and analyzing the results, we implemented an Ensemble ML model and finally our hybrid model, which is a combination of our best-performing ML model and DL model that outperformed all other baseline state-of-the-art models.



**Keywords:** Rumor Detection; NLP; Machine Learning; Deep Learning; Decision tree; Random Forest; Naive Bayes; Support Vector Machine; BERT; RNN; CNN

# Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis has been completed without any major interruption.

Secondly, utmost appreciation for our co-advisor Ms. Najeefa Nikhat Choudhury ma'am, and advisor Mr. Annajiat Alim Rasel sir for their kind support and advice in our work.

And finally, unending gratefulness for our parents without whose love and care throughout it may not have been possible. With their endless aid and prayer, we are now on the verge of our graduation.

# Table of Contents

# List of Figures

# List of Tables

# Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$BERT$ Bi-directional Encoder Representations from Transformers

$BiGCN$ Bi-Directional Graph Convolutional Network

$DL$ Deep Learning

$DT$ Decision Tree

$GCN$ Graph Convolutional Network

$GDBT$ Gradient Boost

$GNN$ Graph Neural Network

$LR$ Linear Regression

$LSTM$ Long short-term memory

$ML$ Machine Learning

$MNB$ Multinomial Naive Bayes

$NB$ Naive Bayes

$RBF$ Radial Basis Function

$RF$ Random Forest

$RNN$ Recurrent Neural Network

$RvNN$ Recursive Neural Network

$SVM$ Support Vector Machine

$TF-IDF$ Term Frequency–Inverse Document Frequency

# Chapter 1

# Introduction

The spread of rumors has become the subject of countless studies. Rumors have become synonymous with fake news, although it covers a much more comprehensive range of content. Many have tried to define fake news, but most definitions become blurry when classifying news as fake in reality. One of the definitions that adequately cover the range of the types of deceptive news is given by the European Commission [7]: fake news is "All forms of false, inaccurate, or misleading information designed, presented and promoted to intentionally cause public harm or for profit." Rumors fall under the umbrella of fake news since they are not verified to be true or false. Authors from [10] adopt the definition that rumors are, at the time of posting, circulating information whose veracity is still unverified in their paper which is in line with most major dictionaries.

The spread of rumors has become blazing fast due to social platform usage being on the rise. The aftermath of such spreads is typically rather negative and brings more harm than good to society, such as civil and political unrest. Human judgment can become heavily skewed by rumors, as has been proven during major events like the U.S. Presidential Election in 2016 and the recent COVID-19 pandemic. Although social media companies employ many different methods to eradicate the spread of rumors, a lot of the time it falls short of the desired success rate. Therefore, improving the rumor detection process has become quite a necessity over the years and has piqued the interest of different scientific communities, ranging from social psychology to computer science.

The task of rumor detection becomes much more tedious when it has to be defined for a computer to understand. Early research, as highlighted in [14], looked at the factors of rumor diffusion and how believability played a role in its circulation. Before the dawn of the internet, the spread of rumors was analog, passed around through in-person conversations and newspapers at best. However, the internet changed the landscape of information and communication. This was further amplified by the introduction of social media platforms and their accumulation of large user bases. Consequently, the task of fact-checking each post became an unanticipated and daunting challenge. With the freedom of posting anything for free at any time, several morally compromised or sometimes uninformed users began spreading rumors with varying intentions, ranging from malicious to benevolent.

To detect and eradicate rumors, computer scientists employ several techniques, which include ML approaches as well as DL approaches. Regardless of the approach, research has indicated that there are four major components of rumor classification. The first is rumor detection, where a binary classifier determines if the information being a rumor is true or false. The second component is rumor tracking, which is tasked with collecting and filtering posts that discuss the rumor, now known as "priori". Alongside the rumor tracking component, the stance classification component tries to establish how each of the posts is contributing to the rumor's veracity. Finally, the veracity classification component tries to ascertain the trustworthiness of a rumor.

## 1.1 Research Problem

Despite the numerous research programs dedicated to rumor detection, building a model that can accurately predict if a piece of information is a rumor or not still brings with it several challenges. The most seemingly difficult task is to find or even create an appropriate dataset. Then comes the challenge of choosing between different approaches, such as ML-based or DL-based approaches. Each approach has its own sets of strengths and weaknesses, and the question of which is the best fit depends on the problem statement itself. Finally, multilingualism poses one of the biggest challenges, as languages other than English scarcely have labeled datasets publicly available. The models as well are fine-tuned for the English language in most cases, which makes it hard to detect rumors on platforms where English is not the only dominant medium of communication.

To evaluate the different rumor detection models researchers come up with, uniform datasets are crucial for training and testing. However, in reality, it is extremely difficult to find datasets that can be used to train all models to make comparisons on the same scale. One of the primary reasons for this is the unavailability of labeled datasets for public use. Furthermore, labeled datasets need to fulfill certain criteria to be used for training models. Authors of [19] conducted a survey for their proposed FNDD characterization, consisting of eleven characteristics for existing and future datasets. They examined 61 datasets, of which only 27 were explicitly made available for research purposes. The paper goes on to define the previously stated characteristics of a dataset, and the existing datasets only partially fit those criteria. They also highlighted challenges regarding multimedia, multilingual, cross-domain, and COVID-19 datasets that researchers face regularly. The lack of big, comprehensive datasets is also a major point of their paper.

Rumor detection, in general, is a relatively new field of research as pointed out by [7] who further say that very little work has been done in automatic rumor detection. The existing approaches mostly focus on stance and veracity classification, leaving rumor tracking behind. In recent years, ML approaches have gained notable attention in detecting rumors and obtaining cutting-edge outcomes. ML-based techniques like Support Vector Machine and Decision Tree have all managed to achieve noticeable improvements in rumor detection, but none are suitable for all use cases. The choice of approach largely depends on the datasets available and the target

itself. Furthermore, ML models depend on data pre-processing to a large extent and also rely on manual feature extraction. What's more, they cannot extract high-dimensional features.

DL is an even newer approach to rumor detection than ML but has proven to be a better choice in many cases as pointed out by [22]. They discovered that DL has certain advantages over ML which leads to better scores and, consequently, an improved ability to detect rumors. Some of these are feature extraction automation, reduced dependence on data pre-processing, high-dimensional feature extraction capabilities, and overall better accuracy. However, the downsides include the influence of the feature and selection classifier on the efficiency of the model, lack of feature engineering, scarcity of propagation-based studies, and overall lack of data. Despite outperforming ML models and being a seemingly better choice, DL models require quite a substantial amount of resources that are not always readily available. The challenge becomes bigger as many papers do not fully make their implementation details publicly available for further testing and the sizes of the datasets are not sufficient for improving a DL model performance.

Adding on to the previously mentioned analysis by [19], it is evident that multilingual datasets are a rare breed. Most datasets are developed in English only, which limits fact-checking efficacy in different languages like Bangla. Similar to other countries, Bangladesh heavily suffers from the impacts of rumor propagation. However, tackling the task is no easy task, as only a handful of comprehensive and labeled datasets are available in the Bangla language. This becomes clear from the research of [15] who at the time found that no other computational approaches were available for fake news or rumor detection. They, later on, went on to claim that their creation of the "BanFakeNews" dataset is the first publicly available Bangla dataset for rumor detection. Consequently, with this lack of diversity in datasets, it becomes hard to determine which models are appropriate for general rumor detection. This is proven by the research from [16] who reported that SVM performed better than Multinomial Naive Bayes (MNB) in Bangla rumor detection and is generally a good choice. On the other hand, [11] reported that SVM was particularly worse than other algorithms like DT and K-Nearest Neighbor. In their study, MNB provided the best results instead, unlike the previous paper. Contradictory results like this truly make it difficult to choose an appropriate model for a rumor detection problem.

## 1.2 Research Objectives

This paper aims to make the decision-making process of settling on an approach for a rumor detection algorithm easier to some degree. Our research can be divided into two sections, the comparative study and the development of a hybrid model where the output of the first section establishes the parameters for the second one.

The vision of this paper is to first collect and pre-process different publicly available datasets in order to implement the ML and DL-based models using libraries like Scikit-learn [1] and Keras [2]. ML-based models include Support Vector Machine (SVM), Decision Tree (DT), Naive Bayes (NB), and Random Forest (RF).

For DL-based models, we have chosen, Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), and Bi-directional Encoder Representations from Transformers (BERT). Having trained the models, we would test them and pit them against each other. From the comparison, we would analyze the output and implement an Ensemble ML Model as well as a Hybrid Model that will be able to outplay all the other models. Therefore, the objective of this paper can be summarized in the following key points:

- Collect and pre-process datasets.
- Compare each Machine Learning model.
- Compare each Deep Learning model.
- Show comparisons between ML and DL approaches.
- Analyze the result.
- Implement Ensemble ML Model.
- Implement a Hybrid Model based on analysis.

# Chapter 2

# Literature Review

In the past, a substantial amount of research has taken place on rumor detection using machine learning, but rumor detection using deep learning is relatively new and has proven to be more effective than classic ML methods. In this paper, we aim to determine which ML and DL model outperforms the others, and compare the average results of ML models with DL ones to find out if DL actually has an advantage over ML.

## 2.1 Model Description

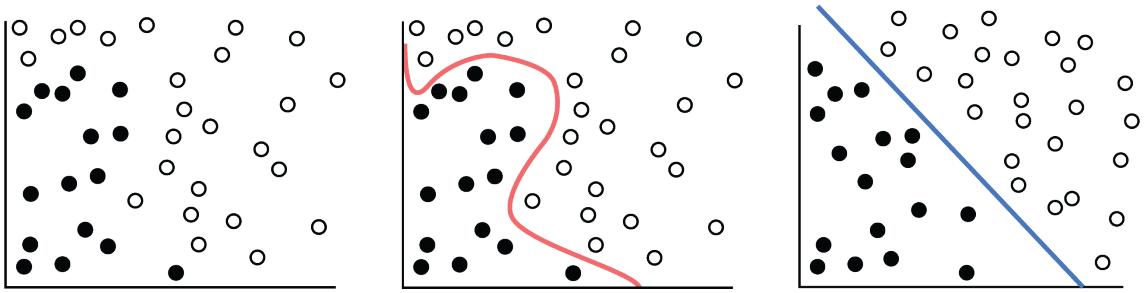
This section holds a brief description of all the models that we worked with in order to accomplish our research.

Firstly, we are going to briefly describe the working mechanism of our ML models which are, Support Vector Machine, Decision Tree, Naive Bayes, and Random Forest.

#### Support Vector Machine (SVM)

We know that SVM is a supervised learning model that categorizes data points by mapping them into "high dimensional feature space". After it finds a separator between the categories, the data is transformed such that a separator in the form of a hyperplane can be drawn. Following that, new data features can be utilized to predict which category a new record should fall into. The function used to accomplish the transformation is known as the "kernel function". There are different types of kernel functions. According to SciKit Learn [1], the major ones are Linear, Polynomial, Radial Basis Function, and Sigmoid. By default, SVM uses RBF kernel, and although some researchers [8] claim that SVM performs faster and better if we use the Linear kernel function for text-categorization, yet chose to apply the RBF kernel as our goal is to test the performance of the models in their very default form. While the kernel function is set to RBF, two parameters must be taken into consideration, C and gamma. Where the gamma parameter, with low values signifying "far" and high values signifying "near," describes roughly how much the impact of a single training sample extends, and the C parameter compromises between correctly classifying training samples and maximizing the margin of the decision function. A

narrower margin will be acceptable for greater values of C if the decision function is more accurate at categorizing training points.
The following figures can be used to visualize:



(a) Original (b) Separated (c) Transformed

Figure 2.1: SVM working mechanism

### Decision Tree (DT)

From the article [9] we can understand that Decision Trees are also a kind of supervised machine learning where data is constantly separated based on a certain parameter. Decision nodes and leaves are the two entities through which the DT can be explained. The decisions are represented by the leaves and the data is separated at the decision nodes. Although there are several techniques for selecting the best feature for each node, the Gini impurity (The frequency of misclassification of a randomly selected attribute), and Information gain (The difference in entropy between before and after a split on a particular property) are the two most often utilized splitting criteria approaches. The three most popular decision three algorithms are ID3, C4.5, and CART. For the default decision tree classifier, Scikit-learn employs an enhanced version of the CART algorithm, and it generally makes use of Gini impurity to determine the best attribute to split on.

$$\text{Gini Impurity } = 1 - \sum_{i} (p_i)^2 \tag{2.1}$$

The equation (2.1), $p_i$ refers to the likelihood of samples belonging to class $i$ at a specific node. Gini impurity is bottom capped by 0, therefore, the resulting 0 denotes that the data set is pure and comprises only one class. When assessing with Gini impurity, a lower value is preferable.

### Random Forest (RF)

Another supervised machine learning algorithm is Random Forest. It is trained with the "Bagging" approach [17]. The combination of multiple learning models to improve the resulting output drives this concept. In accordance, a random forest creates a forest by merging numerous decision trees to derive more precise and consistent predictions.

Now, *Bagging* is the implementation of the bootstrap approach to an ML system with significant variance. Here, Variance is an inaccuracy caused by sensitivity to minor differences in the training dataset. and Bootstrap is a statistical procedure

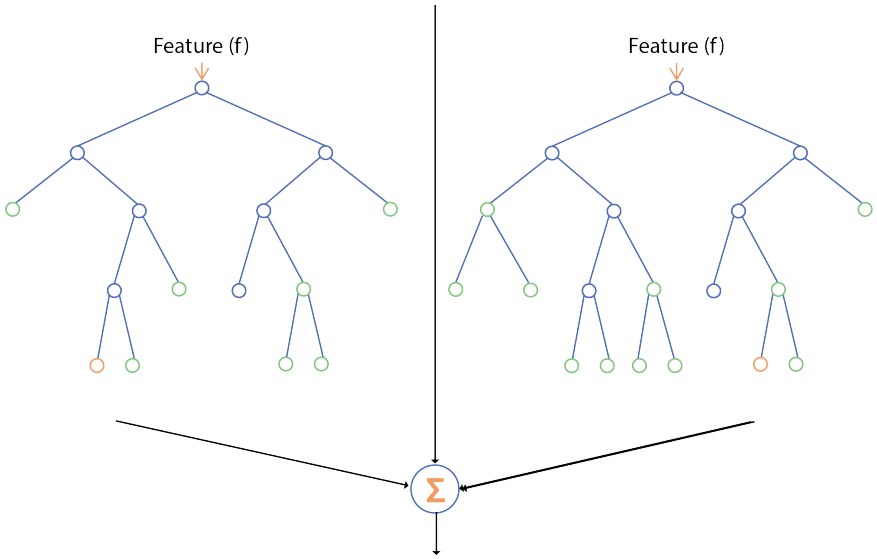



Figure 2.2: RF working mechanism

for resampling data. It entails resampling a dataset using substitution repeatedly.

### Naive Bayes (NB)

Naive Bayes is a type of classifier that employs the Bayes Theorem [32]. It forecasts probabilities for each class, such as the probability that a given data element belongs to a specific category. The most likely category is the one with the greatest probability. This is also referred to as Maximum A Posteriori (MAP).
We know from Bayes' theorem,

$$P(H|E) = \frac{(P(E|H) * P(H))}{P(E)} \tag{2.2}$$

Conditional probability is the cornerstone of Bayes's theorem. Some terms in the theorem are:
**Class prior:** Probability of occurring event $H$ without any prior knowledge of event $E$. Which is $P(H)$.
**Predictor prior:** Probability of occurring event $E$ without any prior knowledge of event $H$. Which is $P(E)$.
**Posterior probability:** Probability of occurring event $E$ with the knowledge about event $H$. Which is $P(E|H)$.
**Likelihood:** Probability of occurring event $H$ with the knowledge about event $E$. Which is $P(H|E)$.

However, based upon the Naive Bayes' theorem,

$$P(H|E) = P(E|H) * P(H) \tag{2.3}$$

Unlike Bayes', Naive Bayes' classifier presupposes that all the features are independent of each other. So, one of the features being present or absent does not anyhow influence the others. This is also one of many reasons why the algorithm is quicker.

Despite being "naive," it outperforms several complex models.

There are three Naive Bayes' Classifiers, Gaussian, Multinomial, and Bernoulli. For our study, we applied the Gaussian one. Because when the variables are continuous, this Naive Bayes' classifier is utilized and it also assumes that all variables follow a normal distribution.

Now for the DL models, we chose to work with CNN, RNN, and BERT. Similarly, the working principle for the mentioned models are presented below:

### Convolutional Neural Networks (CNN)

A CNN works with a volume of inputs and in the simplest of terms it produces a third relationship from the mathematical combination of two. To extract features, a filter or kernel is used, the parameters of which depend on the amount of input and overlapping allowed in a batch. The applied filter generates multiple feature maps using a non-linear connection for the outcome is generated by an activation function. As studies like [4] show, the ReLU seems to be the best-performing activation function. The study also shows that the input word vector representation plays an important part in performance and suggests using open-source embedding libraries such as GloVe or word2vec. To prevent the feature maps from shrinking, padding is used to hold onto information. A pooling layer, such as Max pooling, is used to reduce dimensional complexity and prevent overfitting. A flattened layer can be used as well for changing dimensions. The very last layers of CNN are generally fully connected meaning each node of a layer is connected to the nodes of another layer.

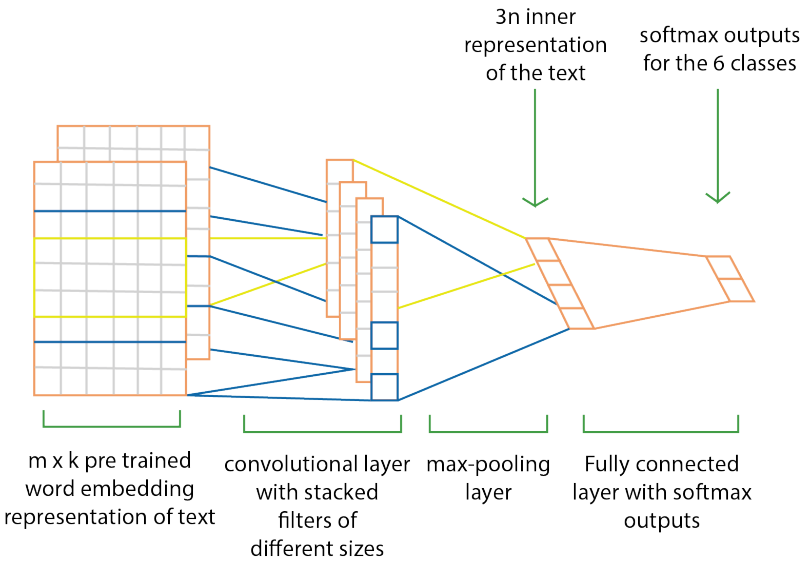



Figure 2.3: CNN working mechanism

### Recurrent Neural Networks (RNN)

RNN belongs to the family of neural networks that usually processes sequential and time-series data. As the authors from [21] show, the input to the current state of the

RNN are gathered from the previous state. This class of neural networks remembers the past and recalls them along with options based on it. The output from the prior state is used to update the hidden state. The architecture of an RNN model depends on the problem at hand. Some variations of it are One-to-one, One-to-many, Many-to-one, and Many-to-many. Much like CNN, some popular activation functions for the hidden layers in RNN include ReLU, Sigmoid, and Tanh. Using the memory it uses to store information from previous inputs, called long short-term memory, which is optimal at predicting time series or sequential data.

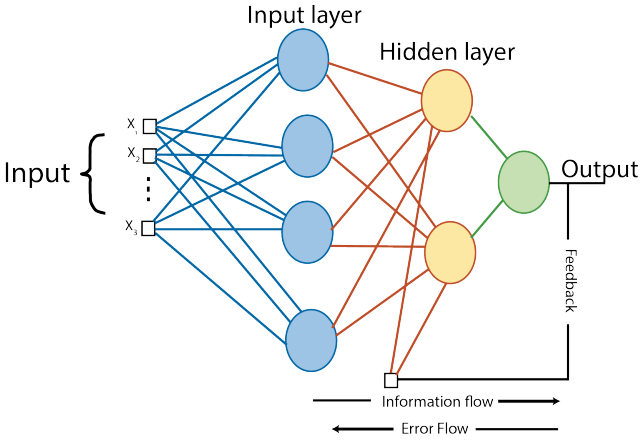



Figure 2.4: RNN architecture

**Bidirectional Encoder Representations from Transformers (BERT):**

BERT is a bi-directional transformer that is made up of an encoder for reading text input and a decoder for predicting task output. As the authors of [18] point out, BERT is able to capture information from unlabeled text by concatenating a token's context representation of both sides from all layers. It generates contextualized embedding using its attention mechanism. Because BERT is bi-directional, it understands word relationships that way and generates vectors to represent that relationship for each word in a sentence. For representing an input, BERT considers the sum of the Token embeddings which is added at the start [CLS] and end [SEP] of a sentence, Segment embeddings which are used to distinguish different sentences, and Position embeddings which denotes the position of a token in a sentence.

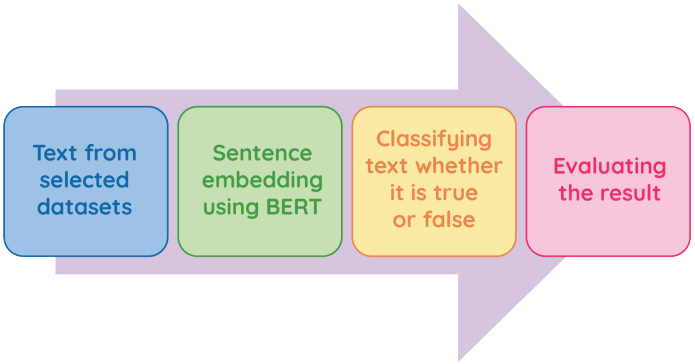



Figure 2.5: BERT working mechanism

### Ensembled Machine Learning Model

For our ensembled ML model, we are going to utilize all of our four models. After being done comparing all the ML models, we are going to take the models positioned as $2^{nd}$, $3^{rd}$, and $4^{th}$, stack them, and feed the training data. Lastly, we are going to use our best-performing ML model as the final classifier in order to extract the output.

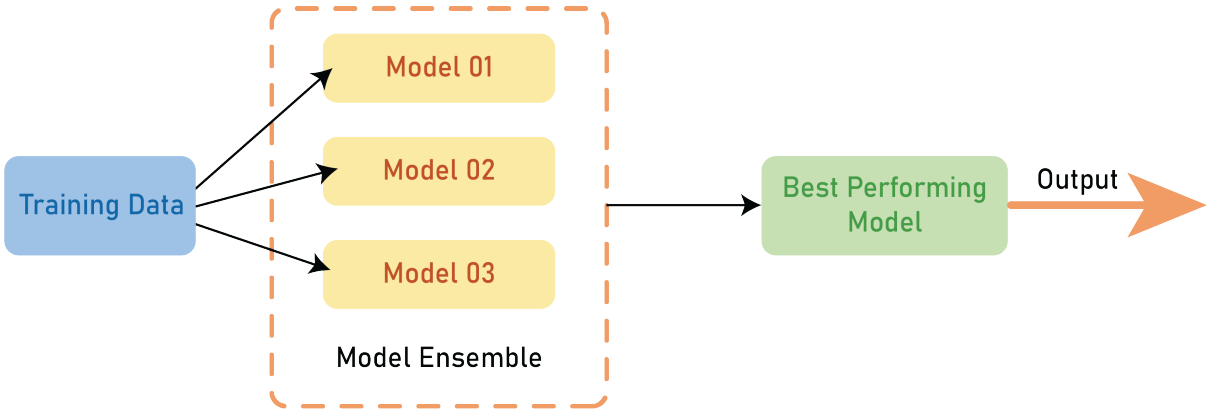



Figure 2.6: Ensembled ML model structure

### The Hybrid Model

The hybrid model proposed in this paper is derived from combining the best-performing baseline ML model and DL model. Here, the DL model is used as the weak learner which makes predictions based on the training data. The predictions made by it are fed into the meta-learner which is the ML model to make the final classifications. In this way, the best performance is achieved using both DL and ML approaches which not only outperform the baseline models but are much easier to train.

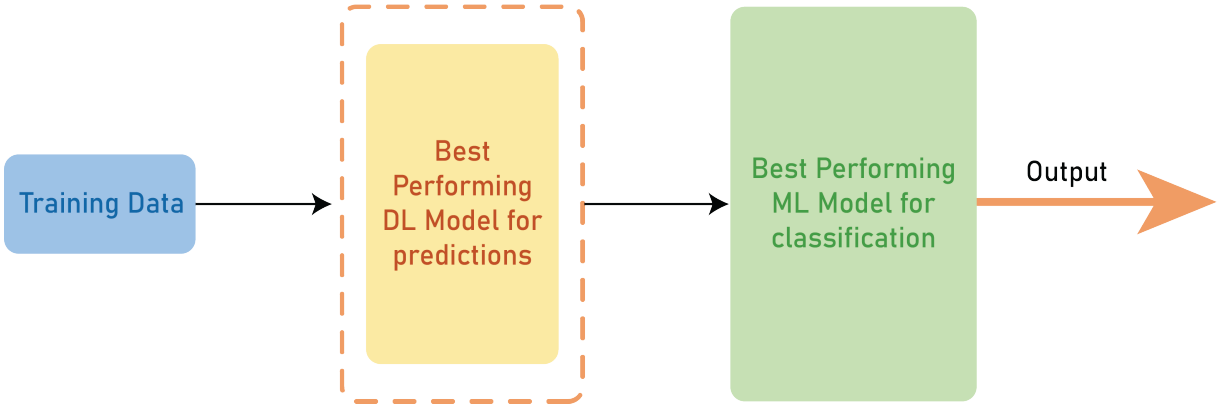



Figure 2.7: Hybrid model structure

For this research, we opted to keep the state of the art models in their default form. Therefore, even if the addition of certain parameters can make a particular model perform better, we adhered to the initial form.

## 2.2 Related Works

Several attempts have been made to verify the accuracy of differentiating rumors and reality. attempted to assess prior studies in which different models based on Machine Learning and Deep Learning were compared and introduced, in the field of rumor detection.

From the findings of [11], we see that the authors came to a conclusion that SVM (75.5%) models fail to handle noisy data, KNN (79.2%) proves to be a lazy learner, and even though decision tree (82.7%) yield good results, it is concluded to be very unstable. Therefore the authors initially decided to go with Multinomial Naive Bayes (90.4%), and Gradient Boosting (88.3%). For a better and more satisfactory result, they also included Random Forest (86.5%). The dataset that was used in this paper is a dataset called "LIAR", which is a very large dataset and available to the public.

But unlike [11], according to [16], where the authors attempted to detect fake Bangla news using the previously mentioned MNB and SVM, between the two supervised algorithms SVM is claimed to be better performant compared to MNB. The authors scraped data from Bangla news articles published on ProthomAlo.

In the paper [12], several datasets suitable for rumor detection are mentioned with their properties such as text, user info, timestamp, and propagation info. Along with datasets, which machine learning approaches were used in a good number of papers and which types of information were used in them are presented via a table in a very clean manner. Aside from mentioning various ML approaches and datasets that are available, the authors also discussed future opportunities. The BanFake-News dataset is a very impactful resource in the field of Bangla Rumor Detection as there are not a good number of Bangla datasets available. The dataset by [15] consists of almost 50k annotated data. To accomplish this, the authors claimed to use linguistic feature-based approaches for which SVM, RF, LR, and Neural Network Models for which CNN, BiLSTM, and BERT models were used. In this paper, SVM again proved to perform better compared to RF, LR as well as neural network models. From [19], we get an insight into what characteristics makes a dataset fit for training our ML or DL models. The authors extracted 61 datasets from 164 papers and enlisted 10 dataset requirements based on which we should evaluate a dataset. The mentioned characteristics were further categorized into 4 categories. In the paper, the authors also mentioned some challenges that researchers face while developing new datasets for rumor detection.

Both [20] and [13] discuss rumor detection methods and tools. The latter compares the different strategies and their objectives. The authors also mentioned in the paper how the improvement of rumor detection models can be made by incorporating linguistics, some pre-defined RD rules, and ML approaches. Similarly in [20] authors also described numerous fake news detection methods (content, knowledge, style, linguistic, visual, social, network, temporal, credibility-based) and datasets. The techniques mentioned in this paper are, MVAE (finds correlations across different modalities), SAFE (jointly learns the textual and visual features), and FANG

(captures the social interactions between users, articles, and media).

3 Decision Tree algorithms were studied by the authors of [9] and they were, ID3, C4.5, and CART, and their study concluded that ID3 cannot take continuous datasets for simulation however C4.5 and CART can do so. The authors used a car dataset to evaluate the 3 DT algorithms and concluded that CART is the slowest but the most accurate.

The authors create a dataset of actual and fraudulent news about Covid-19, comprising of 10,700 social media posts, in the paper [23]. They use different ML models: LR, DT, SVM with linear kernel, and GDBT. SVM performed the best while DT performed the worst. GDBT performed slightly better than DT and LR performed slightly worse than SVM. A method was proposed by [24] for improving fake news detection by automatically gathering evidence for each claim. The fake news classification is performed using several ML (LR and SVM) and DL (LSTM with pre-trained $BERT_{base}$ $RoBERT_{base}$ and $XLNET_{base}$ classifiers) models. The results of their experiments show that ML models perform worse than the DL models.

Furthermore, [26] discusses a system developed as part of the CONSTRAINT-2021 shared task. They compared multiple ML and DL models: SVM, CNN, BiLSTM, and CNN+BiLSTM and they used TF-IDF and word2vec embedding techniques. The authors found that for fake news classification, SVM with TF-IDF features achieves the highest f1 score (94.39%). A combination of CNN and BiLSTM achieved an f1 score of 92.01%. The SVM models with Word2vec obtained slightly better results than CNN+BiLSTM (92.66% and 92.94% with 200 and 150-word embeddings respectively) with the Covid-19 dataset pre-released as part of the CONSTRAINT-2021 shared task.

Moreover, [27] proposes a hybrid method (ML: SVM, RF, LR, KNN + DL: ANNs) for fake news detection. Doc2vec and TF-IDF were used for representations. Different versions of the Hybrid Model were used such as Hybrid V1, V2, V3, and V4.

However, the authors of paper [29] discovered that for rumor detection, RNN architecture outperforms pure CNN, RNN-CNN, and CNN-RNN architectures. This paper discusses a rumor detection system for detecting both rumors and non-rumors using multiple models: SVM, XGBoost Classier, RF, Extra Tree Classifier (ET), and DT. They use these different machine learning models to build a hybrid ensemble model and applied two deep learning models: LSTM and BERT. The results show that the best result is achieved by the hybrid model for the COVID-19 and Twitter 15 and Twitter 16 datasets. Validation loss falls and validation accuracy rises in both the LSTM and BERT models, indicating that the model is learning effectively.

Additionally, [31] explores self-supervised learning (SSL) on heterogeneous information sources, to reveal their relations and improve rumor detection. Over the social network, a GCN encoder was used to aggregate input from neighbors, and a CNN encoder was utilized to produce semantic representations. For the experiments, Twitter, Weibo, and PHEME datasets were used. The researchers found

their framework (SRD) to consistently outperform all the baselines on all datasets. Furthermore, the researchers' proposed method and Bi-GCN performed better than RvNN and PPC (RNN + CNN).

In like manner, [33] focuses on post-related features such as user-based, content-based and lexical-based features as well as post sequences. The proposed method uses essential features and combines two DL models. As manual feature selection for an ML approach is tedious, the authors use a DL-based approach to overcome the shortcomings of an ML-based approach. The authors implemented 4 models: BiLSTM_Embed, Lex_PCA, UCL_PCA, and BiLSTM_UCL. Among these models, BiLSTM_UCL performed the best.

Thenceforth, [25] uses main keywords and connects and searches for those keywords across the internet to see if a piece of news is authentic or unreliable. The authors used RF, FM classifier, Linear SVC, and LR as their models and compared them using 2 and 4 cores in their distributed system. The authors found that a higher number of cores did not always produce a higher accuracy score. Lastly, the authors compared the models across different performance metrics: Precision, Recall, and F1 score. They found the FM classifier, Linear SVC, and LR to be the most accurate and RF the least accurate. [30] discusses how pre-existing fake news detection systems may work for more local contexts. For this, they train models on a dataset of South African (SA) fake news and compare it with models trained on US fake news datasets. The models they use are Logistic Regression (LR) and LSTM. The researchers found that LSTM performed better than logistic regression, however, training and testing on different countries' datasets produced poorer results. The models trained on SA datasets performed best for SA fake news detection, and models trained on US datasets performed best for US fake news detection.

Finally, in [22] the authors encourage using of DL for rumor detection for future research because there has been relatively less research done in this field using DL compared to ML and explain how DL has some certain advantages over ML, for example, it can automate the process of feature selection and has the capability of high dimensional feature extraction. DL is also not heavily dependent on data preprocessing and has better accuracy than ML according to the authors. Therefore, we studied several papers that specifically cover different approaches to deep learning.

# Chapter 3

# Methodology

Our research is primarily divided into two sections. The first one is the comparative study part and the second one is utilizing the result acquired from the comparative study in order to create a hybrid model using ML and DL. In regard to the datasets to be used for our study, we have collected various publicly available datasets constructed mostly in English along with a widely acclaimed large Bangla dataset for rumor detection. However, in the absence of variety in Bangla-labeled datasets, a major portion of our research and testing has been done with datasets composed in the English language.

Once we acquired the datasets, we performed data pre-processing in two different approaches to test each of them out on the models and continued with the one that helped us achieve better output.

Having successfully pre-processed our datasets, we created the four different ML models using Sci-kit Learn's [1] existing classifiers. We then trained the models on all datasets for each pre-processing approach. For training and testing, we evaluated a score by cross-validation using Sci-kit Learn. A 5-fold cross-validation was utilized by passing in the scoring parameter for the "accuracy", "precision", "recall", and "F1-score" for each of the models to acquire their corresponding performance. According to [34], the higher the number of folds is in a cross-validation, the lower prediction error we will encounter. Because it means the model is trained by feeding a larger dataset and evaluated on a smaller test fold. A lower k, on the other hand, indicates the opposite. The possibility for the data distribution in the test fold to deviate from the training set is greater in this case, and we should consequently expect a higher average prediction error. When doing cross-validation, a maximum of 10 folds are widely utilized as stated by the author. Therefore, we chose 5 folds because it is the intermediate value. TensorFlow [3] and Keras [2] were used to implement the DL models, and the evaluation for them has been done in the same way as the ML models.

Having acquired our comparative study results, we have created our Ensemble ML model by stacking the four ML models (RF, NB, DT, and SVM) where we used the best-performing ML model as our final classifier. On the other hand, we have implemented our hybrid model by combining the ML and the DL model with the highest performance from our comparative study.

## 3.1 Datasets

For our research, we have currently used 4 news detection datasets. Among the datasets, all of them (Real or fake news dataset, COVID19 Fake News Dataset NLP [28], and BanFakeNews [15]) were collected from Kaggle except the ISOT Fake News Dataset [6] [5], which was collected from the website of the University of Victoria. All the datasets are binarily labeled (Real and Fake). For the COVID-19 Fake News Dataset, we took the tweet body, and for the rest, we extracted the title and text of the articles. Description of the used datasets are presented below in table 3.1:

| Name | Source | #Data | Labels | Links |
|---|---|---|---|---|
| Fake or real news | Kaggle | 6,060 | Real 50% Fake 50% | Link |
| COVID-19 Fake News | Kaggle | 6,420 | Real 52% Fake 48% | Link |
| ISOT Fake news | University of Victoria | 44,898 | Real 48% Fake 52% | Link |
| BanFakeNews | Kaggle | 49,977 | Real 97%, Fake 3% | Link |

Table 3.1: Summary of the datasets used

The following bar graph in figure 3.1 can help us visualize the data and label ratio better:

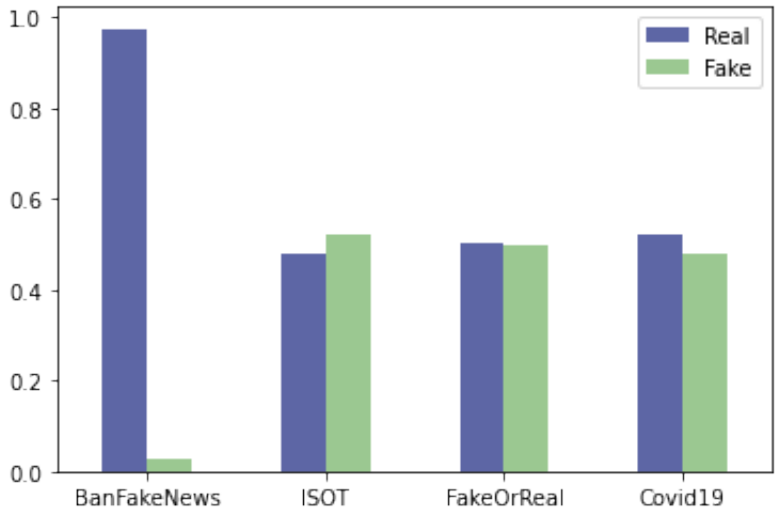
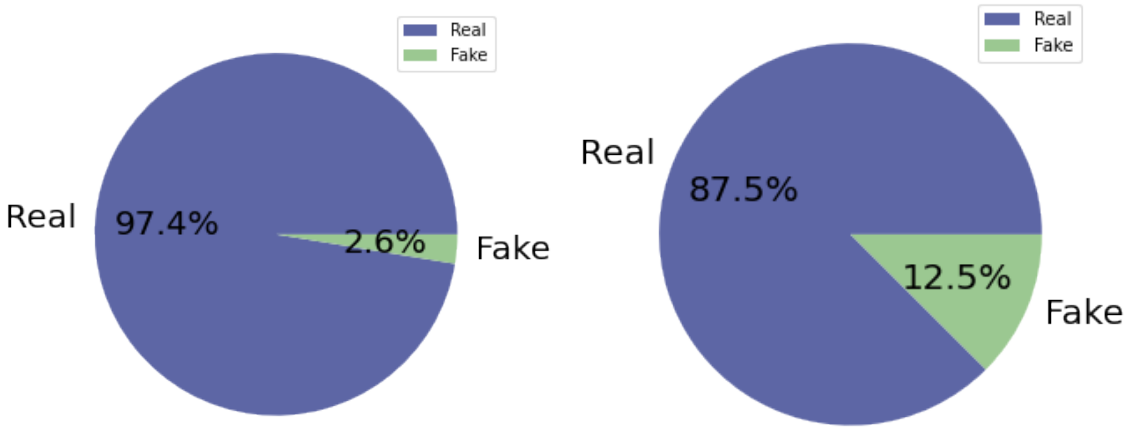



Figure 3.1: Dataset label ratio visualization

If we look at figure 3.1, we can see that all the datasets have an equal distribution of real and fake data aside from BanFakeNews. We see an immense disparity here. In order to balance the ratio, we modified the number of real data to be 7 times the fake data that are present in the dataset while training and testing. The aftermath can be visualized from the following pie charts, figure 3.2:

In figure 3.2 we can observe the irregular real-fake data ratio being present. Even though after modifying the labeling ratio, the result is not the most desired one,



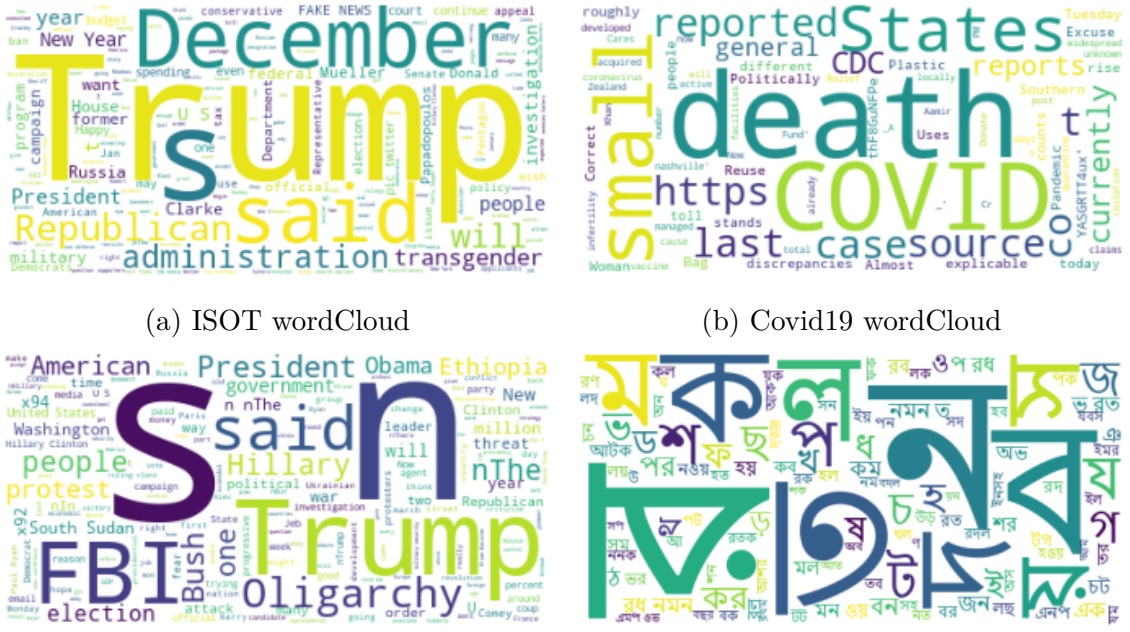

(a) Default label ratio

(b) Modified label ratio

Figure 3.2: BanFakeNews Label Ratio

still it is more balanced than before. Therefore, we proceeded with the modified one.

Furthermore, to depict the significant textual data points in terms of their importance and frequency, we generated a word cloud representation for each dataset in figure 3.3.




(a) ISOT wordCloud

(b) Covid19 wordCloud

(c) FakeOrReal wordCloud

(d) BanFakeNews wordCloud

Figure 3.3: Dataset Wordclouds

## 3.2 Data Pre-processing

For pre-processing the data, we first dropped the rows with null values as well as the columns which are not necessary to train-test our models. Then using LabelEncoder we assigned each label a unique integer. For our case, we encoded Fake→0 and Real→1. For those data sets which had separate CSV files for fake news and real news, we merged the two files and shuffled the rows to distribute fake and real news in random order. Then we used NLTK and CountVectorizer tools to process our dataset. For pre-processing with NLTK, we first divided our content into X and Y sets. Where X contains the features which in our case are the headline and the whole article body itself while Y contains the encoded labels. Afterward, the data was converted to lowercase, stop words were removed, and stemming was applied, which is a process of removing inflection in words to their basic forms. All these steps aid in the preparation step and subsequent stages of the NLP application while parsing. Lastly, the textual data was converted into feature vectors using TfidfVectorizer which aids in the conversion to a TF-IDF feature matrix from raw documents. We also used CountVectorizer, a tool equivalent to TfidfVectorizer, but it converts a collection of textual data into a token count matrix. We used two different tools in order to find out which one helps to achieve better performance. Furthermore, we concluded from our analysis that for most of the cases, using NLTK and TfidfVectorizer resulted in a higher score.

Similar pre-processing steps were followed for all the English datasets. However, the BanFakeNews dataset had to be processed differently for being in Bengali. For this, we took the help of the bnlp_toolkit. To tokenize the data, we used the BasicTokenizer in the package and its corpus to remove the stopwords and punctuations. After cleaning the text, we used the TfidfVectorizer, just like the English datasets, to vectorize the texts. Finally, to train the ML models we converted vectorized data into a dense matrix for all datasets.

# Chapter 4

# Implementations and results

In this section, the process of implementing each of the models is described, along with the results they yielded for each of the datasets. The results are presented in the tabular form alongside confusion matrices of each model corresponding to the datasets.

## 4.1 Implementation

Implementing the models consisted of three steps: firstly, all the data were pre-processed using the approaches mentioned in the prior section, and the models were then instantiated and fitted. Lastly, to derive the results, 5-fold cross-validation was performed to get the most accurate results.

### 4.1.1 Machine Learning Models

To implement all the ML models, the Sci-kit Learn library [1] was used. For the DT model, the class DecisionTreeClassifier, which can conduct classifications of more than one class on a dataset, was employed. The DecisionTreeClassifier, similar to other classifiers, takes two inputs of arrays: an array X, storing the training samples, and an array Y, holding the class labels for the training data. The X array can be sparse or dense, of the form n samples×n features while the array Y should be of integer values with a shape of n samples. The RF model has been built with RandomForestClassifier where an ensemble of trees is constructed, each using a sample that was taken from the training set and replaced. As with other classifiers, forest classifiers require the fitting of two arrays: an array Y of n samples, storing the target class labels for the training samples, and an array X of n samples, comprised of the training samples' features. To implement the NB model, GaussianNB was used, which uses the Gaussian Naive Bayes classification technique. It is believed that the features conform to a gaussian likelihood. As for the SVM model, sklearn's SVC was utilized. The mathematical description of the SVM's decision function reveals that it is dependent on a subset of the training data known as the support vectors. SVC, like the previous classifiers, requires two input arrays: an array X of the form n samples×n features holding the training data and an array Y comprising class labels which can be strings or integers, of n samples.

### 4.1.2 Deep Learning Models

The DL models were built using the Keras API [2] and TensorFlow library [3]. In order to implement BERT, we used the BERT preprocessor and encoder from Tensorflow-hub. With an 80:20 ratio of train-test, we divided the data to first train and then test our models. After that, we constructed our neural network, where each layer's output was passed down as an argument for the layer after it. Our BERT model consists of one input layer, which makes up the model. This layer would depict every sentence that is supplied into the model. Then we have our bert_preprocess layer to enter our data to prepare the text and the bert_encoder layer, to provide the BERT encoder with the preprocessed tokens. Also, we have, 1 dropout layer, with a 0.2 dropout rate. It receives the BERT encoder's pooled output before passing on to 2 dense layers: one with 10 neurons and the other with 1 neuron, respectively. The first dense layer employed a ReLU activation function and the second used a sigmoid activation function. The layers can be seen in 4.1

```
import tensorflow as tf

# Input Layers
input_layer = tf.keras.layers.Input(shape=(), dtype=tf.string, name='rumor')

# BERT Layers
processed = bert_preprocess(input_layer)
output = bert_encoder(processed)

# Neural Network Layers

layer = tf.keras.layers.Dropout(0.2, name='dropout')(output['pooled_output'])
layer = tf.keras.layers.Dense(64,activation='relu', name='hidden')(layer)
layer = tf.keras.layers.Dense(1,activation='sigmoid', name='output')(layer)


model_bert = tf.keras.Model(inputs=[input_layer],outputs=[layer])
```

Figure 4.1: BERT Implementation Code

Lastly, the Adam optimizer, a binary cross-entropy loss, and an accuracy performance metric were used to train the model over five epochs.

To implement CNN, we began by importing the required libraries made available by Keras. To order for the kernel filter and stride to fit in the input well, we padded our input data. We set a 100-word maximum for each review input and use a Keras function to pad sequences.

The input we accept is defined as the maximum length of a review permitted. The model initially consists of an embedding layer in which we will discover the embeddings of the top 10,000 words into a 32-dimensional embedding. Convolutional and max-pooling layers are then added. We then add dense layers and flatten those matrices into vectors which can be observed in 4.2.

Then we compiled, fitted, and evaluated the model using the same parameters as BERT.

```
# Building the CNN Model
model_cnn = Sequential()
model_cnn.add(Embedding(max_vocab, 32, input_length=max_words))
model_cnn.add(Conv1D(32, 3, padding='same', activation='relu'))
model_cnn.add(MaxPooling1D())
model_cnn.add(Flatten())
model_cnn.add(Dense(64, activation='relu'))
model_cnn.add(tf.keras.layers.Dropout(0.5))
model_cnn.add(Dense(1, activation='sigmoid'))
```

Figure 4.2: CNN Implementation Code

Lastly, we implemented RNN again by padding our input. The model has an embedding layer in for the embedding of the words into a 32-dimensional embedding, bidirectional wrappers, dense layers, and dropouts were also applied to the inputs as seen in 4.3 .

```
# Building the RNN Model
model_rnn = tf.keras.Sequential([
    tf.keras.layers.Embedding(max_vocab, 32),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64,  return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(16)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1)
])
```

Figure 4.3: RNN Implementation Code

We employed an early stop, which terminates when the validation loss stops improving, and, compiled, fitted, and evaluated the model using the same parameters as BERT and CNN. The parameters are given in 4.4

```
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=2,
                                              restore_best_weights=True)
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 4.4: Code for compiling the models

### 4.1.3 Machine Learning Stacked Model

In order to achieve better accuracy than the default machine learning and deep learning models we tried two approaches. The first method is stacking the four ML models that we initially chose [RF, DT, SVM, NB]. But since RF performed the best among the fours chosen models, we used it as our final estimator. Using a classifier for each model from sklearn we made a list of estimators for the DT, NB, and SVM illustrated in 4.5.

Then we imported the StackingClassifier from sklearn.ensemble where we passed our parameters which are the estimators' list and RFC as our final estimator and performed 5-fold cross-validation to calculate the accuracy.

```
dtc =  DecisionTreeClassifier()
rfc = RandomForestClassifier()
gnb = GaussianNB()
svc = SVC(gamma=100, cache_size = 2000)
clf = [('dtc',dtc),('gnb',gnb),('svc',svc)] #list of (str, estimator)

from sklearn.ensemble import StackingClassifier

stack_model = StackingClassifier( estimators = clf,final_estimator = rfc)
score_ML_s = cross_validate(stack_model, X_stack_vec_dense, Y_stack, cv = 5,
                            scoring = ['accuracy', "f1", "precision", "recall"])
```

Figure 4.5: Ensemble Stack Implementation code

### 4.1.4 Hybrid Model

Looking into the result section we can see that both CNN and RNN outclassed BERT by a long margin. However, the average results of RNN and CNN were almost similar with a slight edge towards RNN. While RNN was clearly better, in case of some datasets CNN scored higher. However, we know that CNNs are frequently employed to solve issues involving spatial data, like graphics. Where, on the other hand, RNNs excel at processing temporal and sequential data, like text. Therefore, for a text classification problem such as rumor detection, RNN was chosen for the final hybrid model.

```
def recall(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

Figure 4.6: Code for custom scoring functions

There are several approaches to creating a predictive model combining multiple algorithms. The primary ensemble techniques are Bagging, Boosting, and Stacking. Bagging consists of base learners running on samples and then aggregating their outputs. The RF model already utilizes this method where it combined multiple decision trees. In Boosting the final estimator tries to learn from the mistakes of its input generator models to predict more accurately in its tests. The AdaBoost model is an example that uses this approach. Finally, in the Stacking approach, a model learns from the outputs of other sub-models or weak learners and combines the inputs to produce more accurate output predictions. For our hybrid model, we went with the Stacking approach for combining the DL and ML models. In the previous section we have already discussed the process of creating a stacked model

from multiple ML models using the same approach.

Our DL-ML hybrid is composed of RNN and RF. The loss function is "binary cross-entropy", the optimizer is "adam", and the error measure is "F1-score". As F1-score isn't accessible in Keras, we developed the function ourselves along with functions to calculate recall and precision as well which is given in 4.6.

After saving and loading the model, we began stacking. We start by training the meta learner by presenting the weak learner instances from the test set. The build_meta_dataset() function in 4.7 implements the task of creating the dataset to fit the meta learner, i.e: the RF model. The new dataset is created from the predictions of the RNN model.

```
def build_meta_dataset(models, input_train):
    metaX = None
    for model in models:

        yhat = model.predict(input_train, verbose=0)

        if metaX is None:
            metaX = yhat
        else:
            metaX = dstack((metaX, yhat))

    return metaX
```

Figure 4.7: Code for building meta dataset

Now, we have constructed the dataset that would be used to train the meta-learner. For training, we fitted the meta-learner to the data using the fit_hybrid_model() function in 4.8 and evaluated the model.

```
def fit_hybrid_model(models, input_train, input_label):
    meta_train = build_meta_dataset(models, input_train)
    ml_clf = RandomForestClassifier() #meta learner
    ml_clf.fit(meta_train, input_label)
    return ml_clf
```

Figure 4.8: Code for fitting the meta learner

We choose the RNN model instead of the better-performing CNN model because RNN is better suited for text classification, especially in the long run. While CNN performs better in spatial data, i.e: image classification, RNN excels at temporal or sequential data such as text classification. Moreover, the aim of this research is to derive a hybrid model that outperforms the baseline models efficiently. In our implementation, RNN required much less computational power than CNN. Because of all these factors, we have chosen RNN as the weak learner of our proposed hybrid model. For the meta-learning model, RF has been chosen simply because it outperformed the other chosen ML models in every dataset train-testing.

## 4.2 Results & Analysis

This section contains the accuracy scores acquired from our testing of the different models in tabular form.

| Dataset | Models | | | | |
|---|---|---|---|---|---|
| | DT | RF | NB | SVM | Ensemble |
| Fake or real news | 81.05% | **89.85%** | 79.36% | 51.47% | 78.73% |
| COVID-19 fake news | 88.02% | **92.72%** | 86.57% | 53.82% | 88.32% |
| UVIC ISOT | 98.72% | 98.36% | 82.23% | 52.67% | **98.93%** |
| BanFakeNews | 92.23% | **93.31%** | 85.88% | 88.04% | 90.97% |
| **Average** | 90% | **93.56%** | 83.51% | 61.5% | 89.24% |

Table 4.1: Accuracy of the ML models

Firstly, evaluating the accuracy scores of the ML models from table 4.1 we observe that, RF has outperformed every other ML model for every dataset except the "UVIC ISOT" dataset with a marginally lower score than DT and the Ensemble model. However, on average, RF had the highest accuracy among the ML models with a score of 93.56%. DT followed with a respectable 90%. The Ensemble model performed close to DT with an average accuracy of 89.24%. A significant downfall is seen with the NB model acquiring only 83.51% accuracy. It performed especially badly for the "FakeOrReal News" dataset. The worst performer was the SVM model with an average accuracy of 61.5%. A surprisingly high score was seen in the case of the Bengali dataset, however. The results clearly show that RF is the best choice for rumor detection among the other ML models from our testing. As such we have chosen this as the meta-learner among the ML models for our hybrid model.

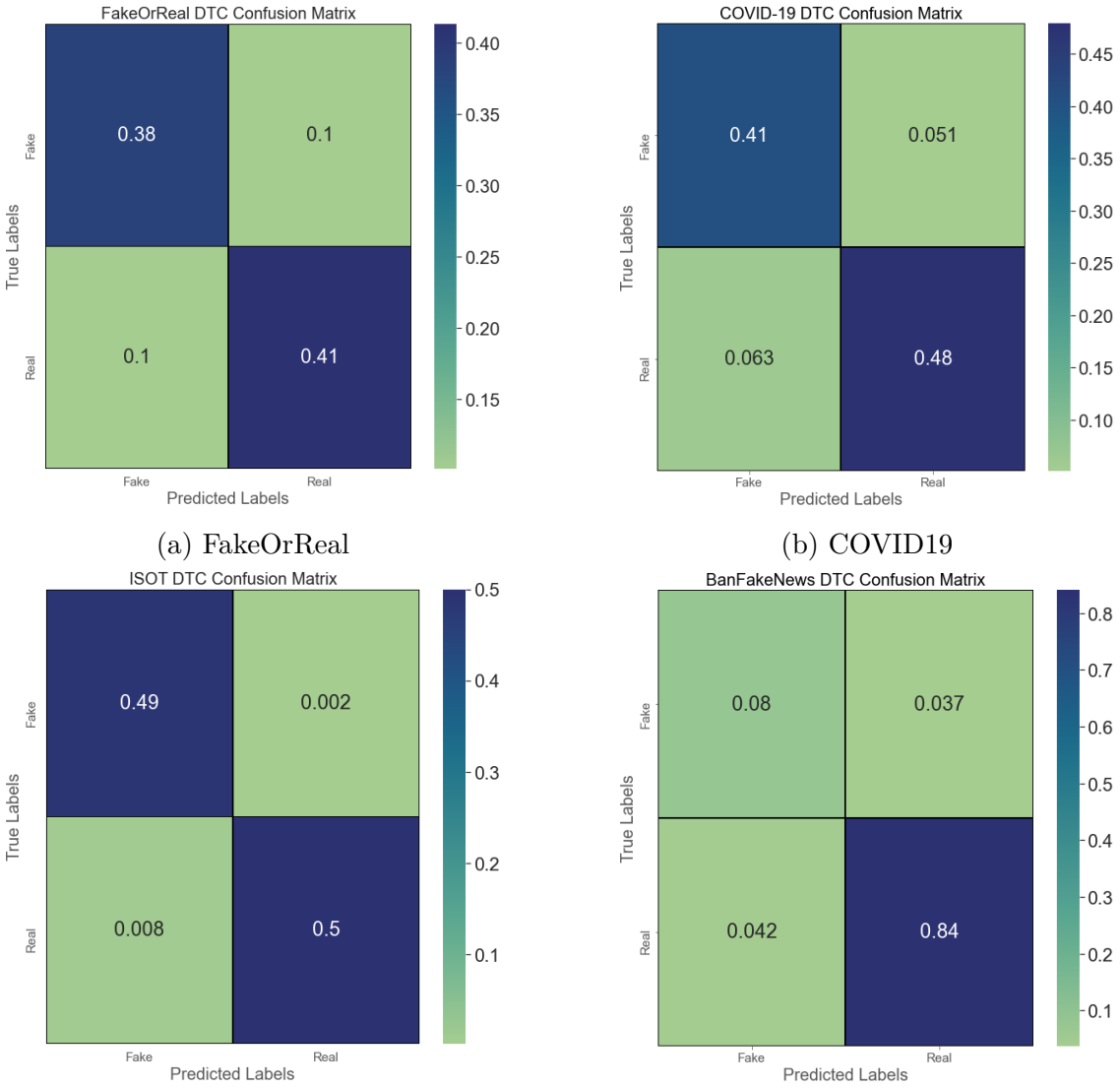
| Dataset | Models | | | | |
|---|---|---|---|---|---|
| | DT | RF | NB | SVM | Ensemble |
| Fake or real news | 81.05% | **89.77%** | 80.11% | 68.89% | 78.26% |
| COVID-19 fake news | 88.59% | **93.06%** | 87.25% | 69.39% | 88.76% |
| UVIC ISOT | 98.72% | 98.35% | 81.75% | 68.77% | **98.97%** |
| BanFakeNews | 92.23% | 93.31% | 85.88% | 88.04% | **94.98%** |
| Average | 90.15% | **93.62%** | 83.75% | 73.42% | 90.24% |

Table 4.2: F1-score of the ML models

Aside from accuracy, the models' performance can be further observed through F1-scores. Table 4.2 represents the F1-scores for all the ML models and here too the RF model comes out on top with an average F1 of 93.62%. This means that RF also has the best precision and recall for rumor detection compared to other models. The Ensemble model sits on the second position with an average F1 of 90.24%, which used to be occupied by DT in the accuracy measure. This time DT follows with a close 90.15% score. A similar jump down to lower score is seen here just like the accuracy measure with NB scoring 83.75%. The worse performer by far is the SVM

model again, with a F1 of 73.42%.

To get an idea of how the models achieved these scores, we need to look at their confusion matrices.




(a) FakeOrReal

(b) COVID19

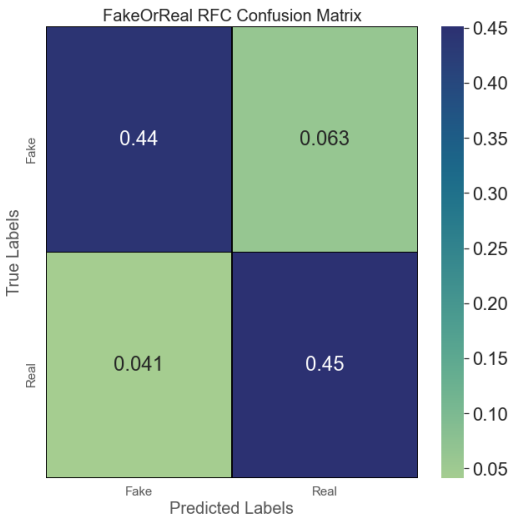
(c) UVIC ISOT

(d) BanFakeNews

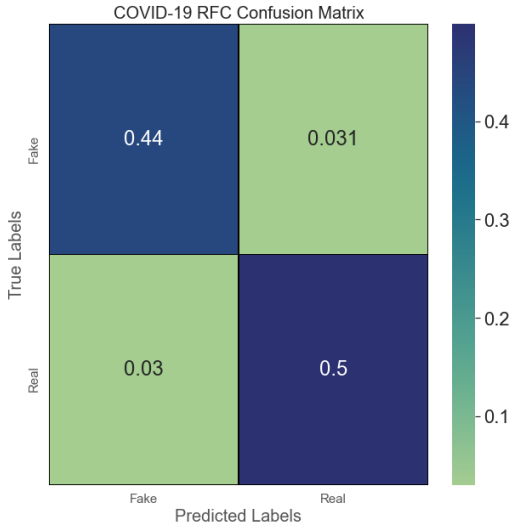
Figure 4.9: Confusion matrices of the DT Model

From figure 4.9 the performance of DT can be visualized. In all instances, it had little trouble correctly identifying the true labels which are expected as it comes second to RF. Another important factor is that the DT model seems to have less trouble correctly identifying the Real texts as opposed to the Fake ones. The matrix for the BanFakeNews dataset can be misleading with its lighter Fake-Fake square but we need to keep in mind that the amount of fake data in the dataset is very low to begin with. As a result, a similar imbalance of colors is seen for every model.

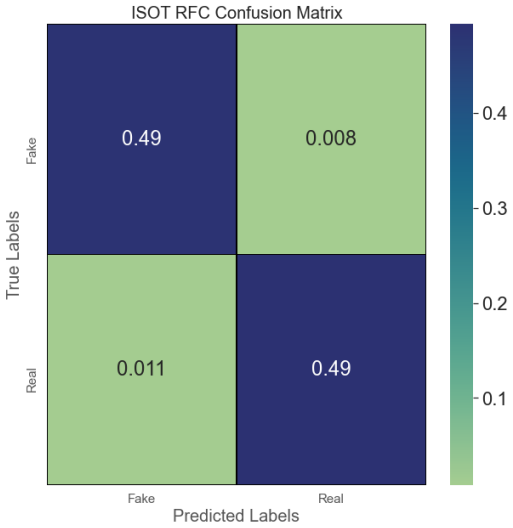
The performance of RF can be seen in figure 4.10 where it is clear that it had barely any trouble correctly predicting the labels. Although it performs better than DT,



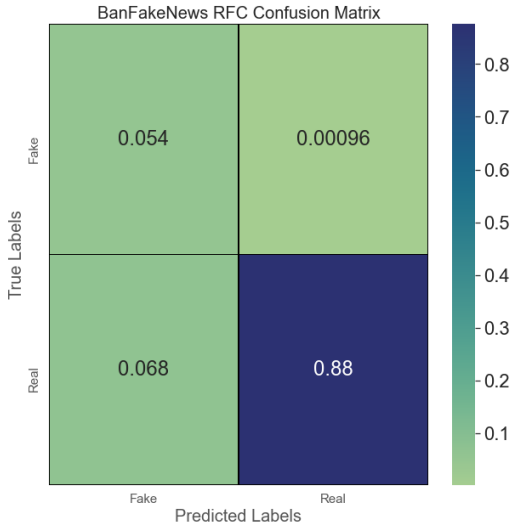

(a) FakeOrReal




(b) COVID19



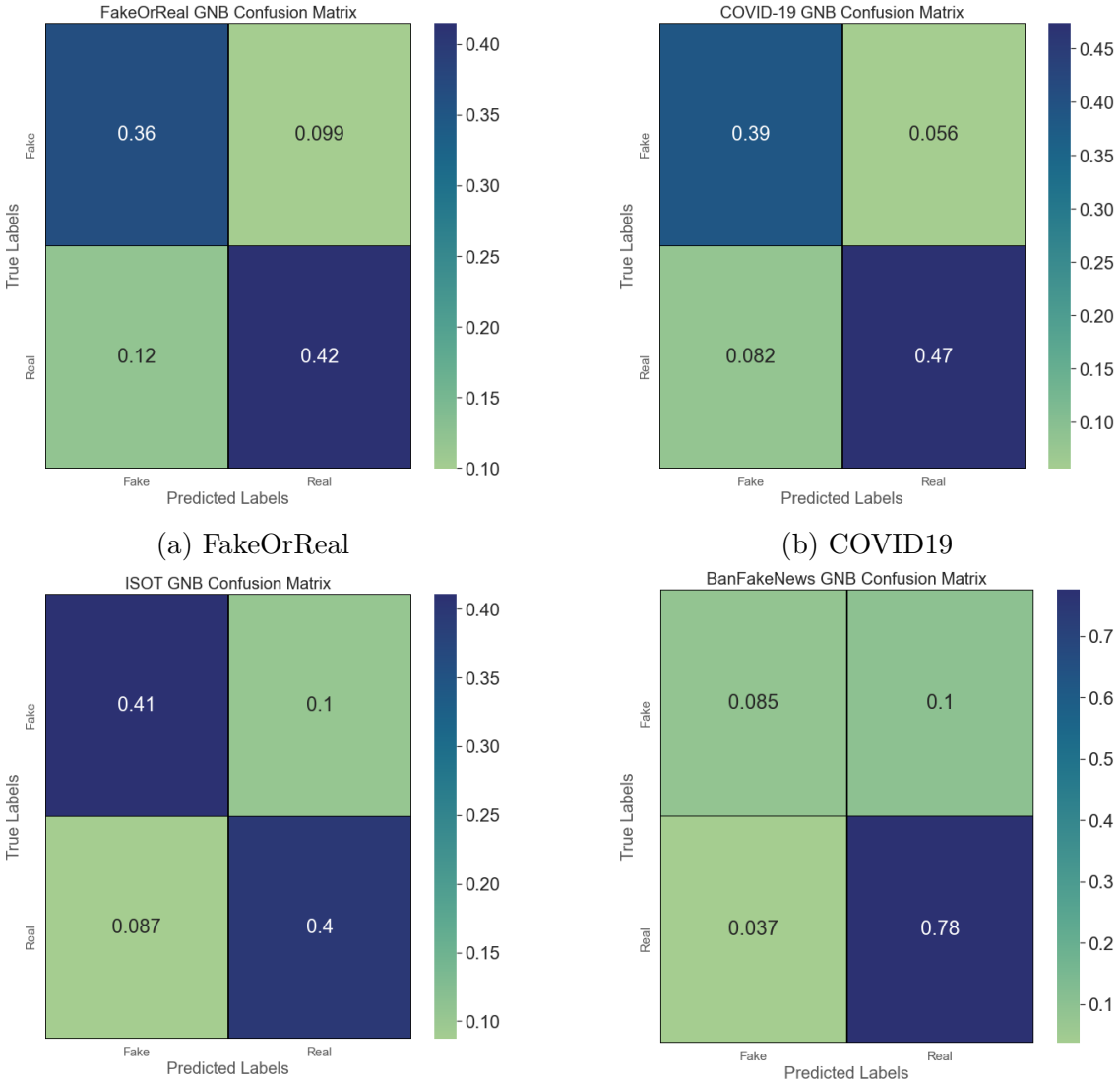

(c) UVIC ISOT




(d) BanFakeNews

Figure 4.10: Confusion matrices of the RF Model

it struggles with fake data as well.




(a) FakeOrReal

(b) COVID19

(c) UVIC ISOT

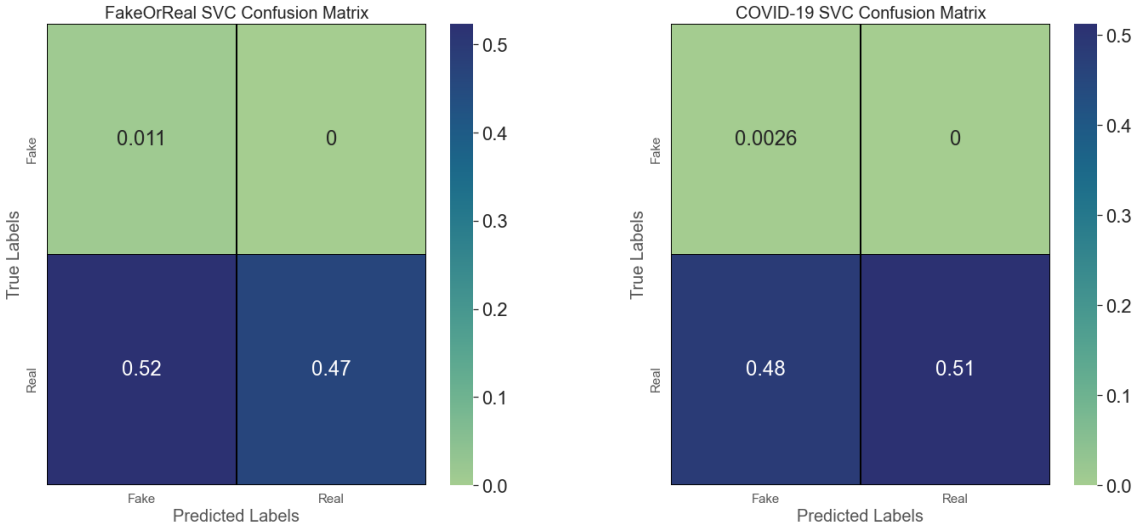
(d) BanFakeNews

Figure 4.11: Confusion matrices of the NB Model

Figure 4.11 illustrates why the NB model failed to score as high as the previous two models. Not only did it make more mistakes in predicting fake texts, it was less useful in labeling real data as well. A drop in performance in both cases led to the model's lower metric in accuracy and F1.

The performance of SVM, with the least impressive scores, can be observed in figure 4.12. The main reason it scored the least can be attributed to it labeling real data as fake. However, it is interesting that it consistently managed to identify all the real data. It seems it has a tendency to label everything as real. Further research can be done to explore the characteristics of SVM.

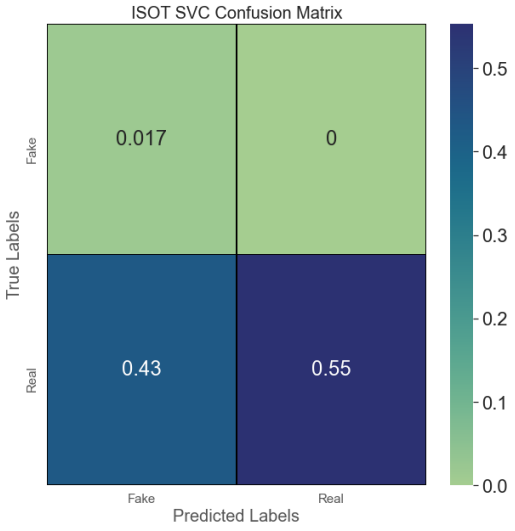
Finally, the confusion matrices of the Ensemble ML model can be seen in figure 4.13. This is our first attempt at creating a model that could possibly perform better than



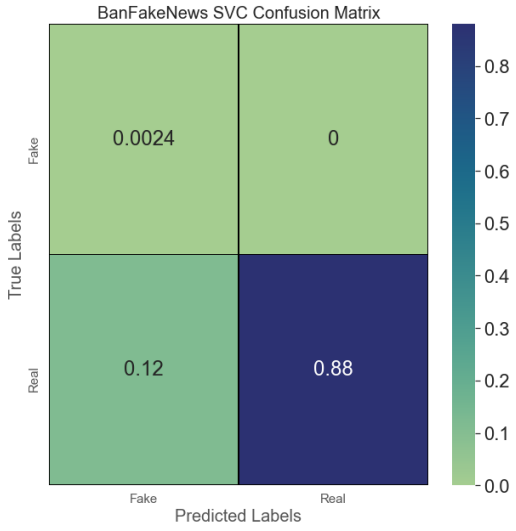

(a) FakeOrReal




(b) COVID19

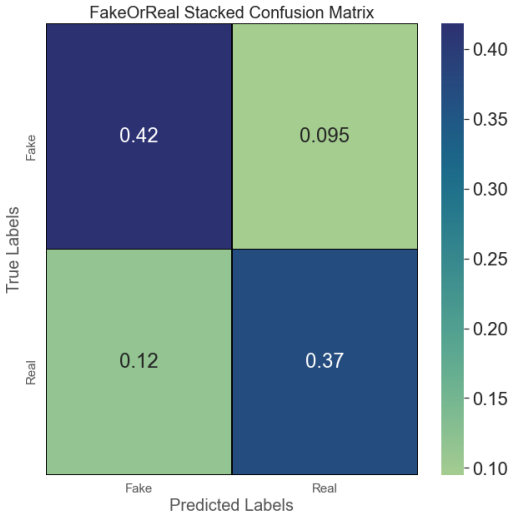



(c) UVIC ISOT

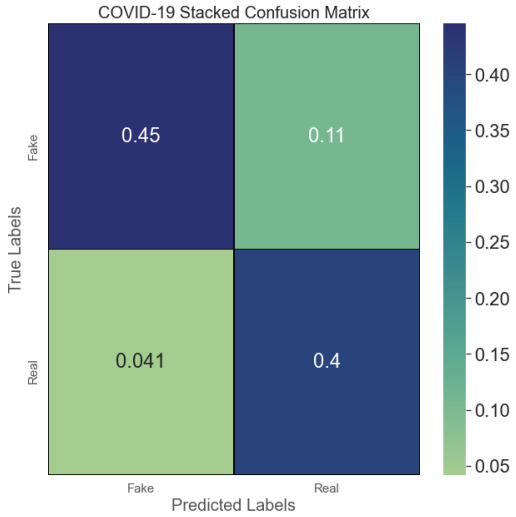



(d) BanFakeNews

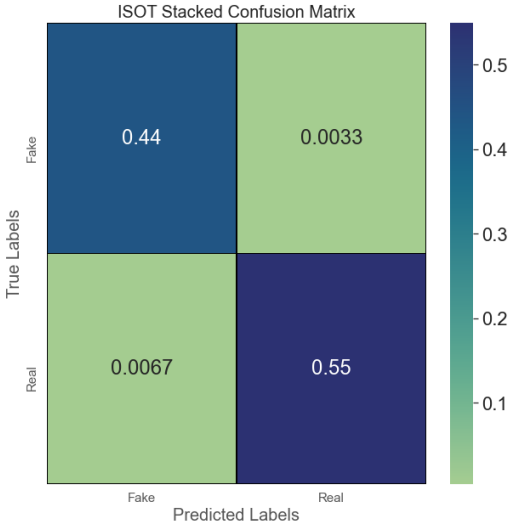
Figure 4.12: Confusion matrices of the SVM Model



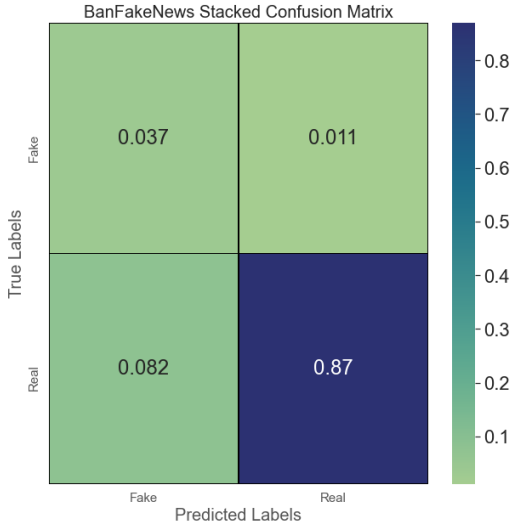

(a) FakeOrReal




(b) COVID19




(c) UVIC ISOT




(d) BanFakeNews

Figure 4.13: Confusion matrices of the Ensemble ML Model

the other ML models. However, it could not surpass either DT or RF. The reason for this could be the presence of SVM. Compared to those, this model had more difficulty in correctly identifying both fake and real data.

| Dataset | Models | | |
|---|---|---|---|
| | BERT | RNN | CNN |
| Fake or real News | 72.37% | 86.74% | **87.21%** |
| COVID-19 fake news | 82.39% | **91.67%** | 91.58% |
| UVIC ISOT | 84.3% | **96.9%** | 95.4% |
| BanFakeNews | 95.67% | 95.14% | **96.24%** |
| **Average** | 83.68% | **92.61%** | 92.60% |

Table 4.3: Accuracy of the DL models

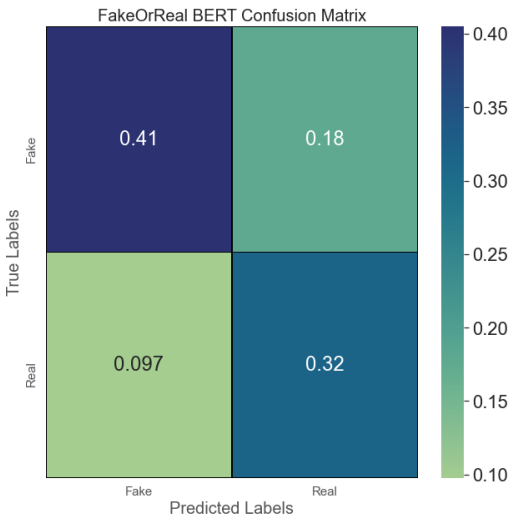
Moving on to the accuracy of the DL models from table 4.3, it is evident that RNN and CNN performed the best with average scores of 92.61% and 92.60%. However, we did not consider CNN for our hybrid model as it is not the best suited for text classification in the long run. For detecting rumors well, we need a model that can learn textual patterns better. Such a model is RNN which, on its own, had marginally higher accuracy on average. It even outperformed the CNN model in half the datasets. Since it has quite a respectable accuracy and is an optimal choice for text classification, we took this as the weak learning neural network for our hybrid model. BERT had the lowest score among all the DL models with an average of 83.68% accuracy. The only exception to this was the "BanFakeNews" dataset where we implemented a pre-trained BanglaBERT model from its authors. Only in that instance did it perform fractionally better than RNN, just shy of CNN.

| Dataset | Models | | |
|---|---|---|---|
| | BERT | RNN | CNN |
| Fake or real News | 73% | 86.09% | **86.84%** |
| COVID-19 fake news | 81.25% | **91.5%** | **91.5%** |
| UVIC ISOT | 84.5% | **97%** | 96.5% |
| BanFakeNews | 85.5% | **97.18%** | 96.73% |
| **Average** | 81.06% | **92.94%** | 92.89% |

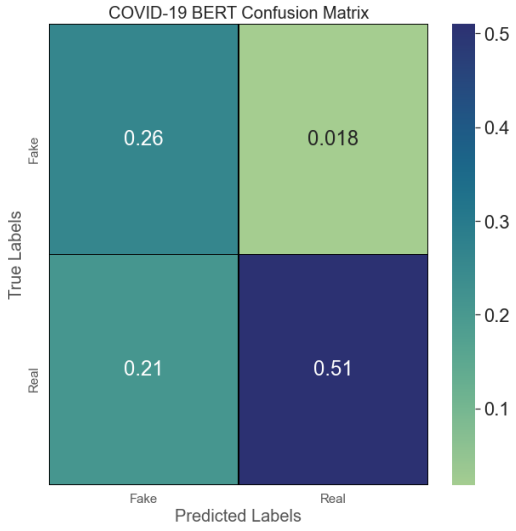
Table 4.4: F1-score of the DL models

The F1 scores from table 4.4 give further insight into the DL models' precision and recall. In this case, as well, RNN came out on top with an average score of 92.94%. CNN resided close by with 92.89%, a minuscule difference. The worst performer by far was BERT, scoring only 81.06%. CNN performed marginally better on the FakeOrReal dataset, scoring 86.84% compared to RNN's 86.09%. RNN performed equally well as CNN in the case of the COVID-19 dataset. It outperformed both the other models in ISOT and BanFakeNews.

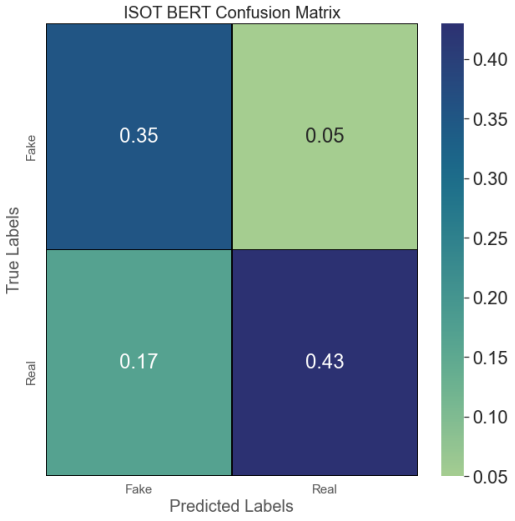
The confusion matrix from figure 4.14 shows us where the BERT model struggled. In all instances, it mixed up either real data with a fake label or fake data with the



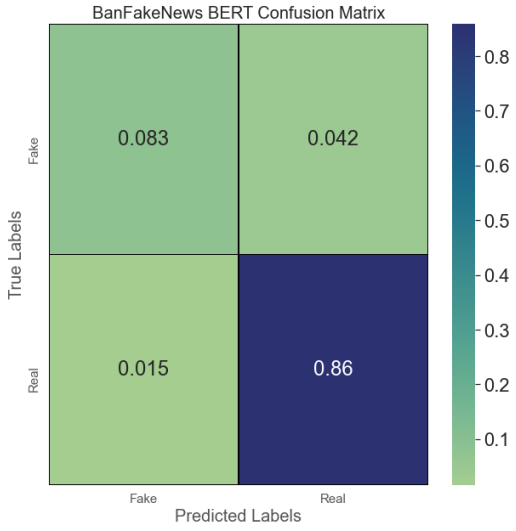

(a) FakeOrReal




(b) COVID19



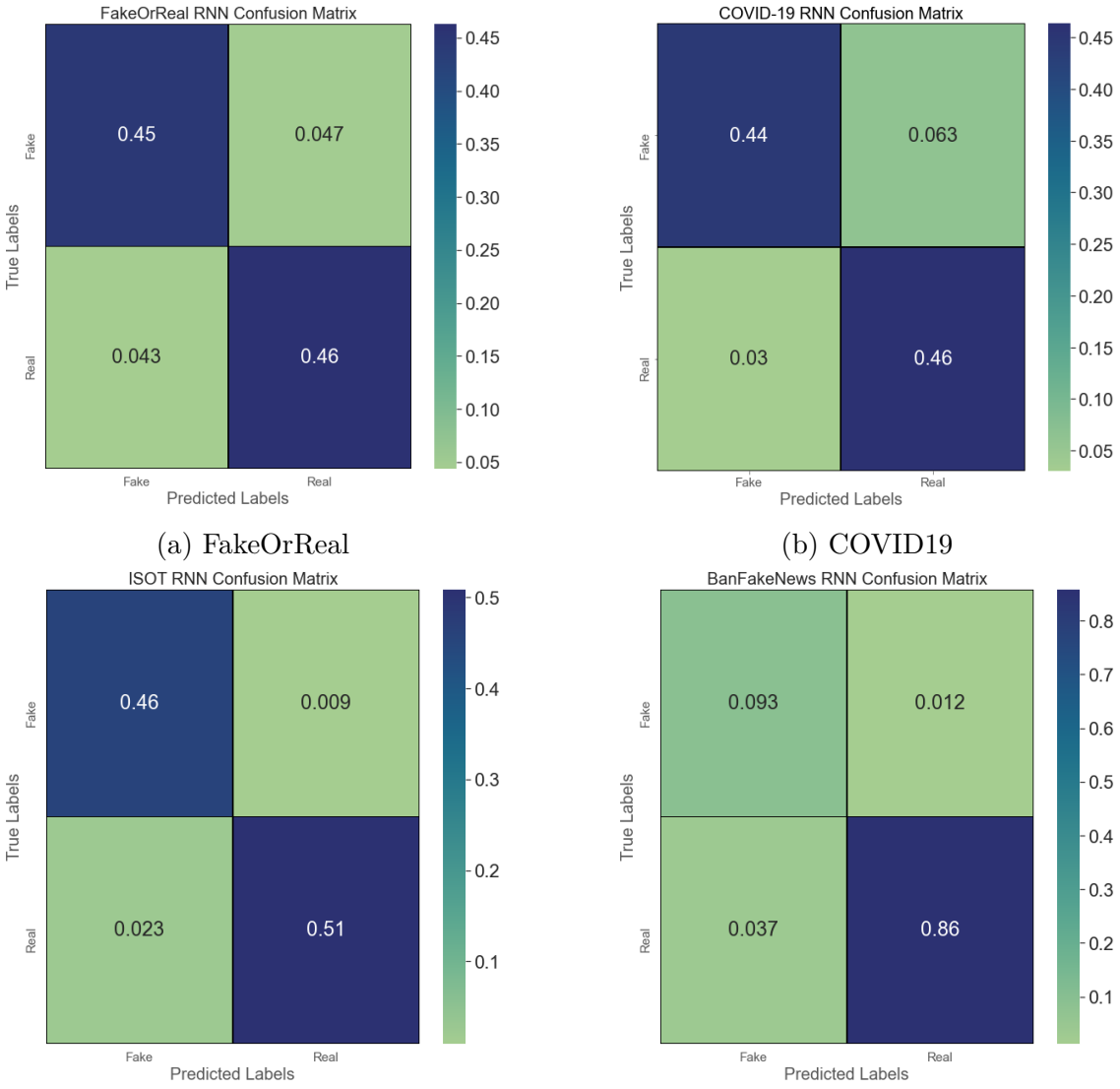

(c) UVIC ISOT




(d) BanFakeNews

Figure 4.14: Confusion matrices of the BERT Model

real label. The latter can be seen in the FakeOrReal dataset whereas in COVID-19 and ISOT the first is true. Only in BanFakeNews, had a noteworthy performance.




(a) FakeOrReal

(b) COVID19

(c) UVIC ISOT

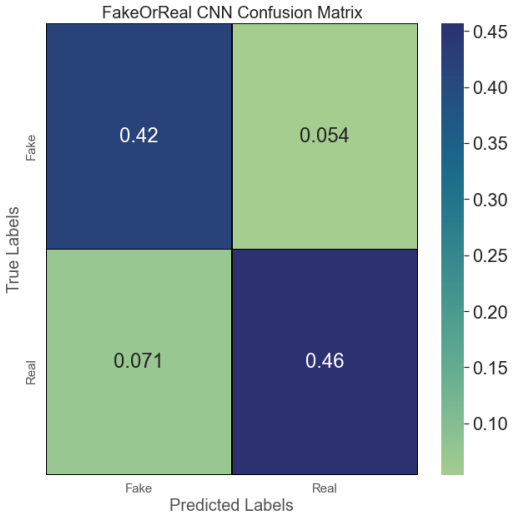
(d) BanFakeNews

Figure 4.15: Confusion matrices of the RNN Model

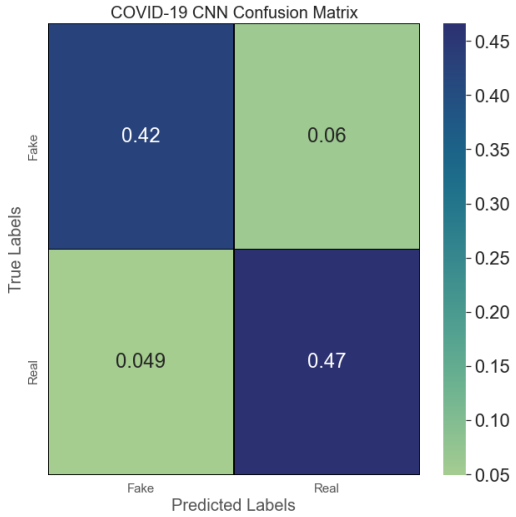
Figure 4.15 illustrates how RNN had ease in identifying the labels correctly. It barely missed any predictions in almost all cases.

The performance of the second-placing CNN model is visible in figure 4.16. Although it managed to accurately label quite well, it particularly slipped when distinguishing fake data compared to RNN. Labeling real data, it was almost on par with RNN.

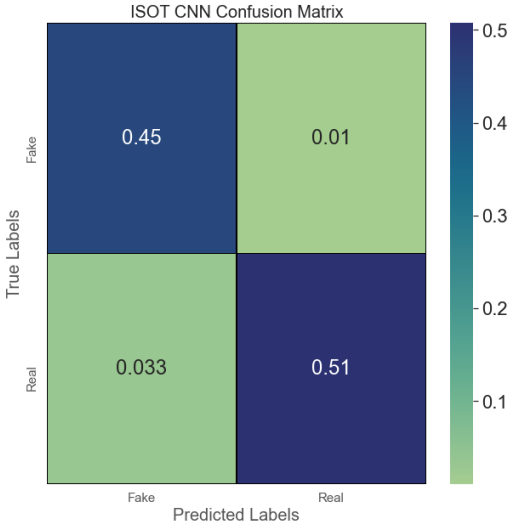
Lastly, looking at the accuracy of the proposed hybrid model from table 4.5, we see that it outperforms all other best performing baseline models for every dataset. Overall, it had an accuracy of 95.29% which is even more than the highest-scoring ML model, RF. It not only outperformed the CNN model as well, but it also did so while taking up lesser resources. Thus, it is shown that the hybrid model we



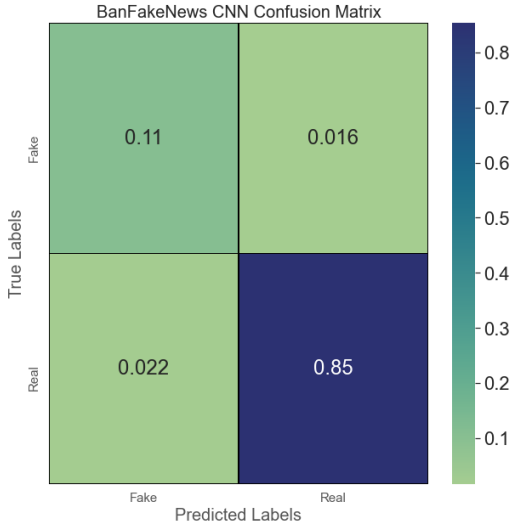

(a) FakeOrReal




(b) COVID19




(c) UVIC ISOT




(d) BanFakeNews

Figure 4.16: Confusion matrices of the CNN Model

| Dataset | RF | Ensemble | RNN | Hybrid Model(RNN+RF) |
|---|---|---|---|---|
| Fake or real News | 89.85% | 78.73% | 86.74% | **92.34%** |
| COVID-19 fake news | 92.72% | 88.32% | 91.67% | **94.47%** |
| UVIC ISOT | 98.36% | **98.93%** | 96.9% | 98.1% |
| BanFakeNews | 93.31% | 90.97% | 95.14% | **96.25%** |
| **Average** | 93.56% | 89.24% | 92.61% | **95.29%** |

Table 4.5: Accuracy comparison of the Hybrid model

suggested, achieves state-of-the-art results by performing better than all the other baseline ML and DL models. This is true for each individual dataset rather than on average.

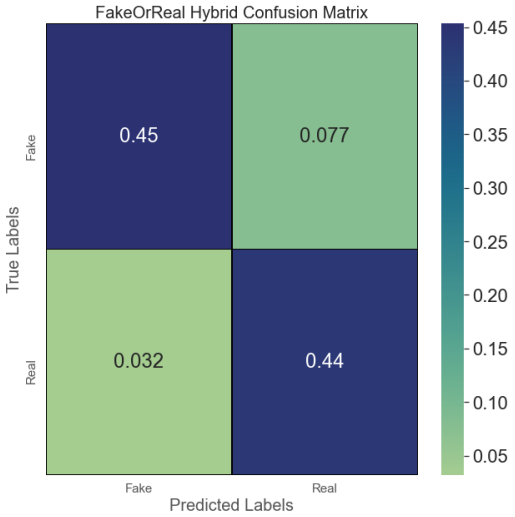
| Dataset | RF | Ensemble | RNN | Hybrid Model(RNN+RF) |
|---|---|---|---|---|
| Fake or real News | 89.77% | 78.26% | 86.09% | **92.79%** |
| COVID-19 fake news | 93.06% | 88.76% | 91.5% | **94.6%** |
| UVIC ISOT | 98.35% | **98.97%** | 97% | 97.5% |
| BanFakeNews | 93.31% | 94.98% | 97.18% | **97.85%** |
| **Average** | 93.62% | 90.24% | 92.94% | **95.68%** |

Table 4.6: F1-score comparison of the Hybrid model

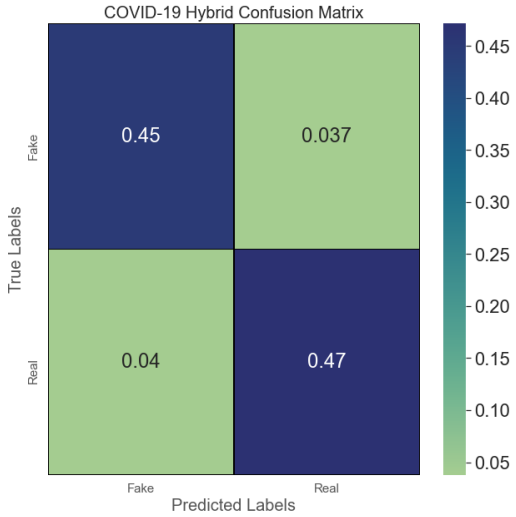
The F1 scores of the hybrid model from table 4.6 further establish its superiority over the other models. With an average of 95.68%, it scored more than all other baseline models. However, the Ensemble model was the best for the UVIC ISOT dataset with a score of 98.97% as opposed to the hybrid's 97.5%. Regardless of that, the overall higher score of the proposed model proves it has better precision and recall than the other state-of-the-art models.

Figure 4.17 visualizes how the hybrid model came out on top of others. Compared to the other confusion matrices a more consistent result can be observed. The proposed model manages to accurately label both real and fake data with ease whereas other models struggled in at least one area. Although some were better at labeling one criterion than the hybrid model, the ability to excel in both is required for real-life applications. Hence, the hybrid model created by combining RNN and RF is the optimal choice for rumor detection.

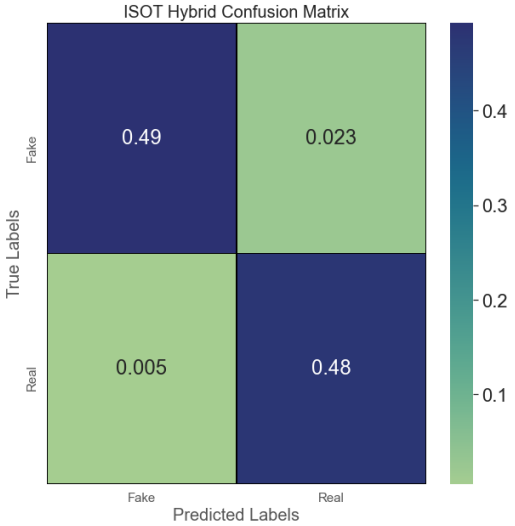
The results observed from the experiments provide us valuable insights into why the models performed the way they did. The superiority of the implemented hybrid model can also be further established by breaking down the models' performances. It has been consistently observed that the DT model was almost on par with the RF model in every instance. This comes as no surprise as RF is the culmination of multiple decision trees. For this reason, RF is able to produce much more accurate predictions on the test sets aggregating the outputs from randomly generated decision trees, creating a forest. The NB model, on the other hand, assumes all the features are continuous and that they are independent. While this is an easier algorithm to implement, the results clearly suggest that it is not the best suited for



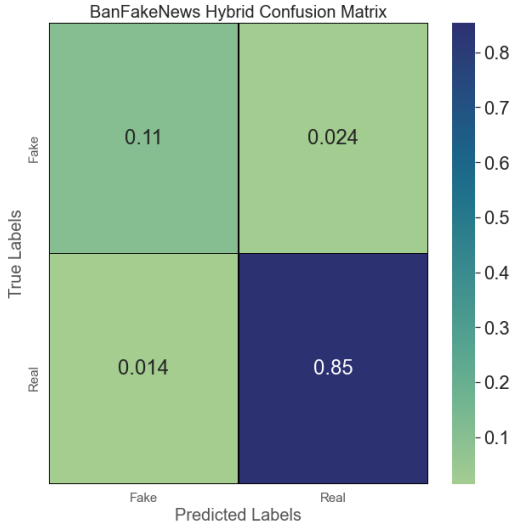

(a) FakeOrReal




(b) COVID19




(c) UVIC ISOT




(d) BanFakeNews

Figure 4.17: Confusion matrices of the Hybrid Model

making the most accurate predictions regarding rumors, at least for the datasets used here. It might be due to the fact that the features are dependent on each other in reality which would make sense in the context of rumors. But it is quite close to DT despite being less complex. By far the worst-performing model was the SVM model which utilizes a RBF kernel. It tries to produce a feature space that resembles the norm distance between a chosen point and another fixed point. However, its ultimate goal is to divide the data points into two planes like other SVM algorithms. Although this is a widely used ML model, it was not suited for rumor detection at all from the testing conducted in this paper. It also took the highest amount of time to train and test which is attributed to its quadratic growth of fit time with the number of inputs. In an effort to reduce that time and prevent overfitting, a gamma value of 100 and a higher cache size were specified. It did not perform well probably due to the increased need for resources for processing such a vast number of inputs. The results from the SVM model could be improved by using a linear kernel rather than a non-linear one. We experimented with such settings which produced similar results to the DT model. But we did not go forward with that approach as we wanted to stick to the default implementations for all models in sklearn. Overall, RF was the best performing among the ML models due to its utilization of decision trees to consider every feature and aggregate among them.

Moving on to the DL models, the BERT models achieved way lower scores than both RNN and CNN. This could be due to the fact that BERT keeps the "meaning" of a certain word fixed irrespective of the circumstance. It tries to account for the change in meaning by employing bi-directional transformers but in the case of rumors it seemed to have less impact than other learning strategies. The title for best performing DL model, was closely contested by RNN and CNN. However, research proves that CNN is better fitted for spatial data like images where input sizes are generally fixed as opposed to varying length inputs in the case of texts. Even though this has been accounted for using a 1-dimensional convolution and pooling layers, CNN is still not suited for long-term learning of patterns. For this reason, despite CNN performing similar to RNN in almost every dataset we chose RNN for the hybrid model as it is better for text classification. It also had higher accuracy in all datasets compared to CNN. RNN is more commonly used for temporal or sequential data and can handle inputs of changing size. It is also better at long-term pattern detection as it utilizes time-series information to better understand the context. For these reasons, it is observed that RNN performs very well on the chosen datasets. Over time, it can also outperform the CNN architecture in the case of rumor detection.

Considering both ML and DL approaches, it can be said that both are necessary for all sorts of classification problems. However, problems such as rumors need massive amounts of data to be detected and prevented from spreading. This is where DL outperforms ML as it can handle big data more comfortably. In fact, they perform significantly better over time. The drawback is that DL generally requires more resources than ML as it consists of more layers that learn from a hierarchy of concepts. Therefore, in order to achieve better results with a more realistic availability of resources, we propose a hybrid model composed of the best ML model and DL model from our testing. The hybrid model consistently outperformed all other state-of-the-

art baseline models. It is evident from the previous tests that the meta-learner, the RF model, manages to make more accurate predictions from a dataset constructed from the output of the RNN model. Because of the inclusion of RNN, it would also become more accurate at detecting rumors over time and the RF model would aid in the final classification.

## 4.3 Future work

We attempted to show our approach to implementing the hybrid rumor detection model in this paper that can outrun several state of the art models by analyzing the outcome of our performed comparative study. Thus far, we believe it has a lot of potential for future research. Some of them are listed below:

- Constructing the Bangla dataset to compensate for the paucity.
- Involving more models in the comparative study.
- Including different variants of the models.
- Building hybrid models with more possible combinations.
- Improving the models that performed poorly.

# Chapter 5

# Conclusion

The only way to defend against the threats of mass rumors spread through virtual platforms is to generate methods that can accurately predict rumors. In this paper, we first presented a comparative study among several state-of-the-art ML and DL models using the aforementioned datasets and drew conclusions for which ML and DL to proceed with. From the analysis, we chose Random Forest and RNN for our further experiment. Then we built our Ensemble ML model where we stacked SVM, DT, and NB and used RF as our final classifier. Lastly, we implemented our unique hybrid model for rumor detection combining RF and RNN and it has been able to outdo the rest of the state-of-the-art models that we selected for our experiments.

# Bibliography


[1] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[2] F. Chollet *et al.*, *Keras*, https://keras.io, 2015.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[4] Y. Zhang and B. Wallace, *A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification*, 2015. DOI: 10.48550/ARXIV.1510.03820. [Online]. Available: https://arxiv.org/abs/1510.03820.

[5] H. Ahmed, I. Traore, and S. Saad, "Detection of online fake news using n-gram analysis and machine learning techniques," in *Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, I. Traore, I. Woungang, and A. Awad, Eds., Cham: Springer International Publishing, 2017, pp. 127–138, ISBN: 978-3-319-69155-8. DOI: 10.1007/978-3-319-69155-8_9.

[6] H. Ahmed, I. Traore, and S. Saad, "Detecting opinion spams and fake news using text classification," *SECURITY AND PRIVACY*, vol. 1, no. 1, e9, 2018. DOI: https://doi.org/10.1002/spy2.9. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spy2.9. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spy2.9.

[7] E. Commission, C. Directorate-General for Communications Networks, and Technology, *A multi-dimensional approach to disinformation : report of the independent High level Group on fake news and online disinformation*. Publications Office, 2018. DOI: doi/10.2759/739290.

[8] J. Ma, W. Gao, and K.-F. Wong, "Rumor detection on Twitter with tree-structured recursive neural networks," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 1980–1989. DOI: 10.18653/v1/P18-1184. [Online]. Available: https://aclanthology.org/P18-1184.

[9] H. Patel and P. Prajapati, "Study and analysis of decision tree based classification algorithms," *International Journal of Computer Sciences and Engineering*, vol. 6, pp. 74–78, Oct. 2018. DOI: 10.26438/ijcse/v6i10.7478.

[10] A. Zubiaga, A. Aker, K. Bontcheva, M. Liakata, and R. Procter, "Detection and resolution of rumours in social media: A survey," *ACM Comput. Surv.*, vol. 51, no. 2, Feb. 2018, ISSN: 0360-0300. DOI: 10.1145/3161603. [Online]. Available: https://doi.org/10.1145/3161603.

[11] A. Kumar and S. Sangwan, "Rumor detection using machine learning techniques on social media: Proceedings of icicc 2018, volume 2," in Jan. 2019, pp. 213–221, ISBN: 978-981-13-2353-9. DOI: 10.1007/978-981-13-2354-6_23.

[12] Q. Li, Q. Zhang, L. Si, and Y. Liu, "Rumor detection on social media: Datasets, methods and opportunities," 2019. DOI: 10.48550/ARXIV.1911.07199. [Online]. Available: https://arxiv.org/abs/1911.07199.

[13] M. Ali, *Rumour detection models and tools for social networking sites*, Feb. 2020.

[14] T. Bian, X. Xiao, T. Xu, *et al.*, *Rumor detection on social media with bi-directional graph convolutional networks*, 2020. DOI: 10.48550/ARXIV.2001.06362. [Online]. Available: https://arxiv.org/abs/2001.06362.

[15] M. Z. Hossain, M. A. Rahman, M. S. Islam, and S. Kar, *Banfakenews: A dataset for detecting fake news in bangla*, 2020. DOI: 10.48550/ARXIV.2004.08789. [Online]. Available: https://arxiv.org/abs/2004.08789.

[16] M. G. Hussain, M. R. Hasan, M. Rahman, J. Protim, and S. A. Hasan, *Detection of bangla fake news using mnb and svm classifier*, 2020. DOI: 10.48550/ARXIV.2005.14627. [Online]. Available: https://arxiv.org/abs/2005.14627.

[17] T. Zhu, "Analysis on the applicability of the random forest," *Journal of Physics: Conference Series*, vol. 1607, p. 012 123, Aug. 2020. DOI: 10.1088/1742-6596/1607/1/012123.

[18] R. Anggrainingsih, G. M. Hassan, and A. Datta, *Bert based classification system for detecting rumours on twitter*, 2021. DOI: 10.48550/ARXIV.2109.02975. [Online]. Available: https://arxiv.org/abs/2109.02975.

[19] A. D'ulizia, M. C. Caschera, F. Ferri, and P. Grifoni, "Fake news detection: A survey of evaluation datasets," *PeerJ Computer Science*, vol. 7, 2021.

[20] S. Hangloo and B. Arora, "Fake news detection tools and methods – a review," 2021. DOI: 10.48550/ARXIV.2112.11185. [Online]. Available: https://arxiv.org/abs/2112.11185.

[21] A. Mathew, P. Amudha, and S. Sivakumari, "Deep learning techniques: An overview," in *Advanced Machine Learning Technologies and Applications*, A. E. Hassanien, R. Bhatnagar, and A. Darwish, Eds., Singapore: Springer Singapore, 2021, pp. 599–608, ISBN: 978-981-15-3383-9. DOI: 10.1007/978-981-15-3383-9_54.

[22] M. F. Mridha, A. J. Keya, M. A. Hamid, M. M. Monowar, and M. S. Rahman, "A comprehensive review on fake news detection with deep learning," *IEEE Access*, vol. 9, pp. 156 151–156 170, 2021. DOI: 10.1109/ACCESS.2021.3129329.

[23] P. Patwa, S. Sharma, S. Pykl, *et al.*, "Fighting an infodemic: COVID-19 fake news dataset," in *Combating Online Hostile Posts in Regional Languages during Emergency Situation*, Springer International Publishing, 2021, pp. 21–29. DOI: 10.1007/978-3-030-73696-5_3. [Online]. Available: https://doi.org/10.1007%2F978-3-030-73696-5_3.

[24] M. Rawat and D. Kanojia, *Automated evidence collection for fake news detection*, 2021. DOI: 10.48550/ARXIV.2112.06507. [Online]. Available: https://arxiv.org/abs/2112.06507.

[25] M. Saif, M. K. H. Kanon, N. Hasan, M. S. Hossen, and F. Z. Anannya, *Identification of fake news using machine learning in distributed system*, Jun. 2021. [Online]. Available: http://dspace.bracu.ac.bd/xmlui/handle/10361/15199.

[26] O. Sharif, E. Hossain, and M. M. Hoque, *Combating hostility: Covid-19 fake news and hostile post detection in social media*, 2021. DOI: 10.48550/ARXIV.2101.03291. [Online]. Available: https://arxiv.org/abs/2101.03291.

[27] L. Singh, "Hybrid ensemble for fake news detection: An attempt," in *Data Engineering for Smart Systems*, Springer Singapore, Nov. 2021, pp. 101–108. DOI: 10.1007/978-981-16-2641-8_10. [Online]. Available: https://doi.org/10.1007%2F978-981-16-2641-8_10.

[28] S. D. Sourya Dipta Das Ayan Basak, *Covid19 fake news dataset nlp*, 2021. DOI: 10.34740/KAGGLE/DSV/2016658. [Online]. Available: https://www.kaggle.com/dsv/2016658.

[29] F. Tafannum, M. N. S. Shopnil, A. Salsabil, and N. Ahmed, Sep. 2021. [Online]. Available: http://dspace.bracu.ac.bd/xmlui/handle/10361/15604.

[30] H. de Wet and V. Marivate, *Is it fake? news disinformation detection on south african news websites*, 2021. DOI: 10.48550/ARXIV.2108.02941. [Online]. Available: https://arxiv.org/abs/2108.02941.

[31] Y. Gao, X. Wang, X. He, H. Feng, and Y. Zhang, *Rumor detection with self-supervised learning on texts and social graph*, 2022. DOI: 10.48550/ARXIV.2204.08838. [Online]. Available: https://arxiv.org/abs/2204.08838.

[32] C. Naulak, *A comparative study of naive bayes classifiers with improved technique on text classification*, May 2022. DOI: 10.36227/techrxiv.19918360.v1.

[33] S. Shelke and V. Attar, *Rumor detection in social network based on user, content and lexical features - multimedia tools and applications*, Mar. 2022. [Online]. Available: https://link.springer.com/article/10.1007/s11042-022-12761-y.

[34] L. R. Olsen, *Multiple-k: Picking the number of folds for cross-validation*, Jan. 2023. [Online]. Available: https://cran.r-project.org/web/packages/cvms/vignettes/picking_the_number_of_folds_for_cross-validation.html.