DETECTING MISINFORMATION ON SOCIAL MEDIA: A GNN-BASED ENSEMBLE LEARNING METHODOLOGY

by

ALSON PRASAI

A thesis submitted to the
Department of Computer Science
in conformity with the requirements for
the degree of Master of Science

Bishop's University
Canada
July 2023

Abstract

The proliferation of microblogging platforms such as Twitter and Facebook has revolutionized the way information is disseminated; however, it has also resulted in an alarming rise of fake news and misinformation. In a world where information is readily accessible and misinformation can cause widespread harm, it is imperative to develop an effective method for detecting fake news. This is particularly relevant in light of the recent COVID-19 pandemic, where misinformation on social media resulted in hospitalizations and deaths. In response to this challenge, this thesis proposes a novel solution for detecting fake news by leveraging a combination of Graph Neural Networks (GNNs) and ensemble learning. The GNN model analyzes the complex relationships between articles, users, and topics to generate outputs that are used to train an ensemble of machine learning classifiers. This approach is shown to be more effective than traditional text classification models, as it leverages the graph dataset and yields improved accuracy. The research concludes that the combination of GNNs and ensemble learning is the best approach for detecting fake news on social media platforms.

Acknowledgements

I would like to take this opportunity to express my profound gratitude and appreciation to my supervisor, Dr. Madjid Allili, who has been an invaluable source of support and guidance throughout my thesis journey. His mentorship, knowledge, and direction have been crucial to my career and have left a lasting impact. Additionally, I would like to convey my deepest appreciation to the Computer Science department at Bishop's University, along with its dedicated faculty members, for entrusting me with the opportunity to pursue a Master's degree. The courses I have taken during my time in the department have been immensely helpful in expanding my knowledge and enhancing my understanding of various subject areas, specifically in the context of my thesis research. I am truly grateful for the invaluable experiences and knowledge I have gained during my time in the department.

Lastly, I would like to thank my friends and family, who have been a constant source of unwavering encouragement, support, and inspiration throughout this journey. Their unwavering faith in me and their steadfast belief in my skills have provided me with the strength and motivation needed to overcome challenges and reach this important milestone. I am truly excited to continue my academic path, armed with the invaluable information, talents, and assistance of each and every one of you. From the bottom of my heart, thank you for your unwavering love and support.

Contents

Chapter 1

Introduction

In recent years, millions of individuals have become more connected to social media platforms and now consume more information from there than from traditional sources (e.g., television, national dailies, etc)[1]. The information gained from these sources is not always true, and it tends to be misleading since most people do not fact-check them before believing them wholeheartedly. This causes anxiety and panic to varying degrees[2].

Micro-blogging platforms like Twitter, Facebook, etc. have become popular platforms in modern times for information sharing[1]. The increasing freedom with which users can share whatever they wish and the lack of control over the content on these platforms have further increased the credibility of this information. Although these platforms have facilitated the timely delivery of information across the globe, the damage caused by misinformation on these platforms outweighs their use if not detected early[3]. The needless deaths recorded during the COVID-19 pandemic are a more recent impact of fake news on the public[2]. According to the study published in the American Journal of Tropical Medicine and Hygiene, a large number of individuals were hospitalized as a result of misleading information on social media [5], and the CDC reported deaths arising from the same. To mitigate a recurrence and protect the public, it is extremely important to develop an effective method for detecting fake news.

Recently, there has been a surge in the use of machine learning and deep learning techniques to tackle the issue of fake news detection on social media. Despite numerous efforts made by researchers over the years, the problem still persists. Various studies have been conducted to address the issue of detecting fake news, including those referenced in [6, 7, 8, 9]. In addition, Mahmud et al.[10] conducted a comparative analysis of Graph Neural Networks with commonly used machine learning algorithms using the UPFD[11] datasets. While Graph Neural Networks (GNNs) are designed to process graph data, they may have difficulty capturing long-range dependencies or global information in the graph. The features learned by the model are typically based on the local neighbourhood of each node in the graph, which means that the model only considers the connections between the node and its immediate neighbours when computing the features for that node. This limitation is particularly relevant when the underlying graph is large and complex, such as in a social network. For example, the behaviour of a user may be influenced by the behaviour of other users who are not directly connected to them in the network. To overcome this limitation, an ensemble of machine learning classifiers can be used, which can learn from a richer set of features that include

[1] https://www.forbes.com/sites/nicolemartin1/2018/11/30/how-social-media-has-changed-how-we-consume-news/?sh=38e98fba3c3c

[2] Hundreds dead because of COVID-19 misinformation: https://www.bbc.com/news/world-5375506

both local and global information in the graph, thereby enhancing its overall performance.

To address the problem at hand with efficiency, the GNN-EN framework is proposed. In this context, "EN" denotes the ensemble of machine learning classifiers. This comprehensive framework leverages the output of a Graph Neural Network (GNN) model to train multiple machine learning classifiers. This approach results in a more diverse set of predictions, which allows the system to leverage the strengths of multiple models and generate more accurate and reliable predictions for detecting fake news. By using ensemble learning, it helps to mitigate the potential biases and limitations of individual models. Additionally, it can improve the generalization performance of the model and reduce the risk of overfitting, which is especially important when training complex models on limited data.

1.1 Objective

This thesis aims to develop an ensemble framework that is based on GNN to extract feature sets and a group of classifiers to boost the model's predictability. The following strategy will be used to achieve this goal:

1. Collect real-world graph datasets rich in features and information that will be used to train, test, and validate fake news.
2. Use GNNs to convert high-dimensional graph data to low-dimensional numeric features that can be used to train machine learning models using a voting-based ensemble approach.
3. Compare the accuracy of the model before and after using ensemble learning.
4. Analyze the results of different word embedding algorithms to see how well each one can detect fake news.

1.2 Thesis Structure

The study begins by reviewing the related works in Chapter 2, where various approaches used by previous researchers to tackle this problem are discussed. In Chapter 3, essential background knowledge is presented to provide a better understanding of the proposed theory and experiments. This includes classification algorithms, ensemble learning, graph neural networks, and word embedding techniques. Chapter 4 outlines the justification of ensemble learning and the methodology used in the study. Chapter 5 provides the data source, data processing, and experimental setup for the research. The results and analysis of the experiment are presented in Chapter 6, demonstrating the effectiveness of the proposed solution. Finally, Chapter 7 concludes the thesis by reviewing the findings with respect to the initial goals and discussing potential future directions for further research in this field.

Chapter 2

Related Works

In an effort to resolve the problem of fake news, different approaches have been proposed by various researchers. These approaches can be categorized into three main groups based on their content, social context, and propagation [12, 13].

2.1 Content-Based Approach

This approach relies on linguistic (lexical and syntactical) features that can capture deceptive cues or writing styles such as special characters, emotions, symbols, sentiments, positive or negative words, and hashtags [14, 15, 16]. The authors of [14] discusses satire and humor conceptually, by explaining and illustrating the unique features of satirical news, which mimics journalistic style and format. They used the SVM-based algorithm, testing their combinations on over 360 news articles. The use of stylistic cues to determine the truthfulness of a text is discussed in a study [15], which compares the language of real news to that of satire, hoaxes, and propaganda. The authors concluded that stylistic cues could help determine the truthfulness of a text. Chen et al. [16] improved on this approach by combining attention mechanisms with RNN to focus on text features with different attention. Similarly, a retrospective analysis of a 2009 Twitter dataset [17] found that content-based features, particularly word frequencies, play a key role in rumour detection as opposed to user-based features.

Despite the advances recorded by various researchers using the content-based approach, drawbacks persist. One of the major drawbacks recorded is that they can be defied by sufficiently sophisticated fake news that does not immediately appear fake. Majority of existing detection algorithms are generally ineffective since fake news is frequently designed to trick readers by replicating legitimate news through linguistic cues [18]. Also, most linguistic features are language dependent, limiting the generality of these approaches.

2.2 Social Context-Based Approach

This approach deals more with user demographics (age, gender, education) [19, 20, 21] and user reactions (likes, retweeting behaviour) [22, 23]. The authors of [19] studied real-world datasets that assess trust levels in fake news and selected representative groups, including ”experienced” users who are capable of recognizing false information and ”naive” users who believe fake news items. They performed a comparative analysis of explicit and implicit profile features between these user

groups, revelaing their potential to differentiate fake news. A novel method to incorporate speakers' profiles into an attention-based LSTM model for fake news detection is proposed in [20]. Based on the speaker profiles, a high level of accuracy was achieved, and it was concluded that the speaker profiles are helpful in validating the credibility of news articles.

Though this approach tends to be more effective at fake news detection than the content-based approach, it is also fraught with significant limitations on efficiency. Owing to the complex structure of social media data where users are connected to each other and where social relations play an important role in knowing the overall characteristic pattern of fake news dispersion, these tasks require dealing with non-Euclidian graph data that contains rich relational information between users. This cannot, therefore, be handled by traditional machine learning and deep learning models. Thus, there is a need for the graph neural network to be employed.

2.3 Propagation-Based Approach

These are one of the most promising and intriguing lines of research based on studying the news expansion over time. Researchers agree on the analogy between fake news dissemination and the spread of infectious diseases [24]. Network epidemic models best describe this process.

Substantially, empirical evidence shows that fake news propagates differently from true news [25], creating spreading patterns that could be exploited for fake news detection. In contrast to content-based features, which must be built separately for each language, propagation-based features are likely to work across languages, locales, and geographic areas because they are based on content empiricism [26]. Furthermore, propogation based features would potentially be very difficult to temper with by foe attacks since controlling the news spread patterns in a social network is generally beyond the capability of individual users.

The GCNFN [26] framework was one of the earliest attempts to detect fake news using news propagation graphs encoded with GCN. It used profile information and comment embedding as the user features. It, however, focused on modelling news content and its user exogenous context (all users engaged with the news propagation path), while ignoring the user endogenous preference (historical tweets).

UPFD [11] addresses feature-rich information by considering both user exogeneous and endogeneous preferences for detecting fake news. Its result was compared with the baseline GCNFN. The results demonstrate that the user's endogenous preference imposes more information when user comment information is limited. The UPFD was shown to outperform the best baseline GCNFN by 1%. The work showed that BERT as the text encoder and GraphSage as the graph encoder produced the best performance on the tested datasets.

Researchers in [10] performed a comparative analysis of several machine learning and graph neural networks using both news content and graph data from the UPFD [4]. Their results concluded that GNN is the most effective approach to detect fake news.

Chapter 3

Background

This chapter provides a comprehensive background on the necessary information to fully understand the research presented in this thesis. Section 3.1 introduces machine learning and its most widely used classification algorithms. In Section 3.2, the specific machine learning classifiers used in this study, in combination with Graph Neural Networks (GNNs), are discussed. Section 3.3 delves into ensemble learning strategies, providing an overview of the techniques. Sections 3.4 and 3.5, provide in-depth descriptions of Artificial Neural Networks (ANNs) and GNNs, respectively, including the fundamental concepts and their applications. Finally, Section 3.6 covers the important topic of word embedding, where the strategy for representing words as vectors is described in detail.

3.1 Machine Learning

Machine learning is the study of computer algorithms that make use of data and gradually improve accuracy via experience without any human intervention [27]. In recent years, due to the huge volumes and varieties of data, affordable processing power, and high-speed Internet, machine learning has gained relevance in our daily lives, such as in Google search, recommendation systems, and facial recognition.

Machine learning systems can be classified depending on the supervision they receive during training [28].There are primarily three types of machine learning that are discussed below:

3.1.1 Supervised Learning

A supervised learning technique predicts unseen data based on labelled training data. One common topic in supervised learning is classification. For example, when implementing a spam filter, they are trained on many examples with labelled classes (spam or ham), and from the training set, they must learn to classify the new unseen emails. Another common task is regression, which is used to predict a numeric value, like the price of a car, based on a set of features, like the car's mileage, age, brand, etc [28]. The commonly used supervised learning algorithms are Logistic Regression, Decision Trees, Support Vector Machines, K-Nearest Neighbors, and Artificial Neural Networks.

3.1.2 Unsupervised Learning

Unsupervised learning analyzes and clusters unlabeled datasets using machine learning methods. These algorithms uncover hidden patterns or data clusters without requiring human intervention [1]. This makes it an ideal choice for tasks like exploratory data analysis and customer segmentation. K-means clustering, Principal Component Analysis (PCA), and autoencoders are common examples of unsupervised learning.

3.1.3 Reinforcement Learning

Reinforcement is a learning process that works based on rewards. In this case, the learning system, called an agent, can observe the environment, choose and carry out actions, and get rewards in return [28]. Unlike supervised learning, it does not need labelled input-output pairs to perform the action. Some examples of reinforcement learning are the Markov Decision Process, Q-learning, and Temporal Difference Learning.

3.2 Machine Learning Classifiers

In machine learning, a classifier is an algorithm that automatically categorizes data into one or more classes. This section gives a brief summary of the machine learning classifiers used in this study.

3.2.1 Logistic Regression

Logistic regression,also commonly referred to as a binomial logistic regression model, is a supervised learning technique that can predict the probability that an observation falls into one of two categories based on a given set of independent variables. The result of this statistical model is a probability, and the value of the dependent variable is between 0 and 1. In logistic regression, the linear function is used as an input to another function, such as σ [29]:

$$f_\beta(x) = \sigma(\beta^T x) \;\; where \;\; 0 \leq f_\beta \leq 1 \tag{3.1}$$

Here, σ is a sigmoid or logistic function and is represented by given formula:

$$\sigma = \frac{1}{e^{-z}} \;\; where \;\; z = \beta^T x \tag{3.2}$$

For binary classification, cross-entropy is commonly used as a cost function. Cross entropy increases when the predicted probability diverges from the true label. In order to get the optimal result, we use gradient descent to update the coefficients and minimize the cost function. It is given by the formula:

$$g(h_\beta(x), y) = -y.\log(h_\beta(x)) - (1-y)\log(1-h_\beta(x)) \tag{3.3}$$

where, $g(h\beta(x), y)$ represents the cost function that measures the dissimilarity between the predicted value $h_\beta(x)$ and the true label y.

Logistic regression is easier to implement and has better fitting flexibility since different regularization techniques like lasso and ridge can be used to reduce the error in the model. However, they tend to overfit if the number of observations is less than the number of features. This work selects feature-rich information with a sufficient number of samples to overcome this issue.

[1] https://www.ibm.com/topics/unsupervised-learning

3.2.2 Decision Trees

A decision tree is a supervised learning algorithm that can be used to solve both classification and regression problems. It is organized in a hierarchical tree fashion, which consists of a root node, branches, internal nodes, and leaf nodes.

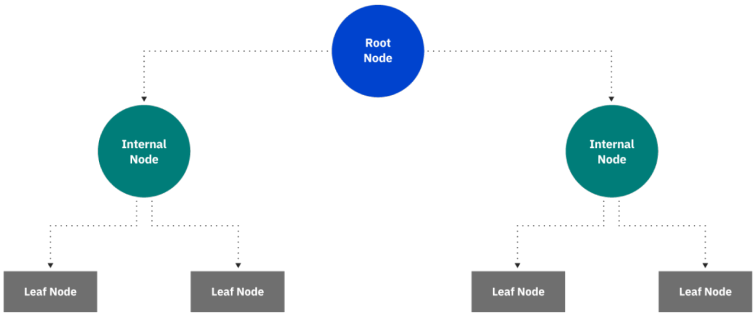

Figure 3.1: A representation of a decision tree [30].

The root node of a decision tree is the topmost node that has no incoming branches. The internal nodes, also called decision nodes, receive outgoing branches from the root node. Both of these nodes conduct evaluations on available feature sets to form leaf nodes (terminal nodes) that do not further split into more nodes. There are a number of approaches for selecting the best attribute at each node. The most common splitting criteria for decision tree models are information gain and Gini impurity. Both entropy and Gini impurity are metrics to measure the impurity of a node.

Entropy

The entropy E(S) for a dataset consisting of an N-class number of classes is calculated by the below formula:

$$E = \sum_{i=1}^{\mathbb{N}} -p_i \log_2(p_i) \tag{3.4}$$

where p_i represents the probability of class i occurring within the set or node. These probabilities are calculated by dividing the number of occurrences of class i by the total number of instances in the set or node.

Entropy values can fall, ranging from 0 to 1. A dataset, S, with all samples belonging to one class has zero entropy, whereas a dataset with half of the samples classified as one class and half as another, will have the value at its peak at 1.

Gini Impurity

Gini impurity, also called the Gini index, measures the frequency of mislabeled elements in a given dataset. The value of gini impurity ranges from 0 to 0.5. Here, the value zero refers to the pure

node, where all elements in the node belong to one single class, and 0.5 refers to the impure node, with elements in the node belonging to multiple classes. The optimal split is the one having with the lowest of the Gini index. The Gini index is calculated using the formula given below.

$$GiniIndex = 1 - \sum_{j=1}^{\mathbb{N}} p_j^2 \tag{3.5}$$

where p denotes the probability that a training instance belongs to a specific class and N is the upper limit of the summation, representing the total number of classes.

Information Gain

In order to find the optimal decision tree with the best feature split, the attribute with the least amount of entropy should be used. This can be obtained via the method of information gain. Information gain is the difference in entropy before and after a split of an attribute. The attribute with the highest information gain yields the best split to perform the classification task. Information Gain (I.G.) is calculated using the formula given below:

$$\text{I.G} = \text{Entropy}(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} \cdot \text{Entropy}(S_v) \tag{3.6}$$

where Entropy(S) represents the impurity of the original dataset. The specific impurity measure used can be entropy or Gini impurity. v represents the possible values that the feature A can take, and $|S_v|$ represents the number of instances in the subset S_v.

Compared to other algorithms, decision trees are easy to understand and do not require normalization or scaling of data. However, one of the difficulties with using decision trees is that they are sensitive to small variations. So ensembles should be used with them. This study uses decision trees in order to eliminate the sensitivity to variation associated with ensemble learning.

3.2.3 Support Vector Machine

The Support Vector Classification is a supervised-learned algorithm. The primary goal of this classifier is to locate a hyperplane in an N-dimensional space that can classify the data points even when they are not linearly separable. It accomplishes this by locating the boundary that divides the categories and transforming the data in such a way that the separator can be drawn as a hyperplane. The transformation is carried out by the kernel function. In many cases, linear kernel functions are useful for solving problems with linear separations. In other cases, polynomial, sigmoid, and radial basis functions can be used instead.

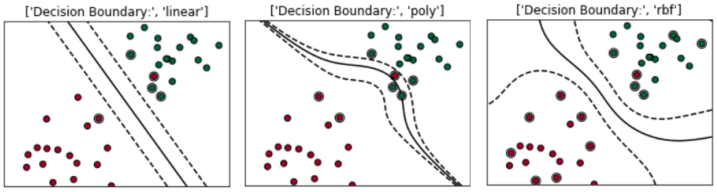

Figure 3.2: A representation of the SVM kernels' [31].

SVM is effective in high-dimensional space and also works well when the number of features is greater than the number of samples. One major drawback of using SVMs is that they employ 5-fold cross validation by default, and thus have a tendency to overfit if there are a significant number of unwanted features in the dataset. In this work, adequate feature selection and extraction are performed to mitigate this problem.

3.2.4 K-nearest neighbors

KNN is a non-parametric algorithm that can be used to solve classification and regression problems. It is called non-parametric because it does not make any assumptions about data distribution.

In KNN, k refers to the number of close or nearest neighbors. For classification problems, the number of neighbours (k) is a core deciding factor when predicting a class label. In the case of binary classification problems, the value of k is generally an odd number. For instance, if P1 is a point of interest for predicting the class label, a prediction is made by combining the votes of the k nearest points to P1. This is done by finding the k closest points to P1 and then classifying them by the majority vote of their k neighbors. In order to determine the closest similar points for P1, distance metrics like Euclidean distance, Hamming distance, Manhattan distance, and Minkowski distance are used.

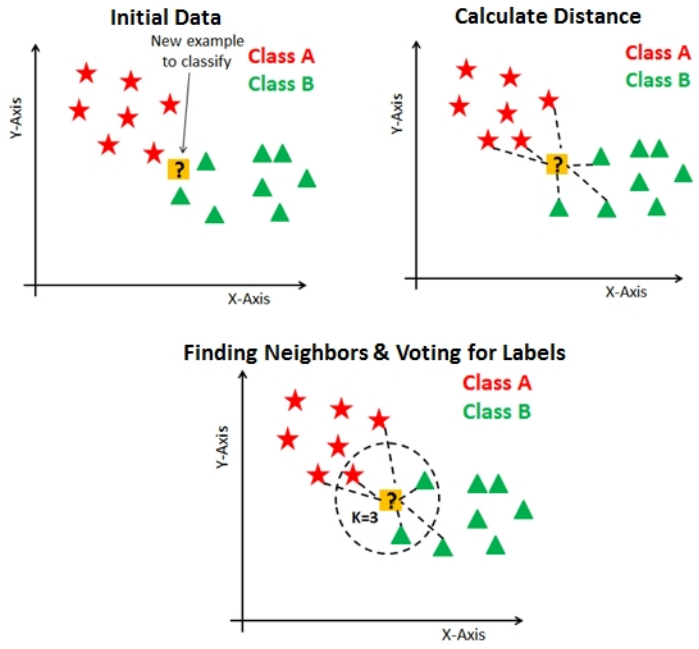

Figure 3.3: An illustration of the working mechanism of KNN [32].

KNN is simple to construct since it requires only two parameters (distance metric and the value of k). It is significantly quicker than other algorithms such as logistic regression and SVM since it only requires learning during the prediction phase, without a traditional training period. Therefore, it is also known as a lazy learner. However, they are sensitive to noisy data, missing values, and outliers. It also performs poorly on unbalanced datasets since it gives preference to the class with the majority of training labels. To minimize these limitations, proper feature selection and a data set with a balanced class label are used.

3.3 Ensemble Learning

In the field of machine learning, the term ”ensemble machine learning” refers to a method that makes a prediction by combining the results of several machine learning models, also known as base learners. The main objective of ensemble learning is to combine several base learners to obtain a single predictive model, which is likely to increase predictability by decreasing variance. Different types of ensemble learning techniques are discussed below.

3.3.1 Voting Ensemble

Max-voting is a technique that is typically used for classification problems. In this method, a number of base learners each make a prediction for every data point. The prediction of each of these data points is counted as a vote for that model. The final prediction is the one that is chosen from the majority of the models.

Furthermore, voting classifer implements two types of voting mechanisms - hard and soft voting. Hard voting takes the mode of the predicted class label as the final prediction [29]. Soft voting, on the other hand, uses the average probabilities from each base classifier instead of a binary input.

3.3.2 Bagging

"Bagging classifiers train each classifier model on a random subset of the original training set and aggregate the predictions, then perform a plurality vote for a categorical outcome" [29]. Bagging works especially well when the learners are unstable and tend to overfit, i.e., small changes in the training data lead to major changes in the predicted output. Mathematically, bagging is represented by the following formula:

$$f_{\text{bag}} = f_1(x) + f_2(x) + \ldots + f_n(x) \tag{3.7}$$

where:

- f_{bag} represents the combined prediction obtained through bagging.
- $f_1(x), f_2(x), \ldots, f_n(x)$ indicate the individual predictions of the models trained in the ensemble. Each model, denoted by the subscript i (ranging from 1 to n), makes its own prediction for a given input or instance x.
- x represents the input or instance for which the prediction is being made.

Random Forest

A random forest is an ensemble learning-based supervised machine learning algorithm used for both regression and classification. The basic idea of a random forest involves building many decision trees and aggregating them to produce an accurate result. A decision tree is a deterministic algorithm, meaning that if the same data is supplied to it, it will always return the same tree [29]. This makes it likely that the model will overfit the data, meaning it will perform well with the observed data but fail to generalize to unseen data. To solve this overfitting issue, each of the decision trees forming a random forest is made unique by using a distinct random subset of data. This results in a more accurate model than a single decision tree. For classification problems, the majority vote of the base learners in a random forest is used to predict class labels.

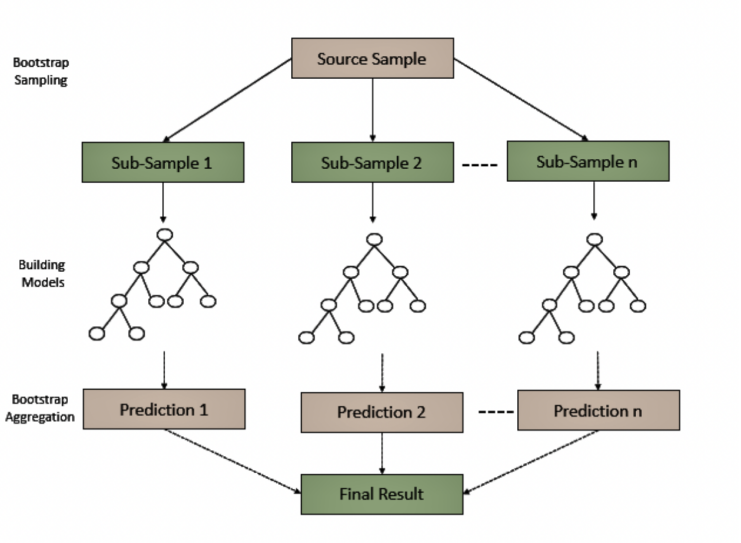

Figure 3.4: An illustration of bootstrap sampling in random forest [29].

3.3.3 Boosting

Boosting is an ensemble-based learning algorithm that enhances the performance of a model by combining a group of weak learners into a single strong learner. The main idea underlying boosting is that predictors should learn from the errors of their predecessors. This can be summarized in two key steps:

1. First, they undergo multiple iterations and,
2. Second, each iteration emphasizes on occurence that were incorrectly classified in earlier iterations.

AdaBoost

AdaBoost is one of the first binary classification boosting methods [29]. It focuses on aggregating multiple weak learners into a single powerful learner. It is represented by the following formula:

$$F(x) = w_1 f_1(x) + w_2 f_2(x) + ... + w_n f_n(x) \tag{3.8}$$

$$F(x) = w_1 f_1(x) + w_2 f_2(x) + \ldots + w_n f_n(x) \tag{3.9}$$

where:

- $F(x)$ represents the final prediction obtained through AdaBoost.
- $f_1(x), f_2(x), \ldots, f_n(x)$ are individual weak classifiers.
- $w_1, w_2, \ldots, w_n$ are the corresponding weights assigned to each weak classifier.

The following steps make up the procedure for an AdaBoost classifier:

1. First, a base classifier such as a decision tree is fitted to the data, and overall errors are calculated. This is the very first iteration of the process.
2. The second iteration involves assigning lesser weights to correctly classified data and higher weights to misclassified classes.
3. The third iteration involves fitting a new classifier (decision tree) to the data and updating the weights again in the next iteration.
4. Finally, a strong classifier is formed by automatically calculating the weights for each classifier based on the error rates at each iteration.

These steps are depicted in the diagram below:

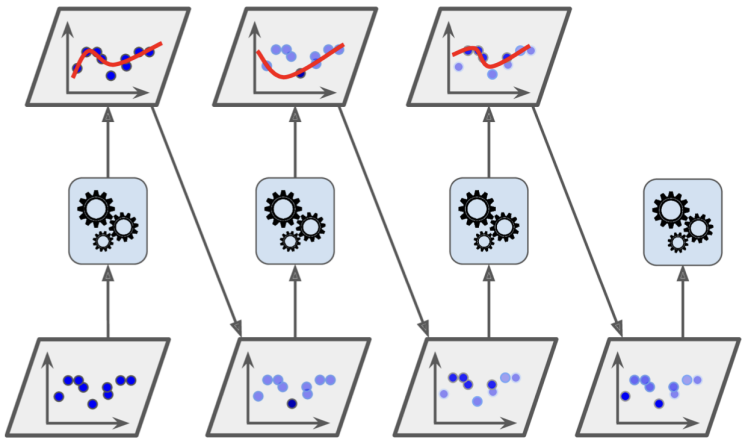

Figure 3.5: An illustration of AdaBoost sequential training [33].

Gradient Boost

Gradient boosting is a machine learning technique based on the principle of boosting in which weak learners (usually decision trees) are trained to improve their performance by focusing on incorrect observations that were difficult to predict in prior iterations to form an ensemble of weak learners [29]. Unlike AdaBoost, which uses weight to identify errors, gradient boosting calculates gradients in the loss function to identify the errors. It trains models sequentially by executing the following steps [29]:

1. Fit a model to the data
2. Fit a model to the residuals
3. Create a new model.

3.4 Artificial Neural Network

Artificial Neural Networks (ANN) are machine learning techniques that are very similar to biological neurons. It consists of various computational nodes and is divided into multiple layers. They are composed of three types of layers:

1. Input Layer: This is the first layer into which data is fed. This layer sends data to the next layer without performing any computation.
2. Hidden Layer: This layer is found between the input and output layers. It takes the output from the input layer,weights these inputs, and passes them through the activation function. These layers are necessary to perform the non-linear transformation of the inputs received in the network. There can be multiple hidden layers inside the network.
3. Output Layer: The final layer is the output layer, where the desired predictions are made.

A computational node, also called a neuron, performs a linear transformation by adding a bias to the weighted sum of its inputs. This sum is then passed through an activation function to make a nonlinear transformation. An artificial neuron calculates the weighted sum of its inputs, adds a bias, and makes this nonlinear with the help of an activation function. The output of a neuron is computed using the following formula:

$$O_i = f\left(\sum_{i=1}^{\mathbb{N}} \alpha_{ij} x_i + b_j\right) \tag{3.10}$$

where:

- O_i represents the output or activation of a neuron in the ANN.
- $f(\cdot)$ denotes the activation function applied to the sum of weighted inputs and bias.
- $\sum_{i=1}^{\mathbb{N}}$ indicates the summation operation over i ranging from 1 to $\mathbb{N}$.
- α_{ij} represents the weight connecting the input x_i to the neuron. Each neuron has its own set of weights, denoted by the subscript j.
- x_i represents the input value from a previous layer or the input layer of the network.
- b_j represents the bias term associated with the neuron. Each neuron has its own bias, denoted by the subscript j.

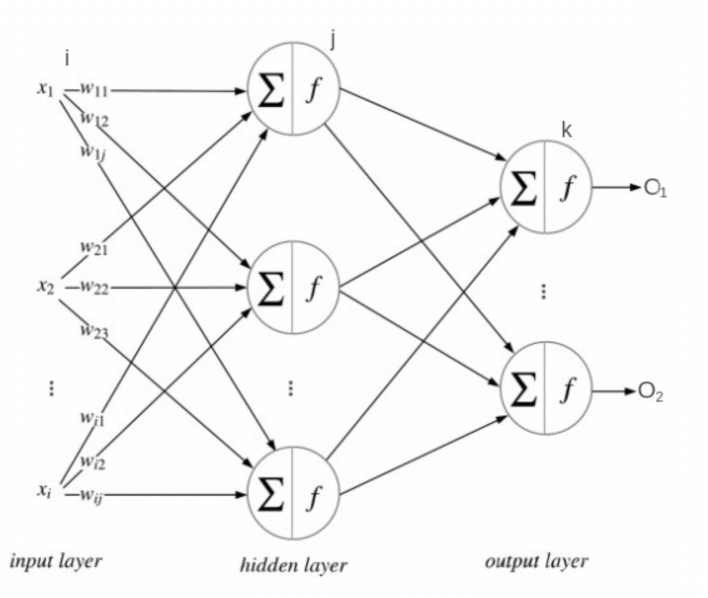

Figure 3.6: An illustration of a neural network with multiple layers [34].

In a neural network, the process of receiving an input and producing an output is repeated until the final layer (the output layer) is reached. Since this occurs sequentially, it is also called a feed-forward network. The actual learning takes place through the backpropagation method, which is discussed below.

Backpropagation

The learnable parameters, like weight and bias, are updated through the backpropagation technique. Generally, random values are assigned to the weight and bias and are iteratively updated during the training phase. The goal is to find the values of the trainable parameters with the lowest loss. This loss is calculated via a loss function, which measures the deviation of the predicted output from the expected output. The loss function, cross entropy, is commonly used for classification tasks.

Given a loss function, the weights are updated using the chain rule by taking the partial derivative of the loss function with respect to the parameter that needs to be updated. This can be represented with the following formula:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial x} \cdot \frac{\partial x}{\partial z} \cdot \frac{\partial z}{\partial w} \tag{3.11}$$

where:

- $\frac{\partial L}{\partial w}$ represents the partial derivative of the loss function L with respect to the weight w. It quantifies how a small change in weight affects the loss.
- $\frac{\partial L}{\partial x}$ denotes the partial derivative of the loss function L with respect to the output x. It measures the sensitivity of the loss to changes in the output.

- $\frac{\partial x}{\partial z}$ indicates the partial derivative of the output x with respect to the weighted sum z. It captures how small changes in the weighted sum influence the output.
- $\frac{\partial z}{\partial w}$ represents the partial derivative of the weighted sum z with respect to the weight w. It measures the sensitivity of the weighted sum to changes in weight.

Similarly, weights are updated by:

$$w = w - \alpha \frac{\partial L}{\partial w} \tag{3.12}$$

where α is a learning rate.

Neural networks are capable of self-learning and producing output that is not limited by the input they receive. They can store the information of an entire network. However, the model requires fine-tuning of a number of hyperparameters in order to function efficiently, and inappropriate scaling of features might negatively impact its performance. Overfitting is another common problem in neural networks, where the model learns the details of training data so well that it fails to generalize to unseen data.

To overcome this problem, effective parameter adjustment and careful feature selection are performed. Similarly, the majority of the data (80%) is used in the training set, where the learnable parameters are updated. 10% of the data is used for a validation set to train base classifiers and see if the model is overfitting. Finally, 10% of the data is used as a test set to evaluate our model for the unseen data.

3.5 Graph Neural Network

Graphs are a kind of data structure that models a set of objects (nodes) and their relationships (edges) [35]. Depending on the directional dependencies between vertices (nodes), edges may be directed or undirected. For example, nodes can represent people and their attributes (e.g., name, gender, age) in a social network, whereas edges indicate the relationships between users. A graph is represented as:

$$G = (V, E) \tag{3.13}$$

where **G** is a graph, **V** and **E** are the vertices (nodes) and edges of the graph respectively.

"Graph neural networks (GNNs) are neural models that capture the dependence of graphs via message passing between the nodes of the graph" [35]. It is a form of neural network that works based on the structure of a graph and facilitates node-level, edge-level, and graph-level prediction tasks. These graph analytics levels and prediction tasks are discussed below:

1. Node-level tasks involve predicting the identity or role of each node in a graph by extracting the high-level node representation via information propagation. For example, detecting malicious actors online.
2. Edge-level deals with edge classification and link prediction tasks. A similarity function can be used to predict the label or strength of a connection between two edges by using the hidden representations of the two nodes in GNN. For example, recommendation systems.
3. Graph-level involves graph classification tasks. GNN, when combined with pooling and readout operations, provides a compact representation at the graph level. e.g., predicting chemical properties of molecular graphs, detecting fake news, etc.

The main goal of GNN is to create a new representation for each node, known as node embeddings, by utilizing all the information about the graph, i.e., the nodes' features and their connections. These node embeddings are low-dimensional vectors that describe nodes' positions in the graph as well as the structure of their local graph neighbourhood. This final embedding is then used to classify the nodes. To accomplish this, GNNs rely on a message-passing framework that acts as their underlying working principle.

3.5.1 Message Passing Framework

During each iteration of message-passing, the embedding $h_u^{(n)}$ associated with each node $u \in v$ is updated based on infromation accumulated from u's graph neighbourhood information $N(v)$ (Fig 3.8) [36]. This can be expressed as follows:

$$\begin{aligned} h_u^{n+1} &= UPDATE^{(n)}(h_u^{(n)}, AGGREGATE^{(n)}(h_v^{(n)}, \forall v \in N(u)) \\ &= UPDATE^{(n)}(h_u^{(n)}, m_{N(u)}^{(k)}) \end{aligned} \tag{3.14}$$

where, **mN(u)** is a message aggregated from u's neighbourhood **N(u)**. The functions **UPDATE** and **AGGREGATE** are arbitrary, differentiable functions. Supscripts are used in the above equation to identify the embeddings and functions at different iterations of message-passing. [36]

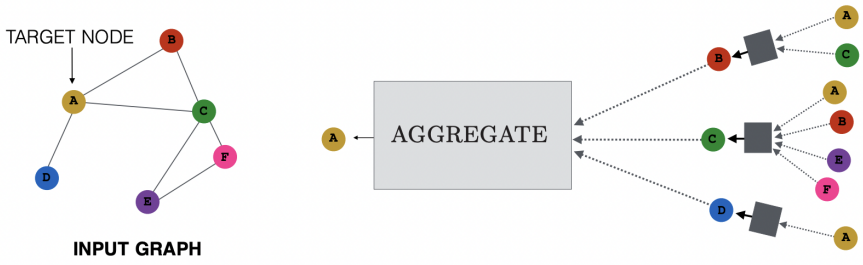

Figure 3.7: An overview of the two-layer message passing model that shows how nodes aggregate neighbourhood messages. The model collects messages from A's immediate neighbours (B, C, and D), and their messages are based on information gathered from their neighbourhoods. [36].

In summary, during message passing, each node aggregates information from its adjacent nodes at each iteration. The steps below explain the GNN's working principles:

1. Following the initial iteration (n = 1) , each node embedding holds the information from its 1-hop neighbors (immediate graph neighbors).
2. In the second iteration (n = 2), each node embedding holds the information from its 2-hop neighbours (i.e., nodes that are rechable on path 2).
3. With each of these iterations, every node becomes more detailed, containing information about its n-hop neighbourhood.
4. Finally, a graph embedding can be obtained by aggregating all the node information present in the graph.

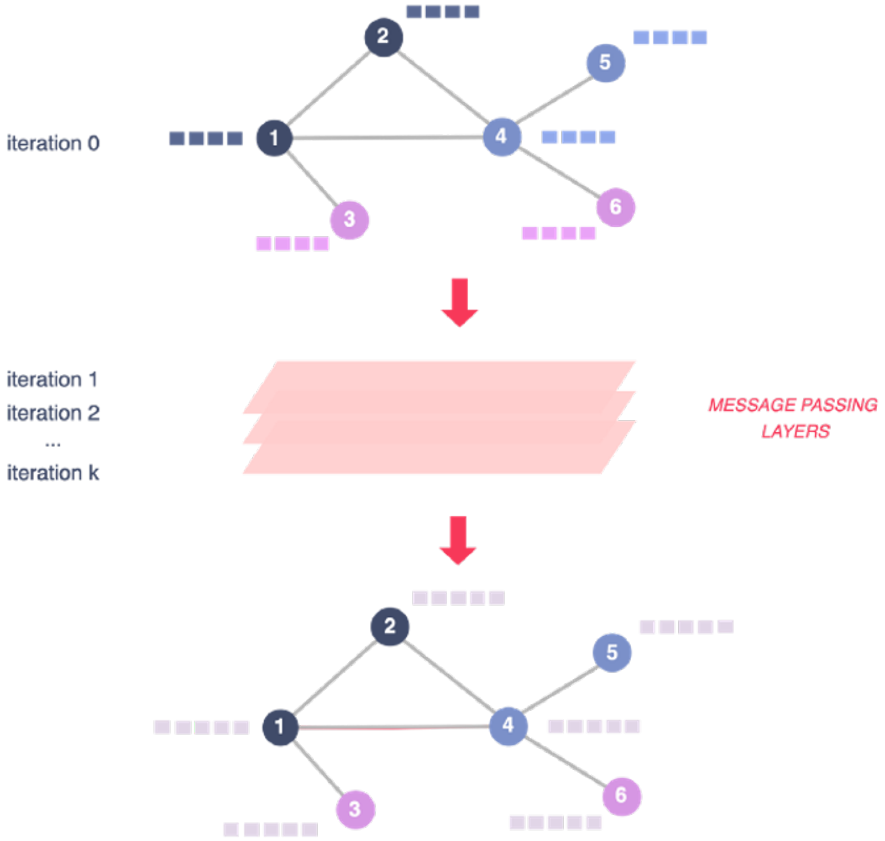

Figure 3.8: A representation of the SVM kernels' [31].

Graph Pooling

The goal of neural message passing is to construct a set of node embeddings, but what if we wish to learn graph representations? To learn an embedding for an entire graph, we pool these node embeddings together, a process referred to as graph pooling. The main objective of graph pooling is to minimize the number of nodes in a network while keeping the graph's semantic information. It can be represented as:

$$G^{'} = POOL(G) \tag{3.15}$$

where, POOL is a pooling function that transforms graph G into a new pooled graph $G^{'}$ such that $G^{'} = (V^{'}, E^{'})$ and $|V^{'}| < |V|$.

Graph pooling could be roughly divided into global graph pooling and hierarchical pooling according to their roles in graph-level representation learning. Once the input has been passed through the GNN layers, global graph pooling transforms the node representations into a single vector as a graph representation. This graph representation is then used by a classifier to accomplish graph classification. The most common examples of global graph pooling are averaging and summation [37]. Similarly, in the case of hierarchical pooling, it gradually reduces the size of the graph to produce

the final graph embedding. The final output of a hierarchical process produces fewer nodes than the input graph. Some common examples of hierarchical graph pooling methods are DIFFPOOL, SORTPOOL, etc [37].

Although the GNN's nodes contain rich information with each iteration of message passing, it comes with a major disadvantage, i.e., oversmoothing. Multiple iterations of message passing can result in identical representations for all the nodes in the graph. To overcome this issue, we can either maintain a low number of layers or use skip connections. With skip connections, layers can jump over layers and connect to layers higher up in the network. This makes it easier for information to move up the network.

3.6 Word Embedding

From the DMO's Data Never Sleeps 5.0 report [2], over 90 percent of the world's data was generated within the last two years. Every minute, Twitter is agog with over 460,000 messages, with Facebook having more than 510,000 comments and 293,000 status updates. Different articles are searched on Google every day by millions of users, providing a few keywords and getting hundreds of results in less than a second. Nate Silver, in his work, analyzed millions of tweets to correctly predict election results. Sentences are typed into Google and translated into various languages. These tasksâclustering, classification, and translationârequire text processing. However, machines do not understand text and therefore require some kind of numerical representation that computers can understand and process.

Mapping a word to an integer could be imagined as a solution. Each word gets an arbitrary integer, and the vocabulary expands as a new word in the assemblage. This is as good as comparing two words as identical [17]. But two words having close meanings might be assigned distant integers, and two adjacent words might be assigned nothing in common. Thus, there is no relationship between the words, and information cannot be easily shared across words with similar properties. This representation of words as unique and distinct leads to sparse data. Overcoming this drawback requires word representation that captures their meanings, semantic representation, and the different types of contexts they are used in. This section discusses two widely used word embeddings in the field of Natural Language Processing (NLP): word2vec and BERT.

3.6.1 Word2Vec

Word2vec is a neural network-based framework that is trained on an excessively large corpus of data to produce word embeddings. The main objective of the word2vec model is to produce a low-dimensional numerical word representation based on their occurrences in the text. For example, the phrases "King" and "Queen" are fairly similar and commonly appear together within the same text. Thus, when performing algebraic operations on word embeddings, it is possible to get an estimate of word similarity. This embedding is generated either using the Continuous Bag of Words (CBOW) or the skip-gram model. The CBOW model and Skip-gram model are predictive word representations based on the idea that neighbouring words encode the meaning of a word vector. Both of these approaches are discussed below.

[2]Data Never Sleeps 5.0: https://www.domo.com/learn/infographic/data-never-sleeps-5

Continuous Bag-Of-Words (CBOW) Model

The Continuous Bag of Words (CBOW) is one of Mikolvâs two model architectures presented in [38]. It uses a probabilistic approach to learn word representations using feedforward neural networks and predicts the target word given the word context by minimizing the given loss function.

$$L = -\log(p(w_t|S_t) \tag{3.16}$$

where, w_t and S_t represent the target word and sequence of words in the context, respectively.

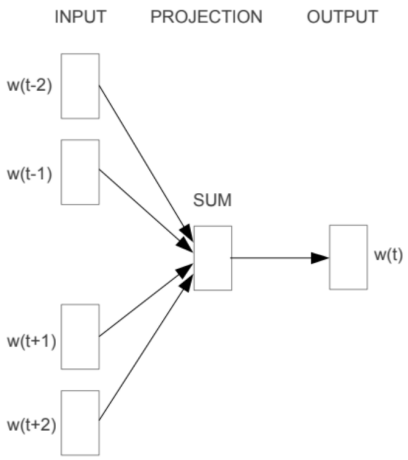

Figure 3.9: A CBOW architecture that predicts target word given context of words [38].

Skip-Gram Model

Skip-Gram is similar to CBOW, but it is different in that instead of predicting the target words, it predicts the context words around the target word within a certain radius. In each step, one target word is selected, and we try to predict its context within a radius of c. To optimize the probability of predicting w(c,j) in the cth context, we try to minimize the negative log probability by iteratively updating the θ, which is given by

$$\begin{aligned} L(\theta) &= -\log P(w_{c,1}, w_{c,2}, ..., w_{c,n}|w_o) \\ &= -log \prod_{i=1}^{n} P(w_{c,i}|w_o) \end{aligned} \tag{3.17}$$

We define the probability function using softmax as:

$$p(w_{c,i}|w_I) = \frac{\exp u_{c,i}}{\sum_{i'=1}^{n} \exp u_{i'}} \tag{3.18}$$

where, $w(c,i)$ is the i^{th} word predicted on the c^{th} context position, $w(I)$ is the only input word; and $u(c,i)$ represents the i^{th} value of the vector U when predicting the word for the c^{th} context.

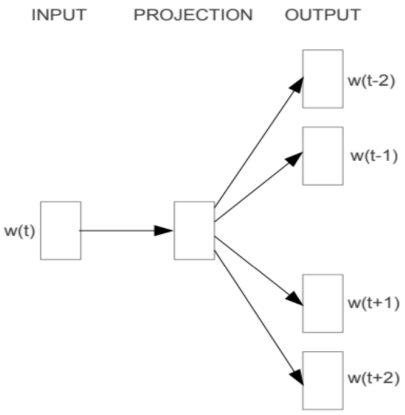

Figure 3.10: A Skip-gram Architecture that predicts context words given the current word [38].

3.6.2 BERT

BERT [39] stands for Bidirectional Encoder Representations from Transformers. It generates a language model by using the encoder part of Transformer [40] which is an attention mechanism to learn the contextual relationship between words in a text. The model was pre-trained using a large corpus from Wikipedia. The traditional language models were only capable of reading left-to-right or right-to-left text sequentially, but not simultaneously. As a solution, BERT was designed to read both directions, i.e., the transformer encodes the entire sequence of words at once.

Word2Vec is context-independent, meaning it generates a word embedding that is just one numeric vector for every word. In the case of words with multiple meanings, they are combined into one single vector representation. However, BERT produces word embeddings with multiple numeric vector representations of the same word based on the context in which the word is used. The main advantage of BERT is the ability to generate context-aware word embeddings over traditional embeddings.

In this work, both word2vec and BERT are used to generate the word embeddings from the text and are compared for their effectiveness.

Chapter 4

Methodology

This chapter discusses the methodologies employed to conduct the experiments, presenting a thorough exploration of the research methodologies and techniques utilized. The results obtained from these experiments are subsequently presented and analyzed in the following chapters, offering valuable insights into the performance of the models and the effectiveness of the ensemble approach adopted.

4.1 Method-Objective Alignment

The related work [Chapter 2] on fake news detection techniques has revealed that different approaches have been employed, including content-based, social context-based, and propagation-based methods. Each of these methods has its own strengths and limitations in addressing the problem of fake news detection. Content-based approaches, for instance, can be deceived by sophisticated fake news that imitates legitimate news through linguistic cues. Social context-based approaches, on the other hand, struggle with the complex structure of social media data and non-Euclidian graph data. Although Graph Neural Networks (GNNs) have shown promising results in detecting fake news, there are some limitations associated with training on GNN alone:

1. **Incompleteness**: GNNs primarily focus on the relational information between users, potentially overlooking other important features from content-based and social context-based approaches.
2. **Overfitting**: Training a GNN on limited data may lead to overfitting, which can reduce the model's generalization performance on unseen data.

Given these limitations, there is a strong motivation to adopt an ensemble approach to improve the effectiveness of fake news detection. An ensemble approach can combine the strengths of various models while mitigating their individual weaknesses, ultimately resulting in a more robust and accurate solution for fake news detection. The proposed methodology involves training a GNN model and retraining the output on an ensemble of machine learning classifiers. This approach offers several advantages over using a GNN alone:

1. **Complementarity**: Combining GNN with an ensemble of machine learning classifiers leverages the complementary strengths of different models, capturing the diverse aspects of fake news detection from content-based, social context-based, and propagation-based perspectives.

2. **Reduced overfitting**: The ensemble of classifiers can help reduce the risk of overfitting by providing a more generalized solution.

3. **Increased accuracy**: By incorporating the output of the GNN into the ensemble of classifiers, the overall accuracy of fake news detection can be improved.

4.2 Proposed Method

The key idea behind this approach is to conduct a series of experiments with the primary objective of achieving higher accuracy than using a single GNN model. To initiate the analysis, a Single Model Analysis is conducted using three different Graph Neural Network (GNN) models: Graph Convolutional Network (GCN), GraphSAGE, and Graph Attention Network (GAT) to establish baseline accuracy. Moreover, each dataset used in the analysis comprises three primary node features, which are represented using text representation learning techniques. Specifically, the three primary node features in each dataset include a 768-dimensional BERT representation, a 300-dimensional Spacy (word2vec) representation, and a 10-dimensional vector that exclusively consists of profile features. These text representation learning techniques enable us to capture and incorporate the semantic and contextual information present within the graph data. By systematically training the graph data using the GNN model and considering the various text representation learning techniques, we aim to establish a solid foundation for accurate analysis and further enhancements in our methodology.

Next, we expand our approach with an Ensemble Model Analysis, incorporating multiple machine learning classifiers to further enhance accuracy. Importantly, the ensemble model analysis is also performed on the three different feature techniques mentioned earlier and involves training the outputs of the GNN model on eight different Machine Learning (ML) classifiers. The results from both analyses are compared to identify the most effective methodology for accurate graph data analysis in terms of accuracy. These steps can be summarized as follows:

Step 1: Single Model Analysis : The accuracy of graph data is evaluated by training it exclusively using a Graph Neural Network (GNN) model, and the resultant accuracy is calculated.

Step 2: Ensemble Model Analysis : The approach is expanded by incorporating an ensemble of machine learning classifiers to further improve the accuracy of the graph data analysis.

Step 3: Comparison of Results : The results from both single and ensemble model approaches are compared to determine the most effective methodology for graph data analysis.

4.2.1 Single Model Analysis (using only a Graph Neural Network)

This experiment employs GNNs with different convolutional layers to predict the result of graph data. For instance, the ”message passing layer” [3, section 3.5] mechanism is used to classify the fake news graph dataset. This message passing happens in every graph neural network layer, where each node in the graph:

1. Gathers information from all neighbouring nodes

2. Perform the node aggregation.

3. Update the node aggreagation

$$\begin{aligned} h_u^{n+1} &= UPDATE^{(n)}(h_u^{(n)}, AGGREGATE^{(n)}(h_v^{(n)}, \forall v \in N(u)) \\ &= UPDATE^{(n)}(h_u^{(n)}, m_{N(u)}^{(k)}) \end{aligned} \tag{4.1}$$

where, **mN(u)** is a message aggregated from u's neighborhood **N(u)**. The functions **UPDATE** and **AGGREGATE** are arbitrary, differentiable functions. Supscripts are used in the above equation to identify the embeddings and functions at different iterations of message-passing. [36]

To classify the graph data, three different types of GNN [3, section 3.5] models are used, namely, Graph Convolutional Networks, GraphSAGE, and Graph Attention Network. The message passing mechanism is common to all GNN models; however, their aggregate and update mechanisms are different. The aggregate and update mechanisms of each of these models are described below:

4.2.2 Graph Convolutional Network (GCN)

Graph Convolutional Network is an advanced form of Convolutional Neural Network (CNN) designed to work directly with graphs. It uses the following function to aggregate and update the nodes' information:

$$H_i^l = ReLU(W^{l-1}\frac{1}{d_i}\sum_{j\in N(i)} H_j^{l-1}) \tag{4.2}$$

where,

- H_i^l represents the feature vector of node i at the l-th layer of the GCN.
- ReLU$(\cdot)$ is the Rectified Linear Unit activation function.
- W^{l-1} represents the weight matrix associated with the $(l-1)$-th layer of the GCN.
- $\frac{1}{d_i}\sum_{j\in N(i)} H_j^{l-1}$ represents the aggregation step in the graph convolution operation.

4.2.3 Graph Attention Networks (GATs)

GATs are based on the concept of transformers [40] that aggregate the neighbouring node's features using a multi-head attention mechanism. Here, the node aggregation is performed by giving them varying weights based on their importance. It uses the following function to update the nodes' information:

$$H_i^l = \alpha(\frac{1}{n}\sum_{n=1}^{n}\sum_{j\in N(i)} a_{i,j}^n W_n H_j) \tag{4.3}$$

where,

- H_i^l represents the feature vector of node i at the l-th layer of the GAT. It denotes the transformed representation of node i obtained after applying the graph attention operation.
- n is the number of heads in the multi-headed attention mechanism.
- W represents the weight matrix.
- $a_{i,j}$ is the attention co-efficient between nodes i and j.

4.2.4 GraphSAGE

In contrast to other graph models, GraphSage uses a subset of neighbouring node features rather than the whole graph to learn the relevant node embeddings. It uses different aggregation functions to update the nodes' information, which can be represented using the following equation:

$$H_i^l = ReLU(W.Concat(H_i^{l-1}, Mean(H_j^{l-1} : j \in N(i)))) \tag{4.4}$$

where,

- H_i^l represents the feature vector of node i at the l-th layer of the GraphSAGE model. It denotes the transformed representation of node i obtained after applying the graph sampling and aggregation operations.
- ReLU$(\cdot)$ is the Rectified Linear Unit activation function.
- W represents the weight matrix used for the linear transformation of the concatenated features.
- Concat$(\cdot)$ denotes the concatenation operation, which combines the features of node i in the previous layer H_i^{l-1} with the mean-pooled features of its neighboring nodes.
- $j \in N(i)$ denotes the set of neighboring nodes of node i in the graph, representing the nodes that are directly connected to node i.
- *Mean* indicates element wise mean pooling. Aggregation functions based on LSTM and Max-Pooling techniques can also be used in GraphSage.

4.2.5 Ensemble Model Analysis

In this study, a novel extension of the GNN architecture is proposed to further improve its predictive performance. Upon obtaining the predicted probabilities for each graph instance, these probabilities are utilized as a new set of features and mapped with their corresponding actual labels (i.e., "true" or "fake"). Thus, the new dataset represents a mapping between the estimated probabilities generated by the GNN and the actual labels.

To take full advantage of this new data representation, various machine learning classifiers [3, section 3.2] are employed to train the dataset. Furthermore, to achieve optimal results, an ensemble approach using Max Voting [3, section 3.3.1] is utilized, combining the predictions of multiple classifiers to produce a single, more accurate prediction.

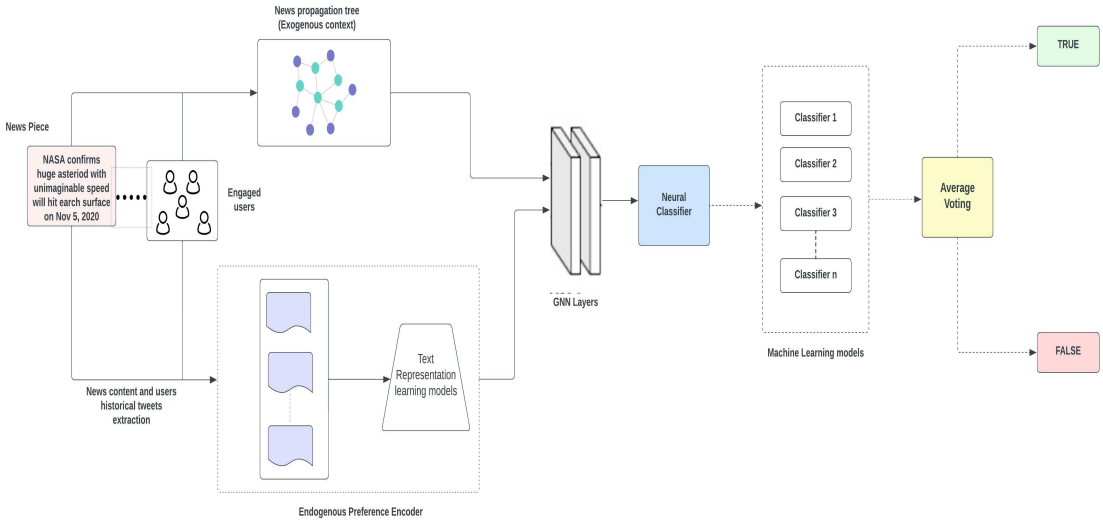

Figure 4.1: Architecture using ensemble of classifiers

The classifiers for ensemble learning are selected based on two criteria.

- The model should be high-performing.
- The model should be diverse.

The high-performing models are determined based on the metric AUC, i.e., accuracy. Similarly, the Pearson correlation coefficient is used to calculate diversity. The steps for finding accuracy using the ensemble approach can be summarized as follows:

1. Train multiple machine learning classifiers on a new dataset.
2. Calculate the accuracy of each classifier.
3. Set a correlation limit.
4. Index and iteratively select models that are not highly correlated.
5. Create combinations of these models, and then use the average ensemble voting method to pick the best-performing model.

Chapter 5

Data Collection and Experimental Setup

This chapter provides a comprehensive overview of the datasets utilized in this experiment, providing detailed descriptions of their specific characteristics, structure, and distribution of fake and real news instances. Additionally, it outlines the experimental setup employed to conduct the study, ensuring transparency and reproducibility.

5.1 Data Source

The dataset from the paper [11] that contains graph data is used to train GNN models in this experiment. The actual news content of this paper [11] is extracted from the FakeNewsNet dataset [41], which contains textual news as well as Twitter social engagement information. This dataset includes both fake and real news, based on two fact-checking datasets, Politifact and GossipCop. The specifics of these details are provided below.

5.1.1 News Content

To gather reliable sources of news that contain both fake and true news, information from the fact-checking websites PolitiFact and GossipCop is employed. Ploitifact contains the political claims that have been verified by journalist and domain experts, whether they are true or false. The actual news about these claims is extracted from the source URL provided on the Politifact website. For the expired news URL in Politifact, the authors of [11] have made use of the Wayback Machine and Google Web Search to identify the news article as the actual news. These news articles serve as the basis for both fake and genuine news articles in this research work.
Similarly, gossipcop is another fact-checking website that contains entertainment news stories accumulated from multiple online media sources. It provides rating levels for new articles ranging from 0 to 10 to classify them according to their authenticity. Since this website mostly showcases fake stories, only news with a score of less than 5 is collected as a source of fake news. However, factual entertainment news is pulled from another reputable, trusted website called EOnline18 in order to maintain a balance between fake and real news.

5.1.2 Social Context

The social context and user engagement for the news article from a fact-checking website are collected using Twitter's Advance Search API. A search query is made from a news article headline with special characters removed to filter out the noise. In addition to obtaining news posts from social media, further information such as replies, likes, user meta data (followers, total posts, etc.), and user engagement in the news dissemination process is also fetched.

5.1.3 Spatiotemporal Information

There are two components to spatiotemporal information: spatial information and temporal information. Spatial information is fetched using the location provided by the user in their profile. Similarly, temporal information tracks the timestamps of user activity in the news propagation process, which is used to study the spread of news articles in this work. This temporal information serves as the foundation for establishing a network for the circulation of news.

	Category	Features	PolitiFact		GossipCop	
			Fake	Real	Fake	Real
News Content	*Linguistic*	# News articles	432	624	5,323	16,817
		# News articles with text	420	528	4,947	16,694
Social Context	*User*	# Users posting tweets	95,553	249,887	265,155	80,137
		# Users involved in likes	113,473	401,363	348,852	145,078
		# Users involved in retweets	106,195	346,459	239,483	118,894
		# Users involved in replies	40,585	18,6675	106,325	50,799
	Post	# Tweets posting news	164,892	399,237	519,581	876,967
	Response	# Tweets with replies	11,975	41,852	39,717	11,912
		# Tweets with likes	31692	93,839	96,906	41,889
		# Tweets with retweets	23,489	67,035	56,552	24,955
	Network	# Followers	405,509,460	1,012,218,640	630,231,413	293,001,487
		# Followees	449,463,557	1,071,492,603	619,207,586	308,428,225
		Average # followers	1299.98	982.67	1020.99	933.64
		Average # followees	1440.89	1040.21	1003.14	982.80
Spatiotemporal Information	*Spatial*	# User profiles with locations	217,379	719,331	429,547	220,264
		# Tweets with locations	3,337	12,692	12,286	2,451
	Temporal	# Timestamps for news pieces	296	167	3,558	9,119
		# Timestamps for response	171,301	669,641	381,600	200,531

Figure 5.1: Statistics of FakeNewsNet [41].

5.2 Data Processing

The graph dataset from [11] is used in this experiment, which generates graphs based on the news content from [41]. The graph data is constructed using both endogenous and exogenous information from the user. Here, endogenous refers to the user's personal preferences that may be useful in determining the user's interest in a particular post, such as historical posts, while exogenous refers to all users involved in sharing a particular news article. The following steps are taken to process and obtain the graph data.

1. First, news posts and user-historical posts are encoded using text learning strategies like word2vec and BERT.

2. Second, a tree-structured propagation graph is constructed for each piece of news. In this case, the news is the root node, and the child nodes are the people who shared the news.

5.2.1 Endogenous Preference Encoding

Modelling users endogenous preferences using only their social network information is not a straightforward task. To implicitly consider a user's preferences, their historical posts are encoded. The FakeNewsNet dataset [41], which contains news content and Twitter social engagement information, is used. Similarly, historical posts are extracted using the Twitter Developer API for each account that has retweeted a particular news post. As part of the user preference modelling process, the most recent 200 tweets from each account are crawled, resulting in close to 20 million tweets. Similarly, the historical posts of suspended or deleted accounts are replaced with randomly selected tweets from accessible users engaged in the same news story.
To encode the news textual information and user preference, two commonly used word embeddings (word2vec and BERT) are utilized rather than training local corpora. These word embeddings have been pretrained on a large corpus and are therefore capable of encoding more semantic similarities between words and sentences. For word2vec, a 300-dimensional vector that has been pretrained using SpaCy is used. SpaCy is a free, open-source Python library used in Natural Language Processing. It comprises pretrained vectors for 680k words, and user preference representation is obtained by averaging the vectors of existing words in the user's 200 most recent historical tweets. The news embedding is obtained in a similar manner. For the BERT model, the news and user information are encoded using the BERT-Large model with a maximum input sequence length of 512 tokens.

5.2.2 Exogenous Context Extraction

The user's exogenous context refers to all users that are engaged in a particular piece of news content. The propagation graph is formed using the retweet information from new pieces. It is constructed using the two rules listed below:

1. Given an account a_i , if a_i shares the same news piece later than at least one following account in a_1,a_2,a_3, ..., a_n , the latest timestamp to account v_i is used to estimate the news spread from the account.

2. The news is conservatively estimated to spread from the account with the highest number of followers if account v_i does not follow any accounts in the retweet sequence, including the source account, because according to Twitter's guidelines for content distribution, tweets from accounts with more followers have a higher chance of being viewed and retweeted by other users.

5.2.3 Information Fusion

Previous research [7, 26] has proven that integrating user attributes (features) with a news propagation graph can improve the efficiency of detecting fake news. GNN can leverage this task since it can encode both node features and graph structure. The fusion of information is performed using a hierarchical approach.
First, endogenous and exogenous information are fused together. Here, as node features, news textual embedding and user preference embedding are used along with user attributes. After a message passing layer where the features of neighbourhood nodes are aggregated, a readout function is applied over all node embeddings to generate the embedding of a news propagation graph. To get the graph embedding, the readout function performs the pooling operation on all node embeddings.
Second, because news content tends to show more signs of trustworthiness [11], textual embedding and user engagement embedding are combined to improve news embedding.

The last step is to feed the fused news embedding, which is made up of user engagement embedding and news textual embedding, into a two-layer multi-layer perceptron (MLP), which has two output neurons that show the probabilities for fake news and real news.
The model is trained with a binary cross-entropy loss function, and then it is updated with mini-batch gradient descent.

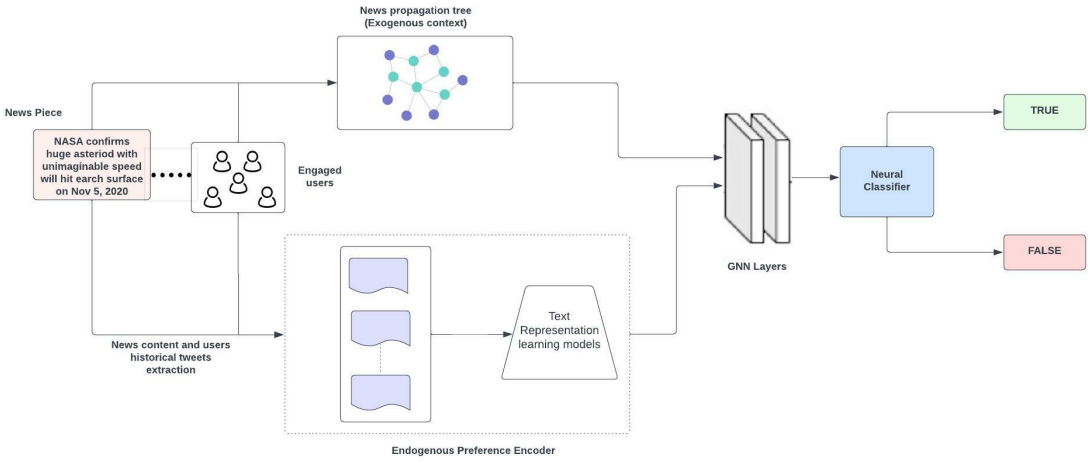

Figure 5.2: A news propagation graph (exogeneous context) is extracted from news content, and endogenous information is encoded based on users' historical posts and news pieces. The endogenous and exogenous data are merged together using a GNN encoder and fed into the neural classifier to predict the credibility of news.

5.3 Expiremental Setup

DataSets

The proposed model is evaluated based on two real-world text datasets: Gossipcop and Politifact from UPFD [11]. In both datasets, nodes refer to users, edges represent retweets or reshares of news, and features are extracted in terms of a user's attributes and their historical tweets. Each dataset contains two binary labels: true (1) or false (0). The statistics of two datasets are shown in the table below:

Experimental Setups

In this experiment, the cutting-edge capabilities of the PyTorch Geometric and Sklearn packages are utilized to achieve better accuracy in graph data analysis. The GNN models are implemented using the PyTorch Geometric library, while individual machine learning classifiers and the ensemble approach are implemented using the default settings from the Sklearn package.

Table 5.1: Graph data for the GNN model

Datasets	Gossipcop	Politifact
Graphs	5464	314
Fake News	2732	157
Total Nodes	314,262	41,054
Total Edges	308,798	40,740
Avg.Nodes per Graph	58	131

A graph dataset composed of a balanced proportion of authentic and fake news is employed, which is then split into three sets: training, validation, and testing, with a 60%, 20%, and 20% distribution, respectively. The validation set is utilized to monitor overfitting and ensure the robustness of the model. The experiment is conducted using a set of hyperparameters that include a unified graph embedding size of 128, 100 epochs, a binary-cross entropy loss function, a learning rate of 0.01, an Adam optimizer with a weight decay of 0.01, and a batch size of 128. Additionally, early stopping with a patience of 5 iterations is employed to prevent overfitting in the GNN model training process. Figures 5.3 and 5.4 display the distribution of the training, validation, and testing sets used in the experiment, providing a clear visualization of the data employed in this analysis.

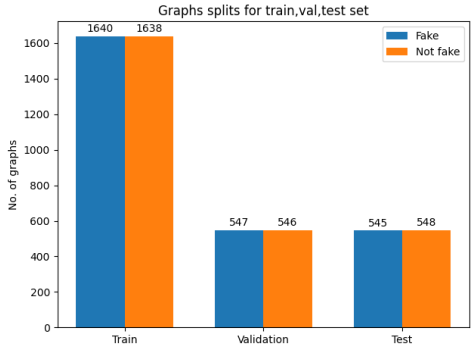

Figure 5.3: Data distribution of Gossipcop

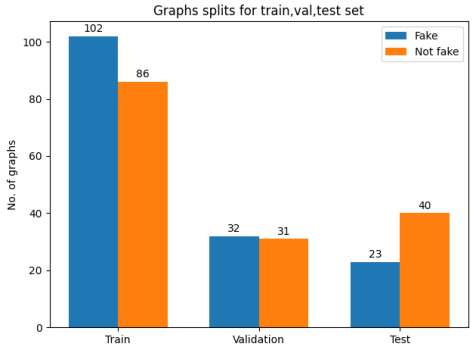

Figure 5.4: Data distribution of Politifact

The experiment is performed on three significant variations of Graph Neural Networks (GNNs): Graph Convolutional Network (GCN), Graph Attention Network (GAT), and Graph SAGE (GSAGE). For each of these GNN models, three different embedding techniques are applied, resulting in a total of 9 experiments for the Gossipcop dataset and 9 experiments for the Politifact dataset.
The aim of this experiment is to enhance the accuracy of each of these GNN variants by employing an ensemble approach. These models are first trained using a carefully curated training set and undergo a rigorous evaluation process. In order to prevent oversmoothing and maintain the structural information of the graph, only a single message passing layer is utilized for all three variants, followed by three linear transformation layers. Global Max Pooling is employed to extract crucial features and reduce the dimensionality of the graph's feature map. The accuracy of the trained models is thoroughly assessed on training, validation, and testing sets.

To further enhance the performance of the Graph Neural Network (GNN) models, an ensemble approach is applied. The ensemble method involves training the outputs of the GNN models to eight different Machine Learning (ML) classifiers, including Logistic Regression, Decision Trees, Random Forest Classifier, Extra Tree Classifier, KNeighborsClassifier, SVC, AdaBoost Classifier, and Gradient Boosting. To ensure that the models are uncorrelated, the Pearson coefficients between them are calculated, and only the models with the highest accuracy are selected for the final ensemble approach. The final ensemble approach employs an average voting system, in which the outputs of the selected ML classifiers are combined and averaged to produce a final prediction. The optimal combination of ML classifiers is determined based on the combination that produces the highest performance.

Chapter 6

Results and Analysis

This experiment employs both the Gossipcop and Politifact datasets. Each of these datasets comprises three primary node features that are represented using the text representation learning technique. They consist of a 768-dimensional Bert, a 300-dimensional Spacy, and a 10-dimensional vector consisting of only profile features. Three types of GNN are selected to detect fake news: GAT, GraphSAGE, and GCN. For each GNN model and text representation learning technique, around 100 epochs are utilized to train and evaluate the accuracy and loss functions.

To begin with, each GNN variant is trained and evaluated using three text representational learning algorithms. First, the dataset is trained solely on GNN variants, and then the ensemble is applied. The GNN trained with ensemble classifiers is represented as GNN-EN, where "GNN" is the variant of Graph Neural Network and "EN" represents the ensemble of ML classifiers. Tables 6.1, 6.2, 6.3 analyze the performance of the model on Gossipcop and Politifact datsets. The performance of the models is determined by their accuracy and F1 score.

Table 6.1: Performance Score on the Gossipcop Dataset

Gossipcop												
Feature	**GNN Models**						**Ensemble Learning**					
	GCN		GAT		GSAGE		GCN-EN		GAT-EN		GSAGE-EN	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
Word2Vec	97.34	97.40	96.98	96.97	97.98	97.99	98.35	98.37	97.98	98.02	98.03	98.05
Bert	96.98	97.05	97.25	97.31	97.07	97.14	97.25	97.30	98.17	98.18	98.17	98.19
Profile	85.45	85.37	90.39	90.49	92.31	92.37	85.45	85.40	90.57	90.74	92.40	92.52

The accuracy and the F1 score of three graph neural network models (GCN, GAT, and GSAGE) and the proposed ensemble method are presented in Table 6.1, showcasing the performance on the gossipcop dataset. The feature column displays the different embedding techniques applied to the GNN and GNN-EN models.
The performance of these models is evaluated using two standard metrics: accuracy (ACC) and F1 score. As illustrated in the performance metrics, the proposed ensemble method achieved better results, outperforming the individual models in terms of accuracy and F1 Score across all embedding techniques. In particular, the ensemble learning approach using the Word2Vec and Bert encoding

techniques demonstrated a 1% increase in accuracy and F1 score compared to the individual models. The performance of the Profile encoding approach was comparable to single GNN model analysis, yet with a slight improvement in ACC and F1 Score. Overall, the results of this study demonstrate that the use of an ensemble model improved the performance of all 9 experiments performed on Graph Neural Networks (GNNs) using three distinct embedding features on the gossipcop dataset.

Table 6.2: Performance Score on the Politifact Dataset

Politifact												
Feature	**GNN Models**						**Ensemble Learning**					
	GCN		GAT		GSAGE		GCN-EN		GAT-EN		GSAGE-EN	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
Word2Vec	93.08	92.39	82.53	86.74	85.71	87.67	94.14	93.64	82.54	86.74	90.47	92.30
Bert	87.30	89.47	85.71	88	84.12	86.84	87.30	89.47	85.71	88	85.71	88
Profile	73.01	77.92	69.84	75.32	71.42	76.92	74.60	79.49	79.36	83.54	73.02	76.92

Table 6.2 showcases the comparison of the accuracy and F1 score between three Graph Neural Network models (GCN, GAT, and GSAGE) trained on the Politifact dataset with the proposed ensemble technique. The feature column indicates the embedding techniques applied to each of the models. The results highlight a significant improvement in performance by the ensemble models (GCN-EN, GAT-EN, and GSAGE-EN) as compared to the individual models. For instance, the GCN-EN model outperformed GCN in terms of accuracy and F1 score by more than 1% for Word2Vec and profile features, while BERT features showed comparable performance. The GAT-EN model achieved a remarkable improvement of 10% in ACC and 8% in F1 score when compared to the GAT model for profile features. Furthermore, GSAGE-EN also showed an improvement of 5% in accuracy and F1 score for Word2Vec and a slight improvement of 1% for BERT and profile features. These results highlight the efficacy of the proposed ensemble technique in enhancing the performance of Graph Neural Network models.

Table 6.3: Performance of GNN Variants and GNN Ensemble

	Model	**Gossipcop**		**Politifact**	
		Accuracy	F1 Score	Accuracy	F1 Score
GNN	GCN	97.34	97.40	93.08	92.39
	GAT	97.25	97.31	85.71	88
	GSAGE	97.98	97.99	85.71	87.67
GNN-EN	GCN-EN	**98.35**	**98.37**	**94.14**	**93.64**
	GAT-EN	**98.17**	**98.18**	**85.71**	**88**
	GSAGE-EN	**98.17**	**98.19**	**90.47**	**92.30**

Table 6.3 shows a comprehensive comparison of the accuracy and F1 scores of three Graph Neural Network models (GCN, GAT, and GSAGE) that were trained on two datasets (Gossipcop and Politifact), as well as their corresponding ensemble methods. The results shown in Table 6.3 are a summary of the findings presented in Table 6.1 and Table 6.2, and provide a clear and concise overview of the best performance results achieved by each model and its ensemble counterpart across the three different embedding techniques.

The experiment results indicate a marked improvement in the accuracy and F1 scores when using the ensemble model compared to the individual models for both the Gossipcop and Politifact datasets. The GCN-EN and GAT-EN models showed an improvement of 1% in both accuracy and F1 scores for the Gossipcop dataset, while the GCN-EN and GSAGE-EN models demonstrated an improvement of 1% and 5% respectively, for the Politifact dataset. This highlights the efficacy of the proposed ensemble method in enhancing the performance of graph neural networks in fake news detection. The results of the 18 experiments conducted on both datasets using the ensemble method demonstrate its superiority in improving the performance of the models.

Best performing GNN ensemble model (GNN-EN)

The results of the experiment, including the combination of the best performing ensemble model for each Graph Neural Network (GNN), accuracy (ACC), F1 score, and graphs of train and validation performance over 100 epochs, are presented below. The accompanying figure provides a visual representation of the results, with the left side showing the validation accuracy versus epoch of the individual GNN models and the right side showing the validation accuracy versus training size of the ensemble models.
To ensure a comprehensive evaluation of the ensemble models, the validation accuracy was assessed using a 10-fold cross-validation approach. This method allows for a thorough evaluation of the model's performance and provides a more robust and reliable representation of the model's accuracy and ability to generalize to new data.

Gossipcop

1. GCN-EN
Feature: word2vec
Combination: SVC(kernel='linear', probability=True)
Test Accuracy: 0.9835
F1 Score: 0.9837

Among the various word embedding techniques tested, word2vec was found to provide the optimal results. The ensemble model used to classify the text data was a Support Vector Classification (SVC) model with a linear kernel and probability set to True, which was determined to be the best performing combination for this dataset. The results showed that this combination achieved a test accuracy of 0.9835 and an F1 Score of 0.9837, indicating a high level of accuracy and a strong balance between precision and recall.

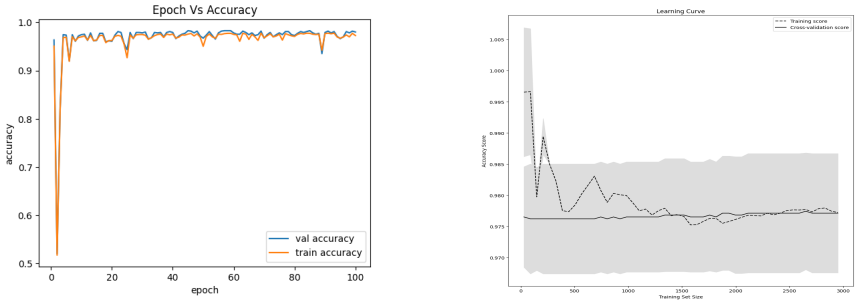

Figure 6.1: Comparison of Validation Accuracy in GCN and Ensemble Model Across Epochs and Training Size

2. GAT-EN
Feature: BERT
Combination: DecisionTreeClassifier(max depth=1, random state=0), SVC(kernel='linear', probability=True), ExtraTreesClassifier(random state=0), KNeighborsClassifier(n neighbors=3)
Test Accuracy: 0.9817
F1 Score: 0.9818

Here, the feature representation - BERT embedding technique was found to be the optimal representation among the other embedding techniques evaluated. The ensemble model used with optimal performance to analyze the data was a combination of four different classifiers: DecisionTreeClassifier with a maximum depth of 1 and random state set to 0, Support Vector Classification (SVC) model with a linear kernel and probability set to True, ExtraTreesClassifier with random state set to 0, and KNeighborsClassifier with 3 neighbors. The results showed that this combination achieved a test accuracy of 0.9817 and an F1 Score of 0.9818 on the gossipcop dataset, demonstrating superior performance compared to others.

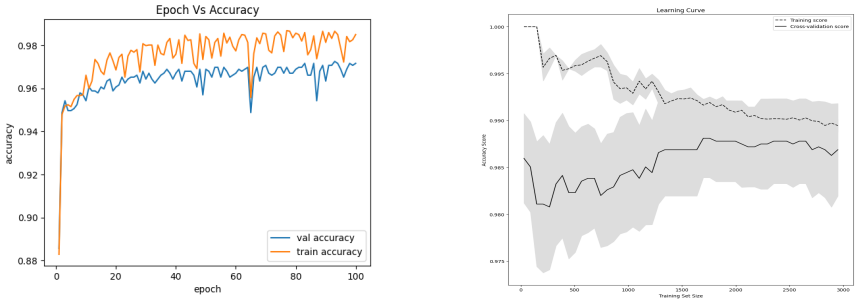

Figure 6.2: Comparison of Validation Accuracy in GAT and Ensemble Model Across Epochs and Training Size

3. GSAGE-EN
Feature: BERT
Combination: RandomForestClassifier(max depth=2, random state=0)
Test Accuracy: 0.9817
F1 Score: 0.9819

Here, BERT as the feature representation was found to be the optimal choice among the other embedding techniques evaluated. The ensemble model used to analyze the text data was a combination of one classifier, a Random Forest Classifier with a maximum depth of 2 and a random state set to 0. The results showed that this combination achieved a test accuracy of 0.9817 and an F1 Score of 0.9819 on the gossipcop dataset.

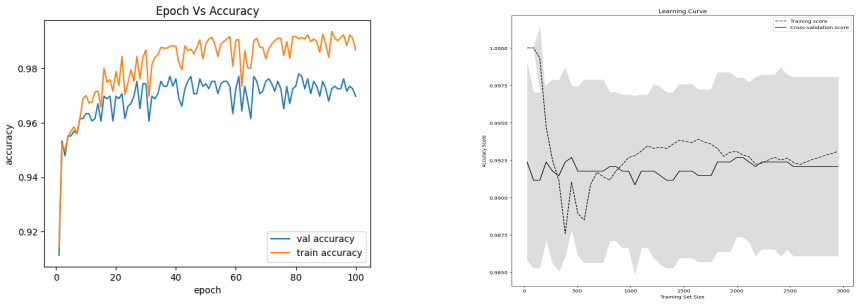

Figure 6.3: Comparison of Validation Accuracy in GSAGE and Ensemble Model Across Epochs and Training Size

Politifact

1. GCN-EN
Feature: word2vec
Combination: DecisionTreeClassifier(max depth=1, random state=0)
Test Accuracy: 0.9414
F1 Score: 0.9364

The feature used in this analysis was the word2vec technique for word embedding, which showed optimal results compared to other embedding techniques. The combination used in this analysis was the DecisionTreeClassifier with a maximum depth of 1 and a random state of 0. The results of the ensemble model showed a test accuracy of 0.9414 and an F1 score of 0.9364 on the Politifact dataset.

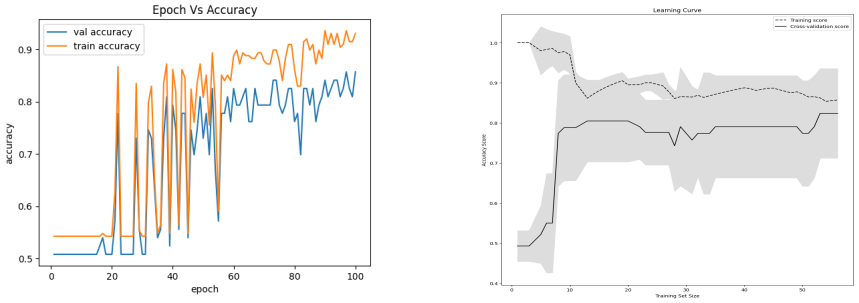

Figure 6.4: Comparison of Validation Accuracy in GCN and Ensemble Model Across Epochs and Training Size

2. GAT-EN
Feature: BERT
Combination: RandomForestClassifier(max depth=2, random state=0)
Test Accuracy: 0.8571
F1 Score: 0.88

The BERT feature was used as the word embedding technique and produced optimal results compared to other embedding techniques. The ensemble model combination used was a RandomForestClassifier with a maximum depth of 2 and a random state set to 0. The combination produced a test accuracy of 0.8571 and an F1 score of 0.88.

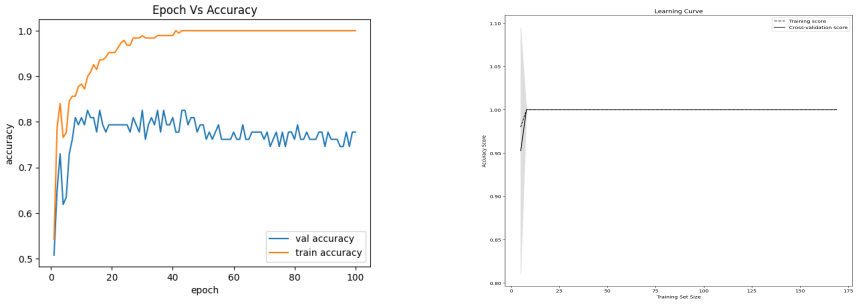

Figure 6.5: Comparison of Validation Accuracy in GAT and Ensemble Model Across Epochs and Training Size

3. GSAGE-EN
Feature: word2vec
Combination: LogisticRegression
Test Accuracy: 0.9047
F1 Score: 0.9230

Here, the feature used was the word2vec word embedding technique, which was evaluated against other embedding techniques and showed optimal results. The combination used was a single Logistic Regression classifier, which achieved a Test Accuracy of 0.9047 and an F1 Score of 0.9230 on the politifact dataset.

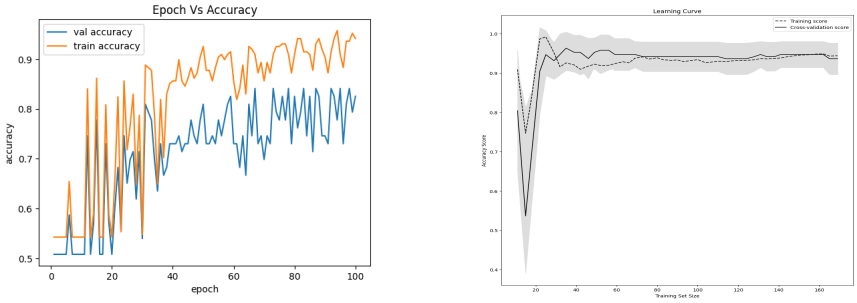

Figure 6.6: Comparison of Validation Accuracy in GSAGE and Ensemble Model Across Epochs and Training Size

6.1 Comparative Performance Evaluation: GNN-EN versus Prior Studies

The section presents a comparative evaluation of the performance of prior studies, including UPFD [11], Mahmud et al. [10], and GCNFN [26] models, alongside the GNN-EN model. The objective of this analysis is to assess the effectiveness and superiority of GNN-EN by comparing its performance with established baselines.

Table 6.4: Performance of GSAGE-EN and GSAGE from UPFD [11]

	Features	**Gossipcop**	**Politifact**
		Accuracy	Accuracy
UPFD (GSAGE)	word2vec	96.81	80.54
	BERT	97.23	84.62
	Profile	92.19	77.38
GNN-EN (GSAGE-EN)	word2vec	**98.40**	**90.47**
	BERT	**98.17**	**85.71**
	Profile	**92.40**	73.02

In the research conducted, a comparison was made between the GSAGE-EN model and the GSAGE model from UPFD in terms of accuracy. The best performing model is represented in bold. Among the tested models in UPFD, GSAGE emerged as the best performer. However, the GSAGE-EN model outperformed UPFD across all features when evaluated on the Gossipcop dataset. Notably, the GSAGE-EN model exhibited significant accuracy improvements, surpassing the UPFD model by approximately 10% for word2vec and over 1% for BERT feature on the Politifact dataset. These results demonstrate the effectiveness of the GNN-EN model in accurately capturing and utilizing relevant features, showcasing its superiority in achieving improved performance in both datasets.

Table 6.5: Performance Comparision of GNN-EN with Mahmud et al.'s [10]

Feature	**Gossipcop**						**Politifact**					
	GCN	GCN-EN	GAT	GAT-EN	GSAGE	GSAGE-EN	GCN	GCN-EN	GAT	GAT-EN	GSAGE	GSAGE-EN
Word2Vec	96.48	**98.35**	96.42	**97.98**	96.52	**98.17**	82.81	**94.14**	78.28	**82.54**	75.57	**90.47**
Bert	96.84	**97.25**	96.31	**98.17**	96.99	**98.17**	84.67	**87.30**	84.62	**85.71**	84.62	**85.71**
Profile	90.80	85.45	93.27	90.57	92.50	90.57	76.47	74.60	74.66	**79.36**	78.28	73.02

In comparison to the analysis conducted by researchers from [10] on GNN models, the evaluation focused on performance in terms of accuracy. The best performing model is represented in bold. Across both the Gossipcop and Politifact datasets, the GNN-EN model surpassed all other models in terms of word2vec and BERT feature vectors. Specifically, on the Gossipcop dataset, the GNN-EN model achieved a 1-2% higher accuracy compared to other models for both word2vec and BERT features. Similarly, on the Politifact dataset, the GCN-EN, GAT-EN, and GSAGE-EN models surpassed other models. For word2vec, GCN-EN, GAT-EN, and GSAGE-EN achieved improvements

of approximately 12%, 4%, and 15%, respectively. In the case of BERT, GCN-EN, GAT-EN, and GSAGE-EN outperformed other models by approximately 3%, 1%, and 1%, respectively. These results emphasize the superior performance of the GNN-EN model in both datasets, showcasing its effectiveness in surpassing other models when considering various feature vectors.

Table 6.6: Performance Comparison with Best-Performing Models of Baselines

Model	Gossipcop	Politifact
UPFD[11]	96.99	84.67
Mahmud et al.[10]	97.23	84.62
GCNFN [26]	96.38	83.16
GNN-EN	**98.35**	**94.14**

The table displays the best performing model from each research study on the Gossipcop and Politifact datasets, with GNN-EN being one of the models. The GNN-EN model, among others, demonstrated the highest performance. It achieved an accuracy of 98.35% on the Gossipcop dataset and 94.14% on the Politifact dataset. These results highlight the effectiveness of the GNN-EN model in accurately predicting outcomes on both datasets, surpassing the performance of the other models.

Chapter 7

Conclusion and Future Work

The present study investigated the effectiveness of ensemble learning as a technique to improve the performance of Graph Neural Networks (GNNs) in determining the authenticity of news articles. The UPFD graph dataset, which includes both endogenous and exogenous user preferences, was used for the analysis. The study involved evaluating the performance of three different encoding features and three diverse GNNs, and the results were combined using the ensemble learning approach. Additionally, several baseline models were compared to assess the performance of the proposed ensemble method. The findings of the study revealed that the ensemble method outperformed individual GNNs in enhancing the performance of news authenticity detection. Using this method we were able to obtain upto 10% more accurate results in detecting fake news. Furthermore, when compared to the baseline models, the ensemble approach demonstrated superior performance. These results indicate the effectiveness of ensemble learning in improving the accuracy and reliability of news article authenticity detection when compared to both individual GNNs and other baseline models.

Although the study produced promising results, there is room for further research to expand the scope of the investigation. One area for future exploration is the collection of datasets from a wider range of social media platforms and multiple languages to improve the generalizability of the findings. This would enable researchers to test the robustness of the ensemble approach in various contexts and improve its practical application.

Another challenge for future research is the computational complexity and scalability of the approach, especially when dealing with large and complex graph datasets. To mitigate the computational expense, model compression techniques can be employed without significant loss in performance. It is important to address the challenge of real-time applications, which may require faster and more efficient algorithms to handle the massive amounts of data involved. One possible solution is to explore the combination of multiple graphs into a single ensemble to improve overall performance. This would involve designing new methods to integrate multiple graphs efficiently and effectively while ensuring the quality of the results.

Bibliography

[1] Nicole Martin. How social media has changed how we consume news. *Forbes*, Oct 2022.

[2] Yasmim Mendes Rocha, Gabriel Acácio de Moura, Gabriel Alves Desidério, Carlos Henrique de Oliveira, Francisco Dantas Lourenço, and Larissa Deadame de Figueiredo Nicolete. The impact of fake news on social media and its influence on health during the covid-19 pandemic: A systematic review. *Journal of Public Health*, pages 1–10, 2021.

[3] Chandra Mouli Madhav Kotteti, Xishuang Dong, and Lijun Qian. Ensemble deep learning on time-series representation of tweets for rumor detection in social media. *Applied Sciences*, 10(21):7541, 2020.

[4] Alistair Coleman. âhundreds deadâ because of covid-19 misinformation, Aug 2020.

[5] Md Saiful Islam, Tonmoy Sarkar, Sazzad Hossain Khan, Abu-Hena Mostofa Kamal, SM Murshid Hasan, Alamgir Kabir, Dalia Yeasmin, Mohammad Ariful Islam, Kamal Ibne Amin Chowdhury, Kazi Selim Anwar, et al. Covid-19–related infodemic and its impact on public health: A global social media analysis. *The American journal of tropical medicine and hygiene*, 103(4):1621, 2020.

[6] Na Bai, Fanrong Meng, Xiaobin Rui, and Zhixiao Wang. Rumour detection based on graph convolutional neural net. *IEEE Access*, 9:21686–21693, 2021.

[7] Yi Han, Shanika Karunasekera, and Christopher Leckie. Graph neural networks with continual learning for fake news detection from social media. *arXiv preprint arXiv:2007.03316*, 2020.

[8] Zhang Jiawei, Dong Bowen, and S Yu Philip. Fakedetector: Effective fake news detection with deep diffusive neural network. In *Proceedings of the 2020 IEEE 36th International Conference on Data Engineering (ICDEâ20)*, pages 1826–1829, 2020.

[9] Shantanu Chandra, Pushkar Mishra, Helen Yannakoudakis, Madhav Nimishakavi, Marzieh Saeidi, and Ekaterina Shutova. Graph-based modeling of online communities for fake news detection. *arXiv preprint arXiv:2008.06274*, 2020.

[10] Fahim Belal Mahmud, Mahi Md Sadek Rayhan, Mahdi Hasan Shuvo, Islam Sadia, and Md Kishor Morol. A comparative analysis of graph neural networks and commonly used machine learning algorithms on fake news detection. In *2022 7th International Conference on Data Science and Machine Learning Applications (CDMA)*, pages 97–102. IEEE, 2022.

[11] Yingtong Dou, Kai Shu, Congying Xia, Philip S Yu, and Lichao Sun. User preference-aware fake news detection. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2051–2055, 2021.

[12] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1):22–36, 2017.

[13] Xinyi Zhou and Reza Zafarani. A survey of fake news: Fundamental theories, detection methods, and opportunities. *ACM Computing Surveys (CSUR)*, 53(5):1–40, 2020.

[14] Victoria L Rubin, Niall Conroy, Yimin Chen, and Sarah Cornwell. Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of the second workshop on computational approaches to deception detection*, pages 7–17, 2016.

[15] Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 2931–2937, 2017.

[16] Tong Chen, Xue Li, Hongzhi Yin, and Jun Zhang. Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 40–52. Springer, 2018.

[17] Raveena Dayani, Nikita Chhabra, Taruna Kadian, and Rishabh Kaushal. Rumor detection in twitter: An analysis in retrospect. In *2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, pages 1–3. IEEE, 2015.

[18] Kai Shu, Suhang Wang, and Huan Liu. Beyond news contents: The role of social context for fake news detection. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 312–320, 2019.

[19] Kai Shu, Suhang Wang, and Huan Liu. Understanding user profiles on social media for fake news detection. In *2018 IEEE conference on multimedia information processing and retrieval (MIPR)*, pages 430–435. IEEE, 2018.

[20] Yunfei Long, Qin Lu, Rong Xiang, Minglei Li, and Chu-Ren Huang. Fake news detection through multi-perspective speaker profiles. In *Proceedings of the eighth international joint conference on natural language processing (volume 2: Short papers)*, pages 252–256, 2017.

[21] Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD workshop on mining data semantics*, pages 1–7, 2012.

[22] Eugenio Tacchini, Gabriele Ballarin, Marco L Della Vedova, Stefano Moret, and Luca De Alfaro. Some like it hoax: Automated fake news detection in social networks. *arXiv preprint arXiv:1704.07506*, 2017.

[23] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684, 2011.

[24] Adam Kucharski. Study epidemiology of fake news. *Nature*, 540(7634):525–525, 2016.

[25] Soroush Vosoughi, Deb Roy, and Sinan Aral. The spread of true and false news online. *science*, 359(6380):1146–1151, 2018.

[26] Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein. Fake news detection on social media using geometric deep learning. *arXiv preprint arXiv:1902.06673*, 2019.

[27] Tom Michael Mitchell. *Machine learning*. McGraw-Hill, 1997.

[28] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[29] Dipayan Sarkar and Vijayalakshmi Natarajan. *Ensemble Machine Learning Cookbook: Over 35 practical recipes to explore ensemble machine learning techniques using Python*. Packt Publishing Ltd, 2019.

[30] What is a decision tree? https://www.ibm.com/topics/decision-trees, 2022.

[31] Vegi Shanmukh. Image classification using machine learning-support vector machine(svm), Mar 2021.

[32] Avinash Navlani. Knn classification tutorial using scikit-learn, Aug 2018.

[33] Learning with. Hands-on machine learning with scikit-learn, keras, and tensorflow, 2nd edition, 2019.

[34] Sunil Ray. Essentials of machine learning algorithms (with python and r codes), 2017, Sep 2017.

[35] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57â81, 2020.

[36] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61â80, Jan 2009.

[37] Zhengyang Wang and Shuiwang Ji. Second-order pooling for graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1â1, 2020.

[38] Christopher Danforth, Advisor Peter, Sheridan Dodds, Kelly Laurent Hebert-Dufresne, Rohan, and Cynthia Forehand. Using word embeddings to explore the language of depression on twitter, 2019.

[39] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Google, and A Language. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Google Brain, Google Research, Llion Jones, Aidan Gomez, RAukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[41] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media. *Big Data*, 8(3):171â188, Jun 2020.