8-2020

Fake News Detection in Social Media Using Machine Learning and Deep Learning

Chandra Mouli Madhav Kotteti
Prairie View A&M University

FAKE NEWS DETECTION IN SOCIAL MEDIA USING MACHINE LEARNING AND DEEP LEARNING

CHANDRA MOULI MADHAV KOTTETI

PRAIRIE VIEW A&M UNIVERSITY
TEACHING
RESEARCH
SERVICE
PVAM
1876

DOCTOR OF PHILOSOPHY
ELECTRICAL ENGINEERING

ROY G. PERRY COLLEGE OF ENGINEERING
PRAIRIE VIEW A&M UNIVERSITY

2020

FAKE NEWS DETECTION IN SOCIAL MEDIA USING MACHINE LEARNING AND DEEP LEARNING

A Dissertation
by

CHANDRA MOULI MADHAV KOTTETI

Submitted to the Office of Graduate Studies of
Prairie View A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2020

Major Subject: Electrical Engineering

FAKE NEWS DETECTION IN SOCIAL MEDIA USING MACHINE LEARNING AND DEEP LEARNING

A Dissertation

by

CHANDRA MOULI MADHAV KOTTETI

Submitted to the Office of Graduate Studies of
Prairie View A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved as to style and content by:

[redacted]

Lijun Qian
Committee Chair

[redacted]

Kelvin Kirby
Committee Member

[redacted]

Xiangfang Li
Committee Member

[redacted]

Xishuang Dong
Committee Member

[redacted]

Na Li
Committee Member

[redacted]

Kelvin Kirby
Head of Department

[redacted]

Pamela H. Obiomon, Dean
Roy G. Perry College of Engineering

[redacted]

Dorie J. Gilbert
Dean, Office of Graduate Studies

August 2020

Major Subject: Electrical Engineering

ABSTRACT

Fake News Detection in Social Media Using Machine Learning and Deep Learning
(August 2020)

Chandra Mouli Madhav Kotteti, Master of Science in Applied Computer Science,
Northwest Missouri State University;
Bachelor of Technology in Electrical and Electronics Engineering, Koneru
Lakshmaiah College of Engineering;
Chair of Advisory Committee: Dr. Lijun Qian

Fake news detection in social media is a process of detecting false information that is intentionally created to mislead readers. The spread of fake news may cause social, economic, and political turmoil if their proliferation is not prevented. However, fake news detection using machine learning faces many challenges. Datasets of fake news are usually unstructured and noisy. Fake news often mimics true news. In this study, a data preprocessing method is proposed for mitigating missing values in the datasets to enhance fake news detection accuracy. The experimental results show that Multi-Layer Perceptron (MLP) classifier combined with the proposed data preprocessing method outperforms the state-of-the-art methods.

Furthermore, to improve the early detection of rumors in social media, a time-series model is proposed for fake news detection in social media using Twitter data. With the proposed model, computational complexity has been reduced significantly in terms of machine learning models training and testing times while achieving similar results as state-of-the-art in the literature. Besides, the proposed method has a simplified feature extraction process, because only the temporal features of the Twitter data are used. Moreover, deep learning techniques are also applied to fake news detection. Experimental results demonstrate that deep learning methods outperformed traditional machine learning models. Specifically, the ensemble-based deep learning classification model achieved top performance.

DEDICATION

To my mother JAYALAKSHMI MULAGADA, and
To my guru late 'BRAHMA SHRI' SRINIVASA RAO YELLAMRAJU.

ACKNOWLEDGMENTS

I am most fortunate to have been born to Mrs. Jayalakshmi Mulagada. She is the one I adore the most. She has done many good things for me, which I cannot express in mere words. I have seen her making significant sacrifices many times for my prosperity, well-being, and happiness. Her fighting spirit towards life and mannish deeds are second to none, and I will wholeheartedly cherish her efforts for a lifetime. I can proudly say that I would not even be in the position where I am now without her.

I am immensely thankful to Nature for giving me a chance to do discipleship under my guru late 'Brahma Shri' Srinivasa Rao Yellamraju. His influence in mylife has made me perceive the world from a whole new dimension that helped me come out of several tough phases in my past. Should for any reason and at any point in my life I am happy, then his contribution is always there in my happiness.

I am grateful to my academic adviser Dr. Lijun Qian for his outstanding support, encouragement, and timely advice throughout my doctoral degree. He helped me in expanding my knowledge boundaries and grow as an independent researcher. His knowledge, experience, and technical advice were hugely instrumental in the success of this study.

I am forever indebted to Dr. Na Li, who has always been a wonderful mentor and who laid a robust foundation for my doctoral program. Her influence on my academic journey is extremely special and phenomenal to me. As a student, working with her was a great opportunity for me to learn, practice, and achieve great heights, both academically and professionally.

I am thankful to my doctoral degree committee, namely, Dr. Lijun Qian, Dr.

Kelvin Kirby, Dr. Xiangfang Li, Dr. Xishuang Dong, and Dr. Na Li, for their immense support at all times during my doctoral program in the Department of Electrical and Computer Engineering at Prairie View A&M University. I thank Dr. Matthew N. O. Sadiku, Dr. Richard Wilkins, and other faculty members and staff for their kindness, teaching, and educational support. I give special thanks to the U.S. Office of the Under Secretary of Defense for Research and Engineering for sponsoring this study.

I am profoundly grateful to all my family members and relatives, especially, the late Chinnamma Mulagada, the late Yenkanna Mulagada, the late Durga Rao Mulagada, Appa Rao Mulagada, Sudhakara Rao Mulagada, Padmavathi Mulagada, the late Tanuja Mulagada, the late Sreedhar Mulagada, and the late Someswara Rao Mulagada for their incredible love towards me and my friends Venkatesh, Praveen Pullagoru, Bhanu Chandan, Reese Hammond, Dinesh Valishetty, Krishna Adoni and others for always being there for me. My appreciation is also extended to all my colleagues and friends at CREDIT Center, Prairie View; thank you for the great times that we have shared. Finally, I sincerely admire all those individuals with whom I have interacted and traveled. Their presence in my life helped me to enhance my wisdom levels.

The research work contained in this study is supported in part by the U.S. Office of the Under Secretary of Defense for Research and Engineering (OUSD(R&E)) under agreement number FA8750-15-2-0119. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Office of the Under Secretary of Defense for Research and Engineering (OUSD(R&E)) or the U.S. Government.

NOMENCLATURE

API	Application Program Interface
ANN	Artificial Neural Network
BDL	Bagging Dictionary Learning
Bi-RNN	Bi-directional Recurrent Neural Network
CF	Characteristic Feature
CFT	Characteristic Feature Tree
CNN	Convolutional Neural Network
DT	Decision Trees
DFT	Density Functional Theory
DSTS	Dynamic Series-Time Structure
FN	False Negative
FP	False Positive
GRU	Gated Recurrent Unit
GNB	Gaussian Naive BayesClassifier
HARNN	Hierarchical Attention RNN
HSI	Hyperspectral Image
K-MPLC	K-Mixture of Product Lifecycle
LinearSVC	Linear Support Vector Classification
LR	Logistic Regression
LSTM	Long Short-Term Memory
ML	Machine Learning

MNRD	Merged Neural Rumor Detection
MAR	Missing at Random
MCAR	Missing Completely at Random
MNAR	Missing Not at Random
MLP	Multi-Layer Perceptron
NN	Neural Network
NNE	Neural Network Ensemble
Nu-SVC	Nu-Support Vector Classification
PES	Periodic External Shocks
PLC	Product Life Cycle
RF	Random Forests
RDL	Random Subspace Dictionary Learning
RNN	Recurrent Neural Network
ResNET	Residual Network
RMS	Root Mean Square
SBCB	Selecting Base Classifiers onBagging
SVC	C-Support Vector Classification
SVM	Support Vector Machines
TF-IDF	Term Frequency and Inverse Document Frequency
TN	True Negative
TP	True Positive
URL	Uniform Resource Locator

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

CHAPTER 1

INTRODUCTION

1.1 Research Background

In this digitalized world, the proliferation of information through social media has become quite easy. For example, anyone can instantly create and spread a piece of information with ease using a smartphone. The time people use to consume news through Television and newspapers has almost reached its culmination point because of social media's power. It is not hyperbole if it is said that social media has become the primary source for news consumers, but the biggest problem with the news on social media is the news veracity. Social media news is a mix of both genuine and false information. No worries exist if the news is genuine and correct; however, in the case of news being incorrect, it may cause social, economic, and political turmoil. In the case of time-critical events, the effects may be dreadful.

Nowadays, people rely more on social media services than traditional media because of its advantages, such as social awareness, education, research, global connectivity, real-time sharing of digital information, etc. Over the years, social media users have been increasing more in number. They play a prominent role in building social media networks to communicate with each other, establish new relationships, or share feelings. Even though social media services are helpful in many ways, it, too, has its disadvantages. Some of the critical social media problems are: cyberbullying, hacking, and information privacy and security.

This dissertation follows the style of the IEEE journal *Machine Learning With Big Data: Challenges and Approaches.*

Social media has become a fast and easy way to proliferate news across the world, and they make news readily available for news consumers. However, fake news on social media has been proliferated for personal or social benefits. According to [1], false information has two forms: misinformation (incorrect information) or disinformation (information that is used to deceive its consumers). Fake news is typically a piece of false information in nature, where its primary purpose is to deceive or mislead readers. It has many similarities with spam messages since they share common features such as grammatical mistakes and false information, using a similarly limited set of words. They contain emotionally colored information that affects the reader's opinion [2]. How to detect false information effectively and efficiently on social media is a challenging problem.

1.2 Fake News Detection

The definition of fake news, its impact, control, detection, etc, are discussed next.

1.2.1 Definition. Fake news consists of intentionally and verifiably false information with a motive to mislead readers [3]. Detecting fake news is a layered process that involves the analysis of the news contents to determine the truthfulness of the news. The news could contain information in various formats such as text, video, image, etc. Combinations of different types of data make the detection process difficult.

1.2.2 Impacts of Fake News. The proliferation of fake news may have a huge impact on society. As the contents of fake news are deliberately false, fake news can be used for personal benefits, financial and political gain, and to spoil the reputation of a company, or person. The severity of the impact caused by fake news depends highly on the news creation time and situation, who created the news and his/her social status, and the social media platform used. If fake news propagation is not prevented in the early stages, society may face unfavorable consequences.

1.2.3 Fake News and Rumors. Fake news is mainly intended to mislead readers, whereas a social media rumor is a piece of information that is not verified for its truthfulness at the time of posting. Zubiaga et al. [4] defined a rumor as a circulating story of questionable veracity, which is apparently credible but hard to verify and produces sufficient skepticism and/or anxiety. A rumor might be true, partially true, or false, but fake news is a deliberate lie that mimics actual news. A rumor is capable of spreading misinformation or disinformation [5, 6]. Fake news detection could be performed using similarities between fake news and rumor [7]. Many methods have been proposed for detecting rumors in social media [8, 9, 10, 11]. Typically the rumor detection problem is formulated as a classification problem, such as a binary one (rumor or non-rumor).

1.2.4 Detection Methods. Fake news detection methods using a variety of features are discussed here.

1.2.4.1 Content-based. Traditional fake news detection methods rely heavily on fake news content. In [12], they present early detection of rumors in social media based on identifying signature text phrases in social media posts, for example, "Is this true?, Really?". In [13], they propose an automatic mechanism for fake news classification using four important processes, i.e., extracting features for prediction accuracy, dataset alignment, per-set feature selection, and evaluating model transfer.

1.2.4.2 Context-based. Only news content is not adequate to enhance the existing fake news detection algorithms. This reason opens the gates for the necessity of auxiliary information, such as a user's social engagements on social media for the better detection of fake news [3]. The network structure of the news could also help identify fake news [14].

1.2.4.3 Propagation Patterns. Fake news detection could be addressed based on propagation patterns of fake news as well. Ma et al. [15] used the propagation structure technique for the rumor detection problem. They used propagation trees to identify clues on how an original message is spread over time. Ma et al. [16] automatically detected deep data representations for the enhancement of rumor detection. Their experiments focused on using variations in the contextual information of relevant posts over time for rumor detection instead of manually extracted features.

Temporal features play a crucial role in the fast-paced social media environment because information spreads more rapidly than traditional media. Many researchers have used temporal features of social media to design a model that can quickly verify news on social media. Hashimoto et al. [17] proposed a framework for rumor information detection on social media. It relies on graph structure visualization of social media messages and capturing graph topology changes over time to identify fast-spreading rumor candidates and to verify them with the reliable sources such as TV programs and newspapers to confirm their reliability.

Chang et al.'s work [18, 19] focused on buzz modeling, which means detecting a burst of topics on social media that captures the variations (i.e., sudden spikes and heavy tails) in temporal patterns of buzz time-series sequences via Product Life Cycle (PLC) models and uses a graph model K-Mixture of Product Lifecycle (K-MPLC) to detect lifecycle patterns of buzzes automatically. Buzz modeling could help prevent malicious rumor spreading.

In [20], Twitter data's temporal, structural, and linguistic properties were studied for the rumor identification problem. For temporal properties, a Periodic External Shocks (PES) model was proposed, and features introduced by this model played a big role in classifying rumors. In [21], an RNN-based model was developed for early detection of rumor circulation, which uses time-series input along with information about rumongers, and psycholinguistic traits of rumor content. Nguyen et al.'s work [22] focused on early rumor detection task by determining the credibility of each tweet using Convolutional Neural Networks and used it with a time-series based rumor classification model.

1.2.4.4 Combination of Above Methods. Kwon et al. [20] employed temporal, structural, and linguistic characteristics of rumor propagation and proposed a new periodic time series model to identify temporal features. They also identified key structural and linguistic features in the rumor propagation and achieved better performance results over the existing state of the arts on rumor classification. In [6], they explored the importance of content-based features, network-based features, and microblog-specific memes for the identification of rumors. Content-based features are extracted from text data, whereas network-based features focus on the user's behavior. Moreover, features such as hangtags and URLs extracted from microblog-specific (Twitter-specific) memes could be helpful in the enhancement of rumor detection models.

Ma et al. [23] used time-series data, in which content-based and user-based features are combined with temporal features for rumor detection problem. In [21], an RNN-based model was developed for early detection of rumor circulation, which uses time-series input along with information about rumongers and psycholinguistic traits of rumor content. Nguyen et al. [22] focused on early rumor detection task by determining the credibility of each tweet using Convolutional Neural Networks and used it with a time-series based rumor classification model.

1.3 Challenges and Proposed Solutions

This section gives the challenges and contributions of this study.

1.3.1 Problems Considered and Challenges. These are presented as follows:

- Raw datasets collected usually consist of missing values. Datasets need to be pre-processed well before a model gets trained with them; otherwise, these missing values reduce the detection performance significantly if left untreated.
- Mitigating missing values in the data is a non-trivial task. It is not possible to check whether the data contains MCAR or MNAR [24].
- In general, datasets contain a variety of data types, for example, strings and numbers. Handling missing values of numeric type is different from categorical missing values because numbers can be more easily processed than text.
- Moreover, when breaking news occurs on social media, a significant amount of information posted in the beginning stages of its propagation is unverified [25].
- It is difficult for social media users to distinguish news fake or real for rapidly spreading events where background information about an event is inadequate, and a minimal amount of time is available for verifying news truthfulness.
- Instant fake news detection techniques are required to prevent the damages that may be caused by fake news spreading.
- Detection of rumors in social media has a lot of importance among research communities because unverified information may be easily disseminated over a large network.

- If the spread of false information is not stopped early, it may cause turmoil in society. In the case of time-critical events, the effects may be dreadful.
- Most of the rumor detection models require a complex feature set, which may increase computational complexity and make training processes of detection models more difficult.
- Given the fast-paced social media environment, fast detection methods are needed to prevent rumors on social media.

1.3.2 Proposed Solutions and Contributions. These are presented next:

- This study focuses on data pre-processing methods for handling missing values in data and generation of time-series data from social media information for early detection of rumors in social media using machine learning and deep learning.
- The researcher used scikit-learn's Imputer with a "mean" strategy for handling missing values in the numerical columns, which replaces the missing values with the mean along the axis (0 - for columns, 1 - for rows) [26].
- CategoricalImputer is a new method available in sklearn-pandas[1] module for handling categorical missing values.
- The researcher combined traditional machine learning models capable of handling multi-class classification tasks with appropriate data pre-processing methods discussed in chapter 3. It is shown that the multi-layer perceptron model significantly outperforms the state-of-the-art [27].

[1] https://github.com/scikit-learn-contrib/sklearn-pandas

- This study also presents a multiple time-series data analysis model that analyzes different time-series Twitter data for early detection of fake news in social media in Chapter 4.

- Generated time-series data from raw datasets can behelpful to simplify the feature extraction process, to reduce the computational complexity of ML models, and to reduce the time required for ML models training and testing processes.

- Results show that the time-series model used with the GaussianNB classifier achieved a high Precision score.

- This study proposes a novel rumor detection method by only using the temporal features of the data for fast rumor detection in social media in Chapter 5.

- As only temporal features are used to generate time-series data, there is no need for the extraction and selection of complex features. This helps in reducing the computational time dramatically, which is critical for timely rumor detection.

- The researcher generated the time-series data in pure numeric type, which is very favorable to the classification models and can be readily inputted into a model.

- By experimenting with advanced deep learning models, the researcher improved the micro-averaged F1 score by 4.6%, compared to the baselines [28].

- This study includes a method proposed in Chapter 6, with that computational complexity can be significantly reduced as timestamps of tweets are used rather than their contents or user social engagements to perform feature extraction. Moreover, the extracted feature set is of numeric type, which is amicable to classification models.

- The proposed ensemble model improves the classification models' performances. It uses the majority-voting scheme on multiple neural networks that are part of the ensemble model and takes advantage of their strengths.

- Validation of the proposed method on the PHEME[2] dataset and the performance results demonstrate the effectiveness of the proposed scheme.

1.4 Outline of the Study

This study has seven chapters. The first chapter introduces the background knowledge on fake news detection in social media, and the second chapter provides a review of related studies. Chapter 3 discusses the mitigation of missing values in data, especially the proposed data imputation method for enhancing fake news detection. Chapter 4 explores the temporal characteristics of fake news for improving the early detection of fake news in social media using traditional machine learning models. Chapters 5 and 6 focus on the rumor detection problem by exploring the temporal properties of rumors using deep learning. Chapter 5 discusses the implementation of deep learning techniques to the specified problem, and Chapter 6 highlights an ensemble-based deep learning approach to tackle the problem. Finally, Chapter 7 contains concluding remarks and future work.

[2] https://figshare.com/articles/PHEME_dataset_for_Rumour_Detection_and_eracity_Classification/6392078

CHAPTER 2

LITERATURE REVIEW

Fake news detection attracts a massive amount of attention because of its application values. It employs ideas for detecting rumor [7] from texts for implementing fake news detection based on similarities between fake news and rumor. Machine learning, especially deep learning, is a key technique applied for fake newsdetection. Extracting and selecting useful features from data could enhance fake news detection using machine learning and deep learning. Moreover, the news' contents and the network structure of the news can be useful for identifying fake news [14].

On the other hand, rumor detection on social media itself is a well-known research topic. Some of the studies in the literature addressing the rumor detection problem are discussed next. The scheme proposed by Ma et al. [23] combined content-based and user-based features with temporal features to detect rumors. Nguyen et al. [29] focused on the early rumor detection task by determining each tweet's credibility using Convolutional Neural Networks with a time-series-based rumor classification model.

For automatic rumor identification in microblogging websites, Ma et al. [30] proposed a Dynamic Series-Time Structure (DSTS) to capture variations in social context features such as microblog contents, users, and propagation patterns over time. They discussed the strong capability of their proposed model in the early detection of rumors. Hierarchical Attention RNN (HARNN)[8] is proposed for rumor detection, which uses a Bi-GRU with attention mechanism to capture high-level representations of rumor contents and a GRU layer to capture semantic changes over the life-cycle of events. The HARNN model can select informative posts and distinctive words as features and detect rumors at an early stage.

Additionally, the rumor analysis framework [9] was proposed to clarify social media topics and visualize topic structures in time series variation as a first step and then sought help from a reliable external source to determine the topic's reliability. Rumor [10] has three general characteristics: text of an article, articles'user responses, and its source users promoting it. All these characteristics were combined for more accurate and automated fake news predictions. A Merged Neural Rumor Detection (MNRD) [11] was proposed for rumor detection in social media, which separates original posts from retweets and focused on rumor events in three aspects: original post's content, diffusion process of retweets as well as user information.

A deep attention-based model [31] was proposed for the early detection of rumors, which captures long-range dependency in the contextual variation of posting series. In [32], authors explored user-specific features and content characteristics of social media messages. They proposed an information propagation model based on heterogeneous user representation to observe distinctions in the propagation patterns of rumors and credible messages using it to differentiate them. Their study identified that rumors are more likely to spread among certain user groups. To predict a document in a social media stream to be a future rumor and stop its spread, Qin et al. [33] used content-based features, novelty-based features, and pseudo feedback.

In [34], a sentiment dictionary and a dynamic time series algorithm based Gated Recurrent Unit model was proposed that identifies fine-grained human emotional expressions of microblog events and the time distribution of social events to detect rumor events. By treating microblog users' behaviors as hidden clues to detect possible rumormongers or rumor posts, Liang et al. [35] proposed a user behavior-based rumor identification scheme. It focused on applying traditional user behavior-based features and authors' proposed new features that are extracted from users' behaviors to rumor identification task and concluded that rumor detection based on mass behaviors is better than detection based on microblogs' inherent features.

In [36], temporal, structural, and linguistic features of social media rumors were explored for the rumor classification task, and using those features helped in identifying rumors more accurately. Wu et al. [37] proposed a graph-kernel-based hybrid SVM classifier that can capture high-order (message) propagation patterns and semantic features, such as the topics of the original message for automatically detecting false rumors on Sina Weibo. However, most of the existing methods rely on a variety of features, for example, news contents, social context information, and/or complex classification model architectures to enhance fake news or rumor detection in social media. Due to this, the identification of false information on social media may be delayed since social media's fast-paced environment allows a minimal amount of time to analyze a piece of information before it propagates all over the network.

CHAPTER 3

FAKE NEWS DETECTION ENHANCEMENT WITH DATA IMPUTATION

Raw datasets collected for fake news detection usually contain some noise such as missing values. In order to improve the performance of machine learning-based fake news detection, a novel data pre-processing method is proposed in this study to process the missing values. Specifically, the missing values problem was successfully handled by using data imputation for both categorical and numerical features. For categorical features, missing values were imputed with the most frequent value in the columns. For numerical features, the mean value of the column was used to impute missing numerical values. In addition, TF-IDF vectorization was applied in feature extraction to filter out irrelevant features. Experimental results show that Multi-Layer Perceptron (MLP) classifier with the proposed data pre-processing method outperforms baselines and improves prediction accuracy by more than 15%.

In this study, the data were pre-processed by employing imputing strategies for the missing values in the dataset, where sklearn-pandas[1] categorical imputing and sklearn's Imputer[2] with mean imputing strategies were employed for categorical data and continuous data, respectively. Categorical and numerical features were handled together using the sklearn-pandas [1] DataFrameMapper method. After the data pre-processing and feature extraction phases are completed, the researcher supplied the cleaned dataset into classifiers such as Support Vector Machines, Decision Tree,

[1] https://github.com/scikit-learn-contrib/sklearn-pandas

[2] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Imputer.html

Multi-layer Perceptron, and Gradient Boosting for experimental analysis and comparison.

The contributions of this study include:

- The raw dataset has many missing values spread across multiple columns. The researcher successfully processed the missing categorical and continuous values using the categorical imputer and mean imputer.

- The researcher combined traditional machine learning models capable of handling multi-class classification tasks with appropriate data pre-processing methods and showed that the multi-layer perceptron model significantly outperformed the state-of-the-art methods [27].

The outline of this study is as follows: Section 3.1 introduces the LIAR dataset. The proposed method is discussed in Section 3.2. Experimental results and analyses are shown in Section 3.3, followed by Section 5.5, which concludes the Chapter.

3.1 LIAR Dataset

LIAR dataset[3] is a benchmark dataset for fake news detection collected from PolitiFact[4]. It includes both categorical and numerical features combined for a total of 14 columns. Columns containing categorical (text) data include statement identifier, statement, subjects discussed by the speaker, and meta-data for each speaker, such as speaker's job title, state, party, and the location of the speech. The numerical features contain the speaker's total credit history count, including the current statement, which are named as, barely true, false, half true, mostly true, and pants on fire counts [27]. The target labels consist of six classes, including *pants-fire, false, barely-true, half-true, mostly-true, and true.* This dataset is human-labeled, and each

[3] https://www.cs.ucsb.edu/~william/data/liar_dataset.zip
[4] http://www.politifact.com/

statement is evaluated by a PolitiFact editor for its truthfulness. The overall dataset contains 12,836 records in which the training set has 10,269 records and validation, and testing sets have 1,284 and 1,283 records, respectively. The training, validation, and test sets are supplied in separate files. Figure 3.1 shows some records of the LIAR dataset.

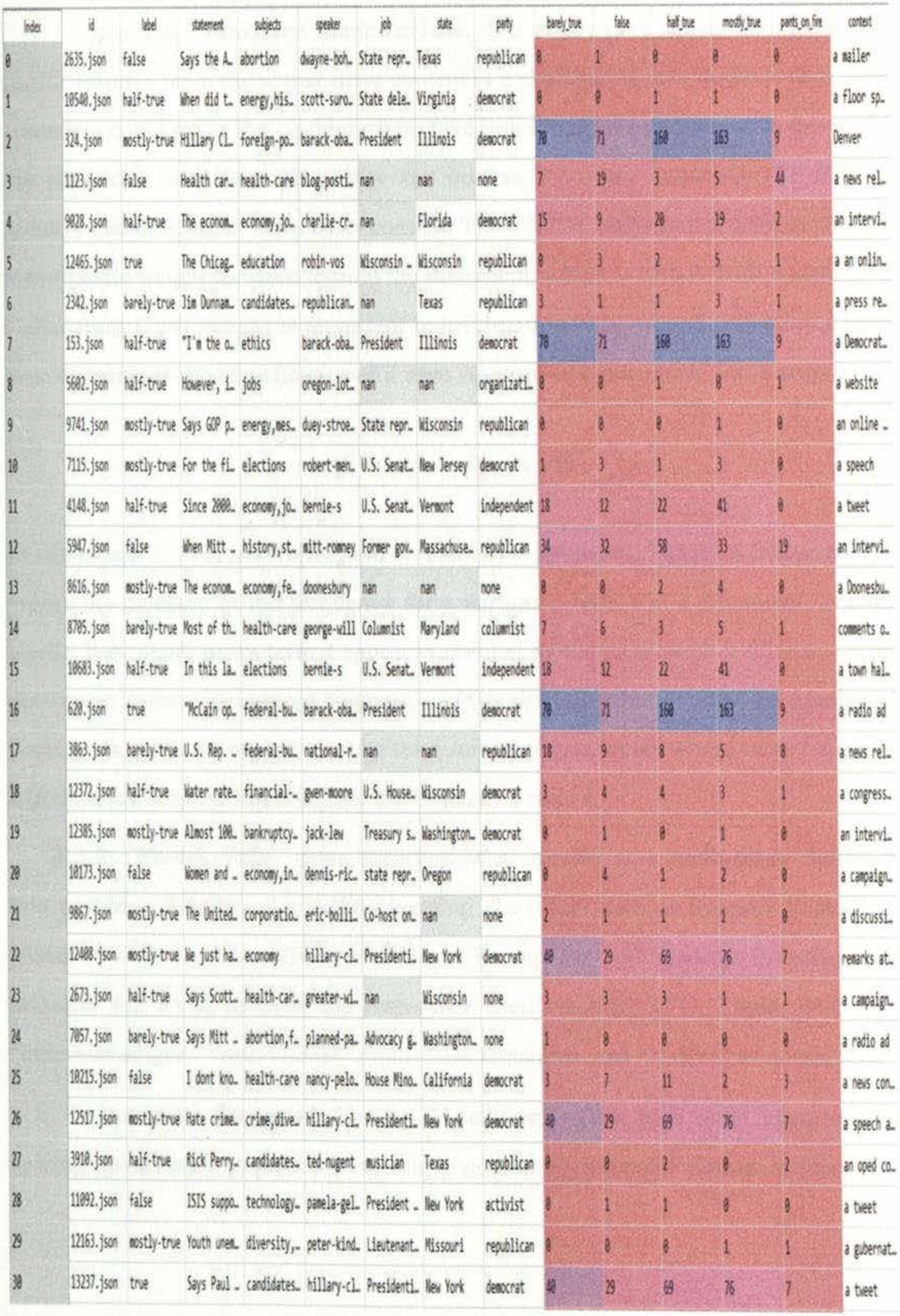
3.2 Proposed Methodology

3.2.1 Data Pre-processing. This section discusses how the researcher mitigated missing values in the LIAR dataset and performed the feature extraction.

3.2.1.1 Mitigate Missing Values. LIAR dataset[3] consists of a combination of categorical and numerical features. This dataset has many randomly located missing values for both types of features. It is not possible to check that the observed data contains missing values of Missing Completely at Random (MCAR) or Missing Not at Random (MNAR) [24]. Therefore, missing data imputation would be a good solution to handle these missing values. Typical imputation methods such as "mean" or "mode" rely on explicit model assumptions. In general, the mean is preferred for quantitative data, and mode is preferred for qualitative data [24].

In this study, the researcher used scikit-learn's Imputer with a "mean" strategy for handling missing values in the numerical columns, which replaces the missing values with the mean along the axis (0 - for columns, 1 - for rows) [26]. CategoricalImputer is a new method available in the sklearn-pandas module for handling categorical missing values. It is applied to data columns that are of type "string." It substitutes null values with the most frequent value in the column. Researchers who use the scikit-learn module cannot impute missing categorical values since scikit-learn module imputing methods are limited to numerical data. Therefore, the CategoricalImputer method helps impute missing categorical values, whereas imputing

FIGURE 3.1 A snapshot of LIAR dataset



Index	id	label	statement	subjects	speaker	job	state	party	barely_true	false	half_true	mostly_true	pants_on_fire	context
0	2635.json	false	Says the A…	abortion	dwayne-boh…	State repr…	Texas	republican	0	1	0	0	0	a mailer
1	10540.json	half-true	When did t…	energy,his…	scott-suro…	State dele…	Virginia	democrat	0	0	1	1	0	a floor sp…
2	324.json	mostly-true	Hillary Cl…	foreign-po…	barack-oba…	President	Illinois	democrat	70	71	160	163	9	Denver
3	1123.json	false	Health car…	health-care	blog-posti…	nan	nan	none	7	19	3	5	44	a news rel…
4	9028.json	half-true	The econom…	economy,jo…	charlie-cr…	nan	Florida	democrat	15	9	20	19	2	an intervi…
5	12465.json	true	The Chicag…	education	robin-vos	Wisconsin …	Wisconsin	republican	0	3	2	5	1	a an onlin…
6	2342.json	barely-true	Jim Dunnam…	candidates…	republican…	nan	Texas	republican	3	1	1	3	1	a press re…
7	153.json	half-true	"I'm the o…	ethics	barack-oba…	President	Illinois	democrat	70	71	160	163	9	a Democrat…
8	5602.json	half-true	However, i…	jobs	oregon-lot…	nan	nan	organizati…	0	0	1	0	1	a website
9	9741.json	mostly-true	Says GOP p…	energy,mes…	duey-stroe…	State repr…	Wisconsin	republican	0	0	0	1	0	an online …
10	7115.json	mostly-true	For the fi…	elections	robert-men…	U.S. Senat…	New Jersey	democrat	1	3	1	3	0	a speech
11	4148.json	half-true	Since 2000…	economy,jo…	bernie-s	U.S. Senat…	Vermont	independent	18	12	22	41	0	a tweet
12	5947.json	false	When Mitt …	history,st…	mitt-romney	Former gov…	Massachuse…	republican	34	32	58	33	19	an intervi…
13	8616.json	mostly-true	The econom…	economy,fe…	doonesbury	nan	nan	none	0	0	2	4	0	a Doonesbu…
14	8705.json	barely-true	Most of th…	health-care	george-will	Columnist	Maryland	columnist	7	6	3	5	1	comments o…
15	10683.json	half-true	In this la…	elections	bernie-s	U.S. Senat…	Vermont	independent	18	12	22	41	0	a town hal…
16	620.json	true	"McCain op…	federal-bu…	barack-oba…	President	Illinois	democrat	70	71	160	163	9	a radio ad
17	3863.json	barely-true	U.S. Rep. …	federal-bu…	national-r…	nan	nan	republican	18	9	8	5	8	a news rel…
18	12372.json	half-true	Water rate…	financial-…	gwen-moore	U.S. House…	Wisconsin	democrat	3	4	4	3	1	a congress…
19	12385.json	mostly-true	Almost 100…	bankruptcy…	jack-lew	Treasury s…	Washington…	democrat	0	1	0	1	0	an intervi…
20	10173.json	false	Women and …	economy,in…	dennis-ric…	state repr…	Oregon	republican	0	4	1	2	0	a campaign…
21	9867.json	mostly-true	The United…	corporatio…	eric-bolli…	Co-host on…	nan	none	2	1	1	1	0	a discussi…
22	12408.json	mostly-true	We just ha…	economy	hillary-cl…	Presidenti…	New York	democrat	40	29	69	76	7	remarks at…
23	2673.json	half-true	Says Scott…	health-car…	greater-wi…	nan	Wisconsin	none	3	3	3	1	1	a campaign…
24	7057.json	barely-true	Says Mitt …	abortion,f…	planned-pa…	Advocacy g…	Washington…	none	1	0	0	0	0	a radio ad
25	10215.json	false	I dont kno…	health-care	nancy-pelo…	House Mino…	California	democrat	3	7	11	2	3	a news con…
26	12517.json	mostly-true	Hate crime…	crime,dive…	hillary-cl…	Presidenti…	New York	democrat	40	29	69	76	7	a speech a…
27	3910.json	half-true	Rick Perry…	candidates…	ted-nugent	musician	Texas	republican	0	0	2	0	2	an oped co…
28	11092.json	false	ISIS suppo…	technology…	pamela-gel…	President …	New York	activist	0	1	1	0	0	a tweet
29	12163.json	mostly-true	Youth unem…	diversity,…	peter-kind…	Lieutenant…	Missouri	republican	0	0	0	1	1	a gubernat…
30	13237.json	true	Says Paul …	candidates…	hillary-cl…	Presidenti…	New York	democrat	40	29	69	76	7	a tweet

methods in the scikit-learn module could be applied to numerical data.

3.2.1.2 Feature Extraction. Wu et al. [38] stated that extracting useful features from the actual news content is a challenging task because fake news spreaders could make the content of the fake news look like real news. In this study, the researcher used term frequency and inverse document frequency (TF-IDF) to identify useful features from news content. The TF-IDF technique is used to produce a composite weight for each term in the document, which is called tf-idf weight [39]. Calculating *tf-idf* weight is important in information retrieval and text mining tasks as it determines the significance of a term or word in a document and a corpus

$$tf - idf_{t,d} = tf_{t,d} \times idf_t \tag{3.1}$$

In equation 3.1, t means a term, and d refers to a document. The term frequency $tf_{t,d}$ means the measure of the frequency for a particular term t in a document, in other words, how many times term t appeared divided by the total number of terms in the document. Inverse document frequency idf_t is the logarithm of the total number of documents in the corpus divided by the number of documents where term t appears. idf_t measure helps in knowing the importance of term t.

3.2.2 Model. Fake news detection was treated as a multi-class classification problem. Traditional machine learning classifiers such as Support Vector Machines (SVM), Decision Trees, Multi-layer Perceptron, and Gradient Boosting were selected. For SVM models, the researcher used classical SVC, Linear SVC with "crammer_singer," "one-vs-rest" multi-class strategies, and Nu-SVC as classifiers.

3.2.2.1 Support Vector Machines. They have great importance in solving classification problems consisting of nonlinearly separable classes. In this study, Support Vector Classification (SVC), Nu-Support Vector Classification (NuSVC), and Linear Support Vector Classification (LinearSVC) were used to handle multi-class classification tasks. SVC and NuSVC implement the one-vs-one scheme for multi-class classification, where the classifiers are constructed based on the number of classes presented in the dataset. NuSVC is similar to SVC, but NuSVC controls the number of support vectors and training errors using a parameter ν. LinearSVC is also similar to SVC, but the kernel used for classification is "linear." They can implement "one-vs-rest" and "crammer_singer" multi-class strategies in which the former strategy is generally preferred as the latter strategy is more expensive to compute, and better performance is rarely achieved.

3.2.2.2 Decision Tree. It is a supervised classification and regression model that relies on the decision rules derived from the data features. It could be applied to binary classification problems as well as multi-class problems. It is capable of handling both categorical and numerical data and requires little data preparation. On the other hand, sometimes, this model could create over-complex trees (i.e., overfitting). Data alteration may change the complexity of the decision tree.

3.2.2.3 Multi-layer Perceptron. It is a supervised learning algorithm that learns a function $f(\cdot): R^m \rightarrow R^o$ by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. It consists of one or more non-linear layers, called hidden layers between input and output layers. Input features are a set of neurons $\{x_i|x_1, x_2, \ldots, x_m\}$. In the hidden layer, each neuron transforms previous layers values by using a weighted linear summation $w_1x_1 + w_2x_2 + \ldots + w_mx_m$ and non-linear activation function $g(\cdot): R \rightarrow R$. Values from the last hidden layer are transformed into output values by the output layer. It is useful for on-line learning and to learn non-linear models.

3.2.2.4 ***Gradient Boosting.*** Gradient Tree Boosting is one of the ensemble-based methods. Gradient Boosting builds a forward stage-wise additive model. It could be used for both classification and regression problems. In this model, heterogeneous features are naturally handled, but scalability is an issue because of the sequential nature of boosting.

3.3 Experiment

The researcher employed LIAR dataset [3] to verify ML models. Four evaluation metrics, namely accuracy, precision, F1-score, and recall, are used to evaluate ML models' performance. One of the major challenges of performing classification on this dataset was to handle missing values. To mitigate this problem, the researcher applied three data pre-processing methods on the dataset and examined how effectively each method could impact the classifiers' performances. Feature extraction was performed on the dataset for all three methods as discussed in Section 3.2.1.2, and the researcher utilized all features except *statement id* for the analysis.

Additionally, the models' computational complexity was examined by monitoring the training and prediction time for different classifiers. They are presented in hours, minutes, seconds, and milliseconds (HH:MM:SS:ms) format. The three methods used are as follows:

3.3.1 Delete Records Containing Missing Values. In this method, records consisting of missing values were deleted. This method removed more than 4,000 records from the dataset. MLP classifier outperformed other classifiers in predicting validation and test sets. The performances are shown in Tables 3.1, 3.2, and 3.3.

3.3.2 Replace Missing Values with Empty Text. The researcher used *empty* text to replace the missing values in the dataset. With this method, the

TABLE 3.1 The training time of different classifiers with the delete method

Classifier	Training Time (HH:MM:SS:ms)
SVC	0:21:33.292383
LinearSVC_CS	0:03:11.075396
LinearSVC_OVR	0:00:09.173900
NuSVC	0:13:13.883099
DecisionTree	0:00:06.170634
MLPClassifier	1:40:09.743315
GradientBoosting	0:21:42.929440

TABLE 3.2 Performance results on the validation set with the delete method

Classifier	Prediction Time (HH:MM:SS:ms)	Accuracy %	F1	Precision	Recall
SVC	0:01:56.706200	0.283	0.182	0.217	0.234
LinearSVC_CS	0:00:00.634472	0.174	0.173	0.179	0.181
LinearSVC_OVR	0:00:00.021058	0.265	0.228	0.258	0.237
NuSVC	0:01:29.970123	0.258	0.240	0.255	0.244
DecisionTree	0:00:00.047126	0.326	0.324	0.328	0.322
MLPClassifier	0:00:00.105282	0.416	0.370	0.515	0.370
GradientBoosting	0:00:00.071388	0.400	0.377	0.441	0.368

TABLE 3.3 Performance results on the test set with the delete method

Classifier	Prediction Time (HH:MM:SS:ms)	Accuracy %	F1	Precision	Recall
SVC	0:01:58.420455	0.286	0.174	0.179	0.230
LinearSVC_CS	0:00:00.040109	0.173	0.166	0.172	0.174
LinearSVC_OVR	0:00:00.025069	0.225	0.210	0.224	0.217
NuSVC	0:01:26.568999	0.247	0.229	0.239	0.238
DecisionTree	0:00:00.044117	0.339	0.343	0.338	0.351
MLPClassifier	0:00:00.079210	0.394	0.359	0.515	0.356
GradientBoosting	0:00:00.086864	0.390	0.391	0.438	0.381

researcher successfully prevented the data loss problem because no records were deleted. Again, the MLP classifier stood at the top in the list in terms of performance. This time the prediction accuracies for validation and test sets were improved compared to the delete method. Tables 3.4, 3.5, and 3.6 show the respective performance results for validation and test sets with the replace method.

TABLE 3.4 The training time of different classifiers with the replace method

Classifier	Training Time (HH:MM:SS:ms)
SVC	1:38:25.423104
LinearSVC_CS	0:06:32.412224
LinearSVC_OVR	0:00:15.322742
NuSVC	1:05:17.864797
DecisionTree	0:00:24.834177
MLPClassifier	0:42:30.996866
GradientBoosting	1:12:06.434310

TABLE 3.5 Performance results on the validation set with the replace method

Classifier	Prediction Time (HH:MM:SS:ms)	Accuracy %	F1	Precision	Recall
SVC	0:06:49.043784	0.243	0.198	0.291	0.230
LinearSVC_CS	0:00:00.083121	0.191	0.183	0.183	0.183
LinearSVC_OVR	0:00:00.066884	0.280	0.273	0.294	0.277
NuSVC	0:06:25.410218	0.354	0.347	0.363	0.342
DecisionTree	0:00:00.110789	0.393	0.391	0.394	0.389
MLPClassifier	0:00:00.474289	0.458	0.454	0.553	0.443
GradientBoosting	0:00:00.157420	0.446	0.441	0.486	0.432

3.3.3 Impute Missing values Using Data Imputation Techniques. In this method, the researcher evaluated the data pre-processing method as discussed in Section 3.2.1.1, using different machine learning classifiers on validation and test datasets after these models were trained successfully. Tables 3.7, 3.8, and 3.9 show the performance results on the validation set and test set, respectively. It is observed that the classifiers with data imputation outperformed those with the delete method in Section 3.3.1. Moreover, replace and data imputation methods achieved almost similar performance results. With the delete method, examples were obtained by eliminating records with any missing values, which reduces the actual dataset size and causes information loss. This method is suggested only for large datasets with a small percentage of missing values occurrence, and analysis of the complete examples should not make the dataset seriously biased [40]. On the other hand, with replace and data imputation methods, the problem of data loss was eliminated. For the replace method, the researcher considered missing values as blank values and treated them in the same way as other values. Data imputation methods are simple and

TABLE 3.6 Performance results on the test set with the replace method

Classifier	Prediction Time (HH:MM:SS:ms)	Accuracy %	F1	Precision	Recall
SVC	0:07:35.188419	0.248	0.188	0.254	0.224
LinearSVC_CS	0:00:00.084752	0.184	0.174	0.176	0.175
LinearSVC_OVR	0:00:00.061115	0.256	0.242	0.258	0.249
NuSVC	0:06:59.119033	0.353	0.341	0.351	0.337
DecisionTree	0:00:00.116335	0.370	0.381	0.379	0.384
MLPClassifier	0:00:00.502584	0.434	0.434	0.533	0.434
GradientBoosting	0:00:00.205547	0.426	0.432	0.465	0.426

effective solutions when the missing values problem caused by missing at random (MAR) mechanism, which is the case here [40].

Compared to the state-of-the-art methods [27], the proposed method for data preprocessing and MLP Classifier has significantly improved the accuracy of the validation set and test set by 21% and 16%, respectively. Training iterations are limited to 200 with a fixed random state value, and the researcher employed *stochastic gradient descent* to optimize the MLP classifier. Gradient Boosting, Decision Tree, and NuSVC classifiers also achieved satisfactory performances where Decision Tree Classifier consumed less time for training. It was also observed that classifiers, including SVC, LinearSVC with "crammer_singer" and "one-vs-rest" strategies performed poorly and achieved fewer accuracy scores since the dimensionality of the feature is high. Additionally, the researcher measured the total time consumed for the prediction on both validation and test sets as well as some other metrics, such as F1−Score, Precision, and Recall.

The researcher ran the MLP Classifier with the proposed methods for ten rounds to observe its performance without using random state value. The number of it-

TABLE 3.7 The training time of different classifiers with the data imputation method

Classifier	Training Time (HH:MM:SS:ms)
SVC	1:37:29.514189
LinearSVC_CS	0:07:52.727027
LinearSVC_OVR	0:00:20.665244
NuSVC	1:32:44.329309
DecisionTree	0:00:12.401620
MLPClassifier	0:48:19.175377
GradientBoosting	1:02:10.105386

TABLE 3.8 Performance results on the validation set with the data imputation method

Classifier	Prediction Time (HH:MM:SS:ms)	Accuracy %	F1	Precision	Recall
SVC	0:08:10.334100	0.245	0.200	0.293	0.232
LinearSVC_CS	0:00:00.086328	0.195	0.190	0.189	0.191
LinearSVC_OVR	0:00:00.150023	0.267	0.264	0.274	0.272
NuSVC	0:07:54.447672	0.367	0.359	0.394	0.349
DecisionTree	0:00:00.075579	0.394	0.395	0.400	0.393
MLPClassifier	0:00:00.491813	0.457	0.455	0.504	0.444
GradientBoosting	0:00:00.107796	0.442	0.437	0.484	0.428

TABLE 3.9 Performance results on the test set with the data imputation method

Classifier	Prediction Time (HH:MM:SS:ms)	Accuracy %	F1	Precision	Recall
SVC	0:08:15.804666	0.248	0.188	0.254	0.224
LinearSVC_CS	0:00:00.088085	0.178	0.170	0.171	0.171
LinearSVC_OVR	0:00:00.209324	0.239	0.231	0.238	0.244
NuSVC	0:07:37.068152	0.360	0.342	0.366	0.338
DecisionTree	0:00:00.078278	0.381	0.391	0.386	0.397
MLPClassifier	0:00:00.462904	0.436	0.440	0.492	0.435
GradientBoosting	0:00:00.104769	0.426	0.432	0.463	0.426

erations is limited to 300 for the MLP classifier. Table 3.10 lists the results for the training set. It shows that the MLP classifier combined with proposed data pre-processing method is stable by maintaining training loss consistency.

Figure 3.2 gives the training loss curves versus the number of iterations. Tables 3.11 and 3.12 show the details of the MLP Classifier performance for ten rounds on validation and test sets. It is observed that the training loss curves for all the 10 rounds are consistent with the average final loss value of 1.279.

3.4 Concluding Remarks for the Chapter

In this study, a data imputation pre-processing method was proposed for enhancing machine learning-based fake news detection. The proposed method focused on how to process the missing values in the raw data using data imputation techniques. Experimental results showed that machine learning models combined with the proposed data pre-processing method outperformed baselines.

TABLE 3.10 Performance of the MLP classifier

Round	Training Time (HH:MM:SS:ms)	Training Loss	No. of Iterations
1	0:39:48.745564	1.303	127
2	0:49:54.880182	1.280	154
3	0:47:42.052116	1.286	148
4	0:55:40.648856	1.270	171
5	1:04:42.601990	1.265	177
6	1:16:18.708549	1.254	197
7	0:37:16.882210	1.322	116
8	0:48:07.396703	1.265	175
9	0:48:46.390700	1.266	177
10	0:41:38.946055	1.282	152

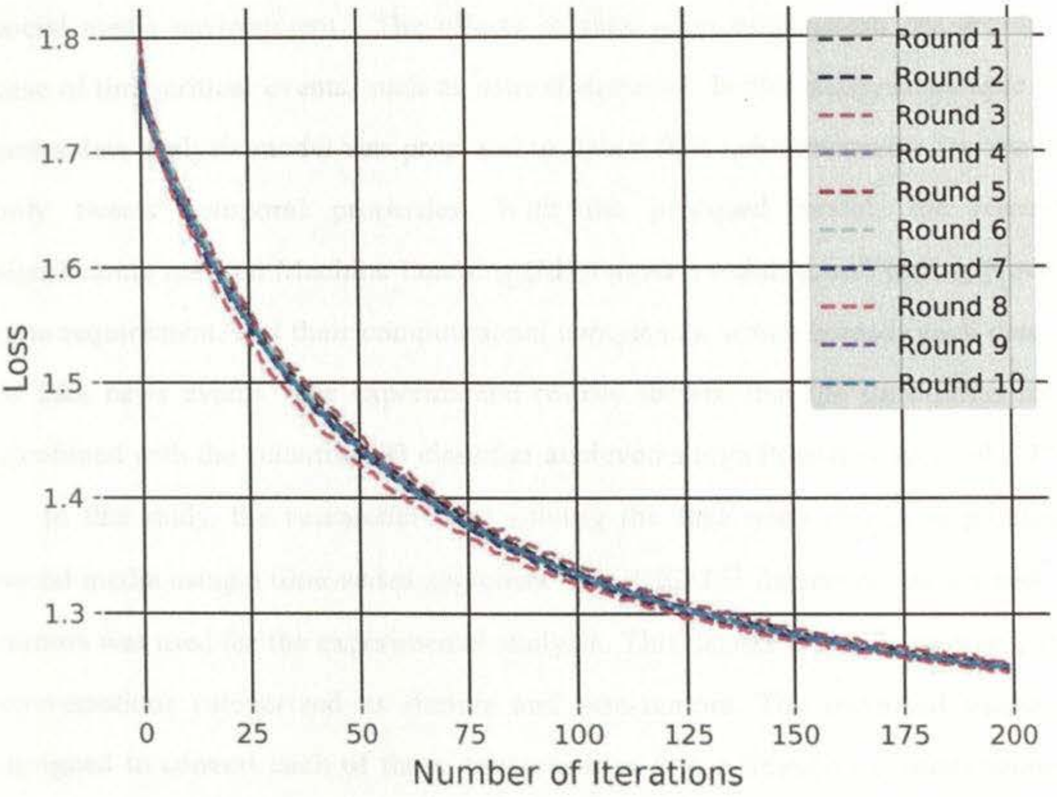
TABLE 3.11 MLP classifier performance results on the validation set

Prediction Time (HH:MM:SS:ms)	Accuracy %	F1	Precision	Recall
0:00:00.480897	0.465	0.454	0.574	0.446
0:00:00.357233	0.470	0.462	0.577	0.454
0:00:00.385952	0.453	0.446	0.571	0.438
0:00:00.481820	0.451	0.445	0.570	0.437
0:00:00.469991	0.469	0.458	0.577	0.452
0:00:00.395029	0.467	0.458	0.557	0.449
0:00:00.334350	0.450	0.449	0.486	0.439
0:00:00.434091	0.461	0.453	0.576	0.447
0:00:00.345711	0.459	0.453	0.555	0.443
0:00:00.355381	0.462	0.457	0.560	0.448

TABLE 3.12 MLP classifier performance results on the test set

Prediction Time (HH:MM:SS:ms)	Accuracy %	F1	Precision	Recall
0:00:00.453560	0.443	0.439	0.532	0.439
0:00:00.437473	0.430	0.429	0.533	0.428
0:00:00.355367	0.449	0.444	0.551	0.445
0:00:00.470456	0.443	0.441	0.548	0.441
0:00:00.622739	0.449	0.443	0.550	0.443
0:00:00.294496	0.445	0.441	0.538	0.443
0:00:00.443745	0.448	0.455	0.483	0.447
0:00:00.251151	0.436	0.433	0.540	0.433
0:00:00.331610	0.444	0.440	0.539	0.441
0:00:00.283757	0.443	0.438	0.539	0.441

FIGURE 3.2 Training loss curves of the MLP classifier



CHAPTER 4

MULTIPLE TIME-SERIES DATA ANALYSIS FOR FAKE NEWS DETECTION IN SOCIAL MEDIA

Fake news detection is a big problem in the fast-paced information spreading social media environment. The effects of fake news propagation are dreadful in case of time-critical events, such as natural disasters. In this study, a multiple time-series data analysis model was proposed to detect fake news events on Twitter using only tweets' temporal properties. With the proposed model, the researcher significantly reduced Machine Learning (ML) models training, and testing processes time requirement, and their computational complexity, which helped quick detection of fake news events. The experimental results showed that the time-series model, combined with the GaussianNB classifier achieved a high Precision score of 94%.

In this study, the researcher tried solving the fake news detection problem in social media using a time-series approach. The PHEME[1] dataset of rumors and non-rumors was used for the experimental analysis. This dataset is a collection of Twitter conversations categorized as rumors and non-rumors. The proposed model was designed to convert each of these conversations into a time-series vector using the timestamps of each tweet in a conversation. The time-series vector representation of a Twitter conversation consists of reaction (tweets) counts corresponding to the source tweet for each time interval from the beginning to the end of the conversation. With this time-series approach, data preprocessing time was reduced. Using the generated time-series data with selected machine models, ML models training time and their

[1] https://figshare.com/articles/PHEME_dataset_of_rumours_and_non-rumours/4010619

computational complexity was reduced by taking advantage of the numeric data type. A 94% Precision score with Gaussian Naive Bayes (GaussianNB) Classifier was obtained.

The outline of this study is as follows: In Section 6.3, a fake news detection problem in the time-series domain was presented. An introduction to the dataset and ML models used in the experimental analysis is provided. Section 4.2 contains the experimental results. Finally, Section 6.6 concludes the Chapter.

4.1 Methodology

4.1.1 Problem Definition. Information spreads quickly on social media, especially news that can capture social media users' attention more likely to propagate faster than ordinary news. In the case of breaking news on social media, a significant amount of information posted in the beginning stages of its propagation is unverified [25]. It is difficult for social media users to distinguish news fake or real for rapidly spreading events, where background information about an event is inadequate, and a minimal amount of time is available for verifying news truthfulness. Instant fake news detection techniques are required to prevent the damages that may be caused by fake news. The researcher's approach to using time-series data for fake news detection is quick in flagging news as fake or true. The data is all numeric, and any information is discarded about the news except the news creation time, which reduces the time required for the ML model training process.

4.1.2 Data Preparation and Analysis. This section introduces the PHEME dataset, and discusses how data pre-processing is performed on the dataset. At last, gives a brief analysis about the dataset.

4.1.2.1 PHEME Dataset. This dataset has a collection of Twitter rumors and non-rumors posted during five breaking news: Charlie Hebdo shooting, Ferguson unrest, Germanwings plane crash, Ottawa shooting, and Sydney siege. The actual dataset consists of a directory for each event, including subfolders for rumor and non-rumor conversation samples. A source-tweet and its correspondingreactions (a set of tweets) are provided for each conversation sample. The dataset was collected using the Twitter streaming API and annotated by a team of expert journalists [5]. The overall dataset consists of 5, 802 annotated tweets for all the five events in which 1, 972 are rumors, and 3, 830 are non-rumors.

4.1.2.2 Data Pre-processing. The researcher used the scikit-learn machine learning library for experiments. The experimental setup was done in Python 3.5.5 using Spyder IDE 3.2.8, and NVIDIA GPU Server. Time-series data were prepared for experiments in five different time intervals: 2, 5, 10, 30, and 60 minutes for all five events present in the PHEME dataset. Preprocessed data contains a numerical vector representation of Twitter conversations in which each row represents one whole conversation. Its columns having total reactions count for a given time interval limit.

The PHEME dataset consists of a set of events $E = \{E_i\}$. Each event E_i consists of rumor and non-rumor Twitter conversations $\{c_{ij}\}$. For each event, the researcher prepared the time-series data for chosen time intervals by iteratingthrough each event's directory and its corresponding rumor and non-rumor subfolders. The number of time intervals $N(c_{ij})$ for a conversation c_{ij} is given by,

$$N(c_{ij}) = \frac{\max timeReactions_{ij} - timeSource_{ij}}{T} \tag{4.1}$$

where $timeSource_{ij}$ is the timestamp of the source tweet, the timestamps of all the reactions corresponding to that source tweet is $timeReactions_{ij} = \{tr_1, tr_2, \cdots, tr_n\}$, and T is a tunable parameter for setting the desired time interval limit. In the experiments, the researcher set the values for $T = 2, 5, 10, 30, 60$ minutes.

For every conversation in both the subfolders, the following steps were executed to generate time-series data:

- Step 1: Read the source tweet and get the originating timestamp.
- Step 2: Read all the reactions corresponding to that source tweet and get their tweet creation timestamps.
- Step 3: For each time interval $t_{ij,k} = [a, b]$, the count of the total number of timestamps of reactions in that interval is given by,

$$count_{t_{ij,k}} = \mathbf{card}(Q) \tag{4.2}$$

where $t_{ij,k}$ is the k-th time interval of a conversation c_{ij}, k has values from $1, 2, \cdots, N$, and $Q \subset timeReactions_{ij}$, which is given by $Q = \{x \mid x > a \wedge x \leq b\}$, and x is the timestamp for a reaction.

Therefore, the vector sequence of a conversation c_{ij} belonging to an event E_i is,

$$V(c_{ij}) = [count_{t_{ij,1}} \quad count_{t_{ij,2}} \quad \cdots \quad count_{t_{ij,N}}] \tag{4.3}$$

and E_i has a feature vector as follows:

$$E_i = \begin{bmatrix} V(c_{i0}) \\ V(c_{i1}) \\ \vdots \\ V(c_{in}) \end{bmatrix} \tag{4.4}$$

The researcher structured the feature vector for an event by filling tailing null values with $0s$, and for the analysis, labeled non-rumors as $0s$ and $1s$ for rumor samples.

4.1.2.3 Dataset Analysis. Table 4.1 shows the number of rumors and non-rumors present in the PHEME dataset for all the five events. Germanwings Crash and Ottawa Shooting events have slightly more than half of the tweets as rumors. Charlie Hebdo and Ferguson events have a high percentage of non-rumors. Sydney Siege event has 57.2% of non-rumors. In total, this dataset has 66% of non-rumor samples among 5,802 tweets.

TABLE 4.1 PHEME dataset of rumors and non-rumors

Event	Rumors	Non-rumors	Total Count
Charlie Hebdo	458 (22.0%)	1,621 (78.0%)	2,079
Ferguson	284 (24.8%)	859 (75.2%)	1,143
Germanwings Crash	238 (50.7%)	231 (49.3%)	469
Ottawa Shooting	470 (52.8%)	420 (47.2%)	890
Sydney Siege	522 (42.8%)	699 (57.2%)	1,221
Total Count	**1, 972 (34.0%)**	**3,830 (66.0%)**	**5,802**

The idea is to explore any distinction patterns in the propagation of rumors and non-rumors using time-series data. I plotted Twitter interactions (i.e., number of rumor and non-rumor tweets) corresponding to each event in the PHEME dataset for all the selected time intervals. Figures 4.1, 4.2, 4.3, 4.4, and 4.5 shows propagation patterns of rumors and non-rumors for different time intervals. From these time-series plots, I observed that with bigger time intervals such as 10, 30, 60 minutes, the difference in propagation patterns of rumors and non-rumors is easily identified for the events Charlie Hebdo, Ferguson, Ottawa Shooting, and Sydney Siege. The only exception is event Germanwings Crash, where the propagation patterns are almost identical for all time intervals. For events, Charlie Hebdo, Ferguson, and Sydney Siege have longer non-rumor spikes.

4.1.3 Machine Learning (ML) Models. The researcher have handpicked eight machine learning models for the experimental analysis: Birch, Decision Tree, Gaussian Na ̈ıve Bayes, KMeans, Logistic Regression, Multi-Layer Perceptron, Random Forest, and Support Vector Machine. All of the models are implemented using scikit-learn Machine Learning in Python package [26].

4.1.3.1 Birch. Birch is a memory-efficient clustering model that constructs a tree data structure called Characteristic Feature Tree (CFT) in which each node has several Characteristic Feature (CF) subclusters. These subclusters help in memory management while handling input data by maintaining necessary information for clustering such as Linear Sum, Squared Sum, Centroids, Squared norm of centroids, and the number of samples in a subcluster. The branching factor decides the maximum number of subclusters in a node, whereas the distance between the existing subclusters and the entering sample is controlled using the threshold parameter.

FIGURE 4.1 Event Charlie Hebdo propagation patterns for different time intervals

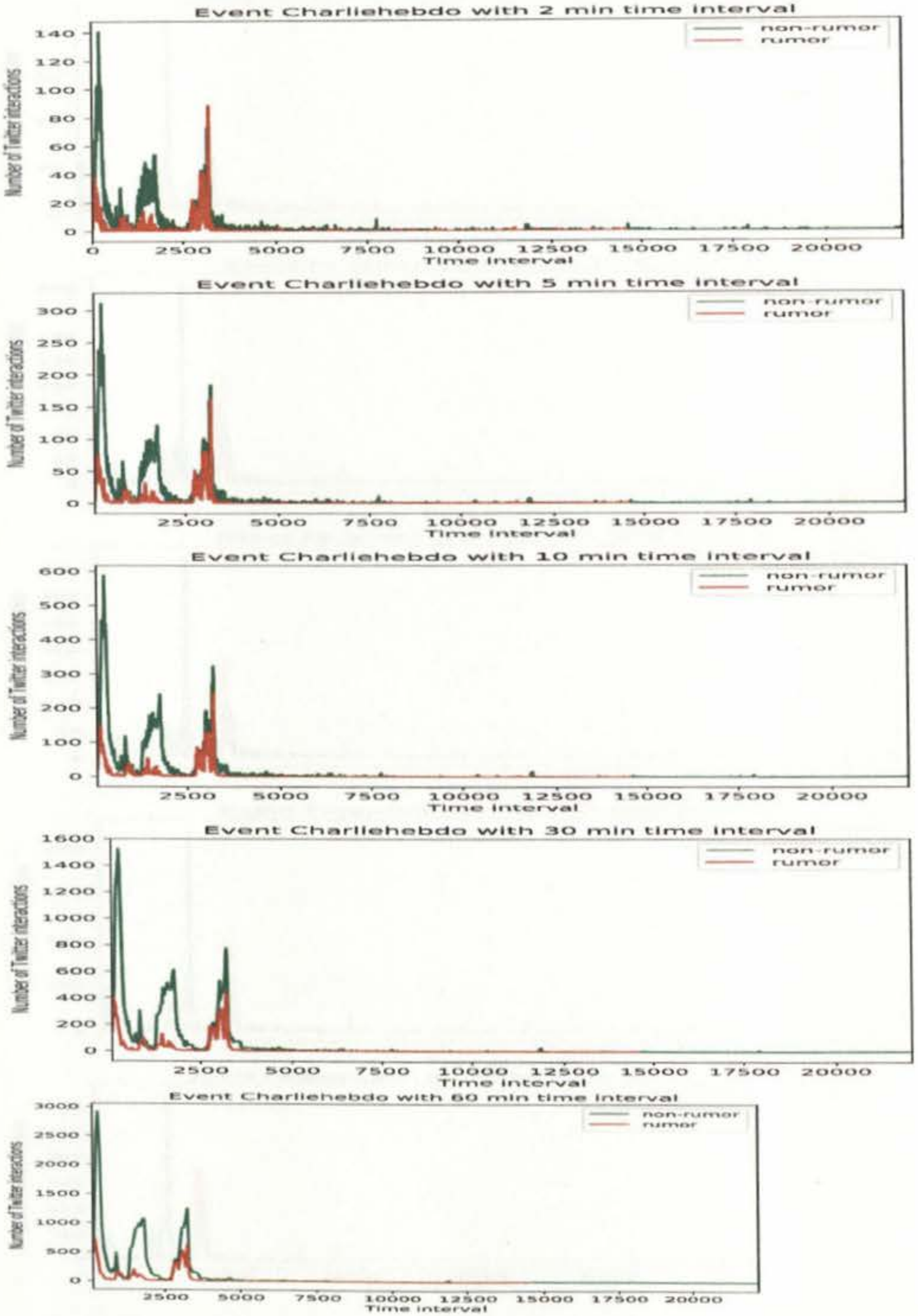


FIGURE 4.2 Event Ferguson propagation patterns for different time intervals

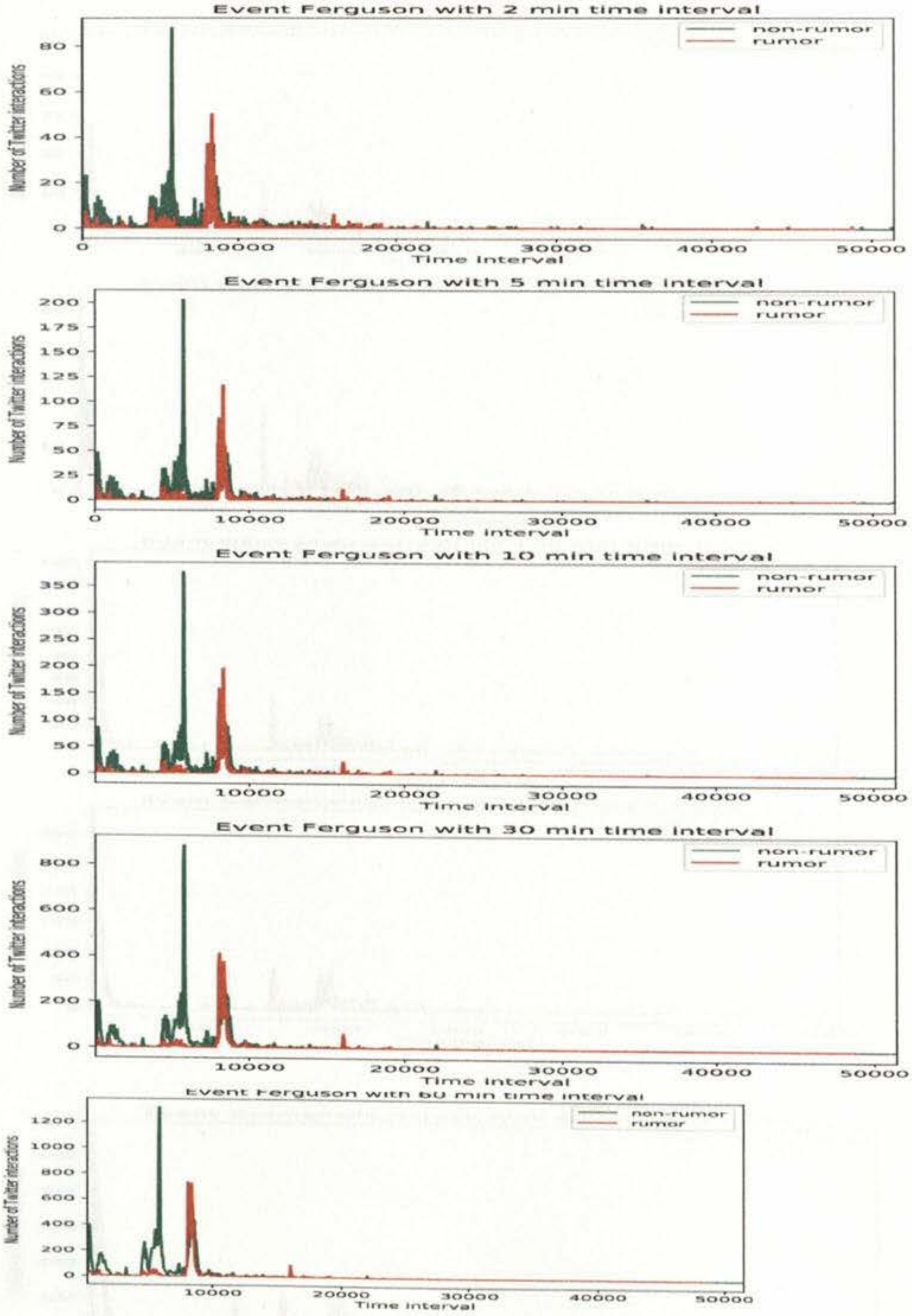


FIGURE 4.3 Event Germanwings Crash propagation patterns for different time intervals

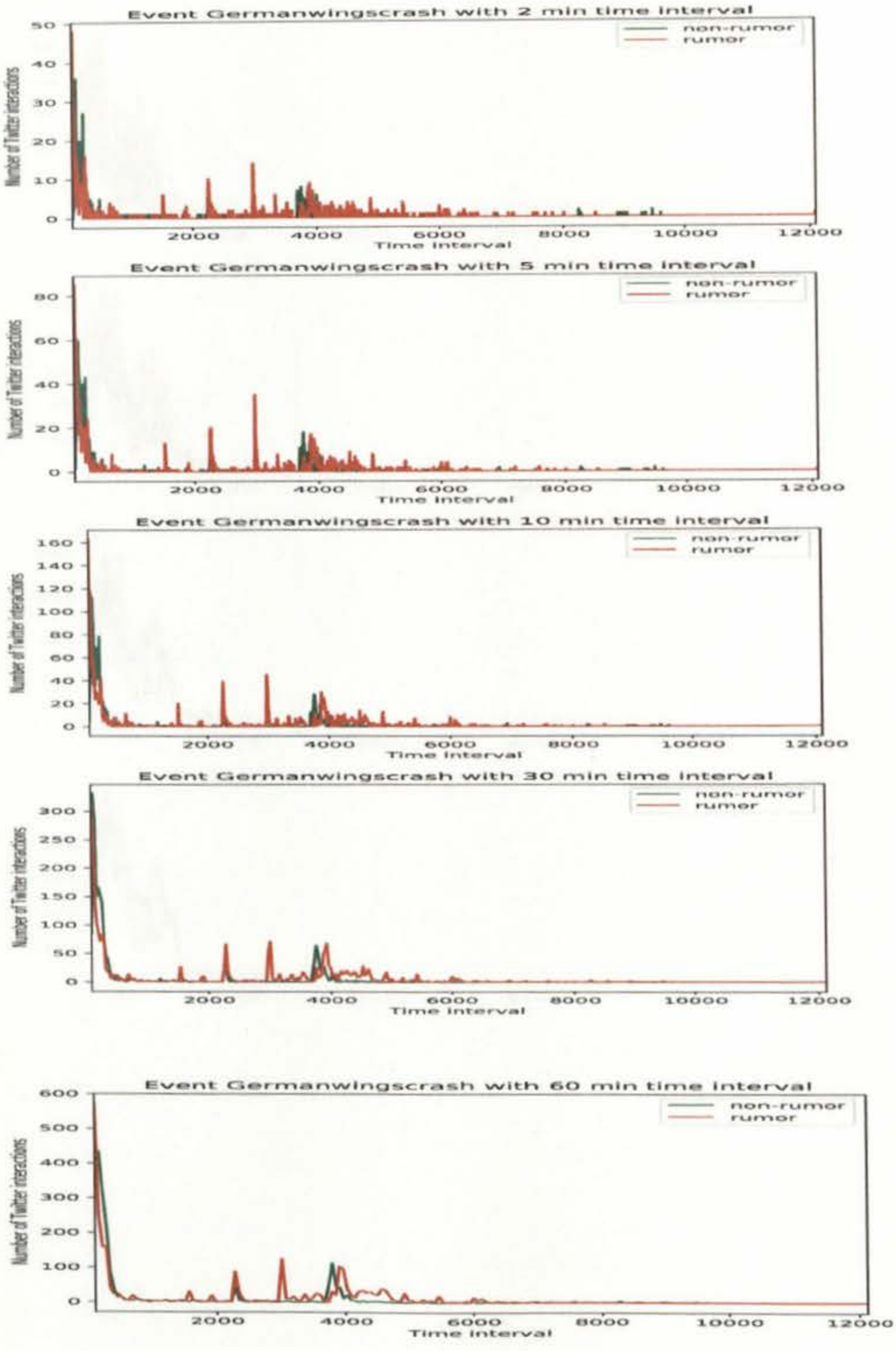


FIGURE 4.4 Event Ottawa Shooting propagation patterns for different time intervals

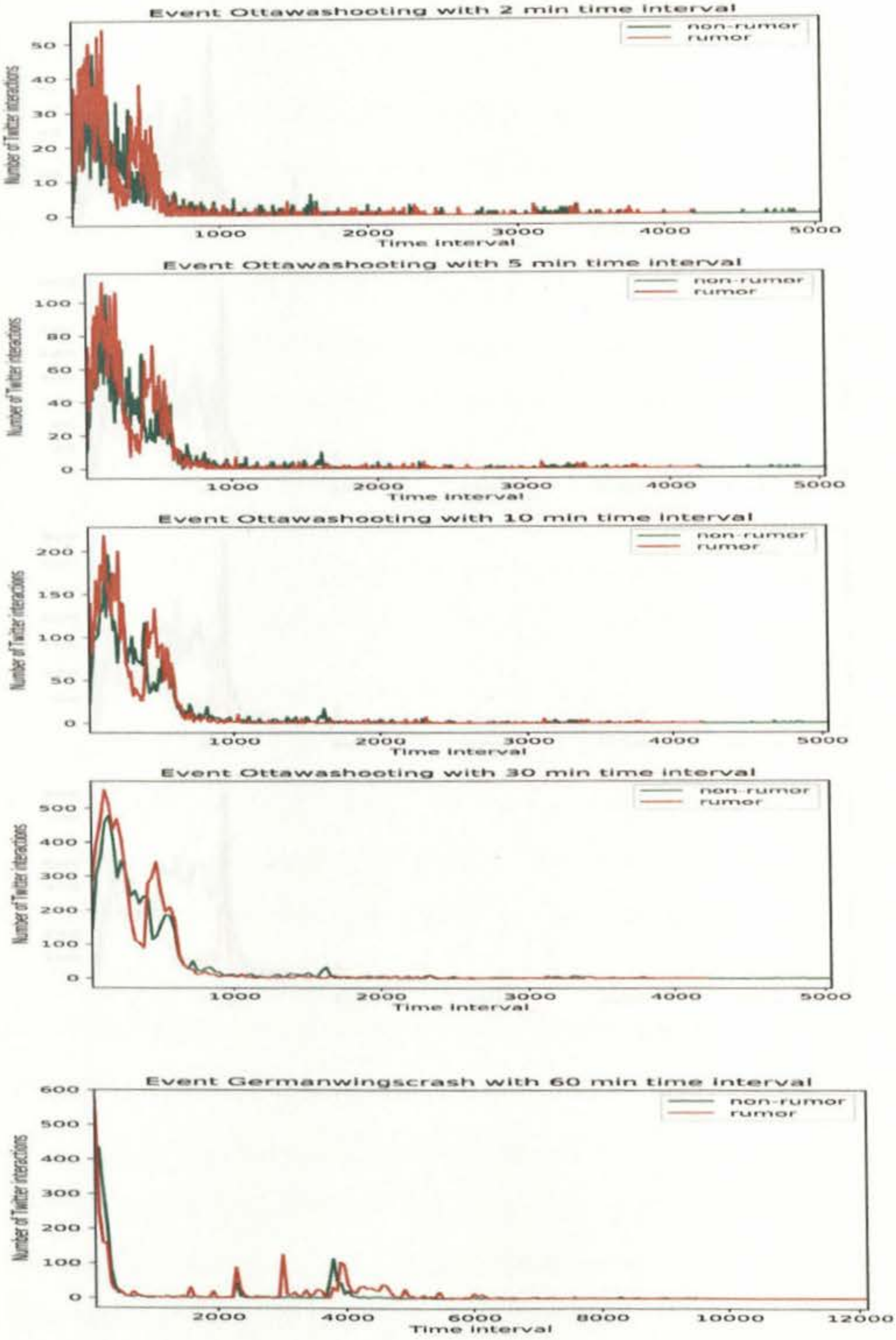
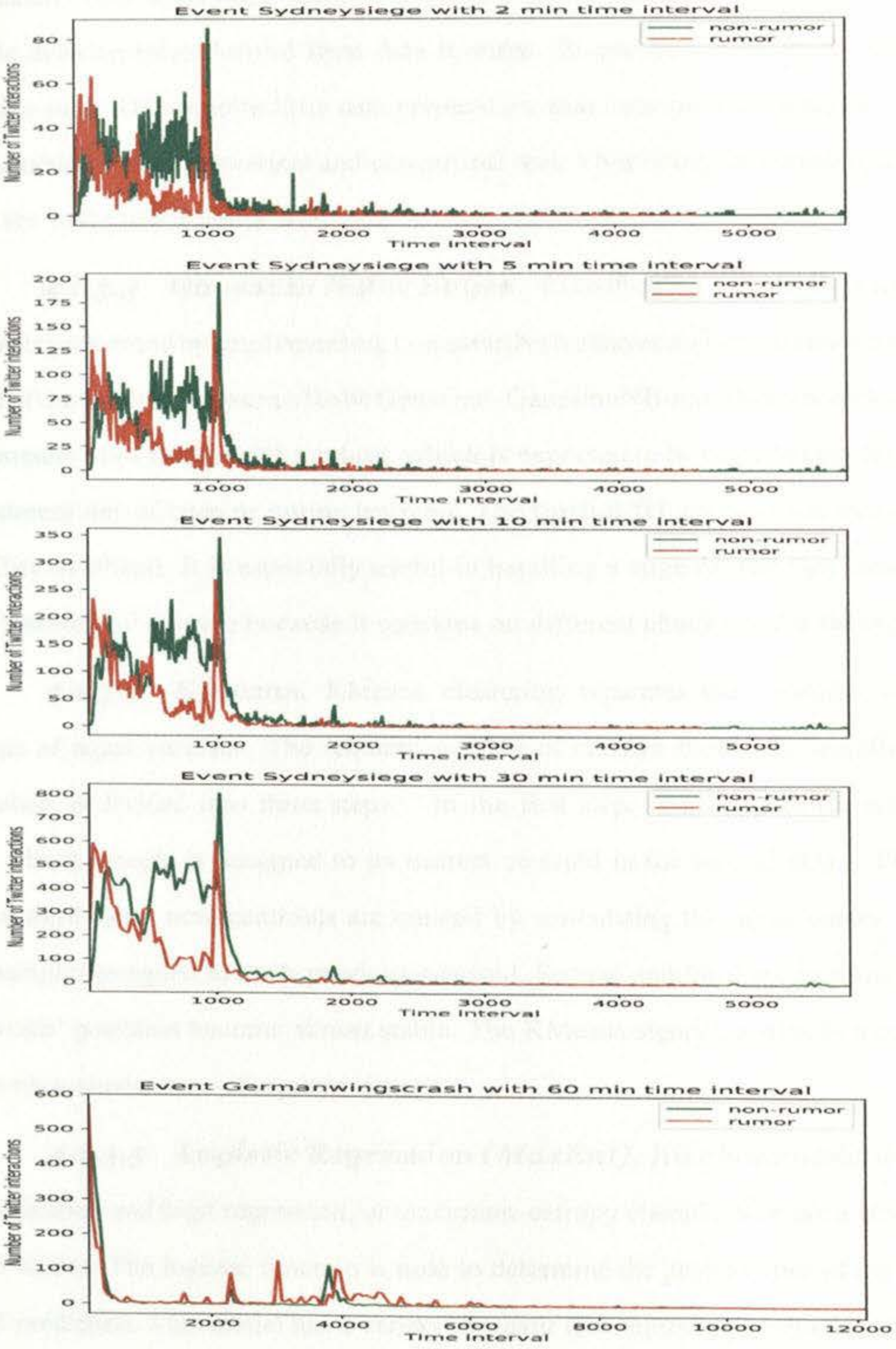


FIGURE 4.5 Event Sydney Siege propagation patterns for different time intervals



4.1.3.2 Decision Trees (DTs). DTs are used for classification and regression. This supervised learning method predicts the target label by learning simple decision rules derived from data features. Deeper trees have more complex decision rules. DTs require little data preparation, and trees can be visualized. They are suitable for both numerical and categorical data. Overfitting and creating biased trees are well-known issues with DTs.

4.1.3.3 Gaussian Naive Bayes. Classification with the GaussianNB model is performed by implementing Gaussian Na¨ıve Bayes algorithm and the likelihood of the features is assumed to be Gaussian. GaussianNB classifier updates model parameters via a partial fit method, which is expected to be called many times to implement out-of-core or online learning. The partial_fit method has numerical stability overhead. It is especially useful in handling a huge dataset that cannot fit into memory all at once because it operates on different chunks of the dataset.

4.1.3.4 KMeans. KMeans clustering separates data samples into n groups of equal variance. The required number of clusters should be specified. Its operation is divided into three steps. In the first step, initial centroids are chosen. Each sample is assigned to its nearest centroid in the second step. Finally, in the third step, new centroids are created by calculating the mean values of all the samples assigned to each previous centroid. Second and third are repeated until centroids' positions become almost stable. The KMeans algorithm tries to minimize the within-cluster sum-of-squares criterion.

4.1.3.5 Logistic Regression (MaxEnt). It is a linear model used for classification and logit regression, or maximum-entropy classification are a few of its other names. The logistic function is used to determine the probabilities of the target label prediction. This model has a variety of solver techniques applicable todifferent cases such as L1 penalty, Multinomial loss, and large datasets.

4.1.3.6 Multi-layer Perceptron (MLP).

MLP is a supervised learning algorithm that consists of one or more hidden layers between input and output layers. In the hidden layer, each neuron updates the previous layer's values with a weighted linear summation, followed by a non-linear activation function. It can learn non-linear models as well as real-time models. MLP validation accuracy is dependent on weight initializations; it is sensitive to feature scaling, and hyperparameter tuning is also required.

4.1.3.7 Random Forests.

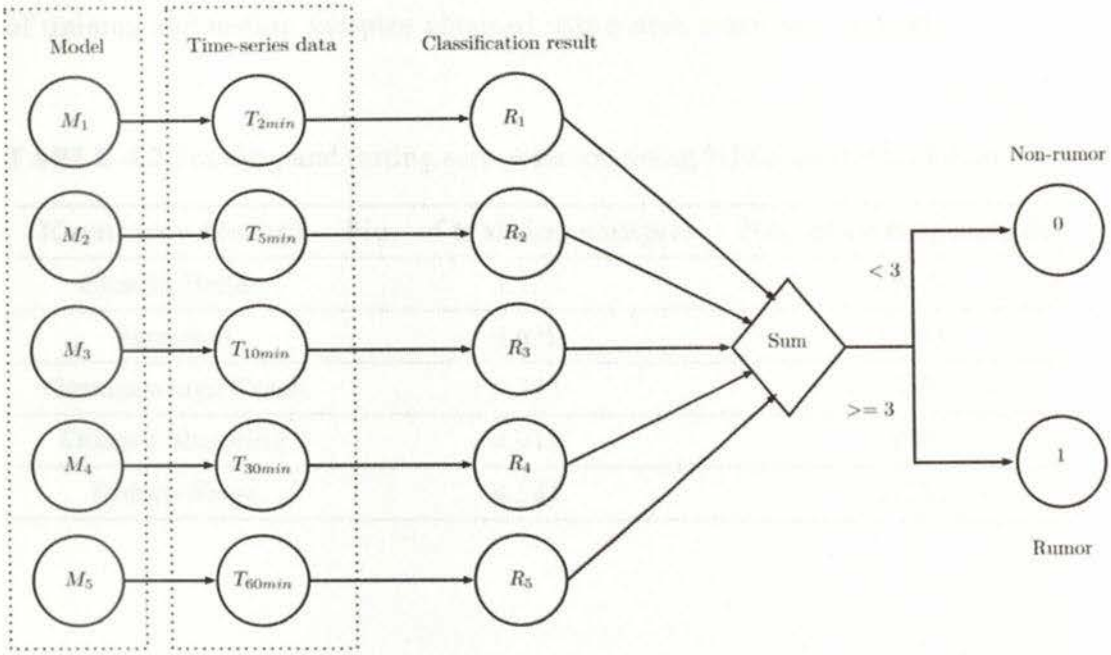
Random Forest Classifier is an ensemble-based method used for classification, anomaly detection, and regression problems. In random forests, bootstrap samples are drawn from the training set to build each tree in the ensemble. While constructing a tree, a node is split by picking the best split among a random subset of features. Due to this random selection of split, forest bias slightly increases, which is compensated by reducing variance by averaging.

4.1.3.8 Support Vector Machines (SVMs).

SVMs are supervised learning models applied to outlier detection, classification, and regression problems. SVMs decision function is dependent on support vectors, which are a subset of training data. C-Support Vector Classification (SVC) is based on libsvm implementation. SVC can be implemented with different kernel functions such as linear, polynomial, rbf, and sigmoid. Due to the complexity in fit time, it is difficult to scale large datasets.

For each of these classification models, the researcher instantiated five different sub-models (i.e., five instances per model) with respect to all of the five time-interval time-series data, as shown in Figure 4.6. Each sub-model gets trained with its corresponding time-series data and gives its prediction result. Once all the sub-

FIGURE 4.6 Multiple time-series data analysis model



model predictions are obtained, majority voting was performed to decide the final prediction result and used it for calculating the ML model's evaluation metrics. In this work, a non-rumor is 0, and a rumor is 1. In the majority voting process, if the overall sum of all the sub-model predictions is less than three, then the final prediction is considered as non-rumor. Otherwise, it is a rumor.

4.2 Experimental Results

The experimental results are provided in this section.

4.2.1 Datasets. The researcher prepared five different sets of data from the actual dataset using a 5-fold cross-validation technique, which means in each case one event is selected as the test set and the other four events data are used for training. This cross-validation technique helps in creating a real-time scenario by predicting an event that is completely unknown to the classifier. Table 4.2 shows the number of training and testing samples obtained using each event as a test set.

TABLE 4.2 Training and testing sets obtained using 5-fold cross-validation

Event as a test set	No. of training samples	No. of testing samples
Charlie Hebdo	3,723	2,079
Ferguson	4,659	1,143
Germanwings Crash	5,333	469
Ottawa Shooting	4,912	890
Sydney Siege	4,581	1,221

4.2.2 Evaluation Metrics. Since the fake news detection problem in this study is a binary classification task, the researcher used popular metrics such as Precision, F1, and Recall for evaluating the proposed model's performance.

- True Positive (TP): when predicted as rumors, which are annotated as rumors.
- True Negative (TN): when predicted as non-rumors, which are annotated as non-rumors.
- False Negative (FN): when predicted as non-rumors, which are annotated as rumors.
- False Positive (FP): when predicted as rumors, which are annotated as non-rumors.

$$Precision = \frac{|TP|}{|TP| + |FP|} \tag{4.5}$$

$$Recall = \frac{|TP|}{|TP| + |FN|} \tag{4.6}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.7}$$

4.2.3 Results. Before considering the majority voting process to evaluate ML models, the researcher had conducted a sample test using GaussianNB and MLP classifiers without considering the majority voting of sub-models to see if any improvement in performance can be achieved by implementing majority votingprocess. Table 4.3 shows the sample results of classifiers GaussianNB and MLP operating on each test event with different time series data without applying the majority voting process. Maximum Precision, Recall, and F1 scores for:

- GNB classifier is 0.629 for the Ottawa Shooting test event on 30min data, 0.555 for the Charlie Hebdo test event on 2min data, and 0.428 for the Sydney Siege test event on 2min data, respectively.
- MLP classifier is 0.595 for the Ottawa Shooting test event on 10min data, 0.533 for the Charlie Hebdo test event on 30min data, and 0.527 for the Charlie Hebdo test event on 30min data, respectively.

In using the majority voting process, as discussed in Section 4.1.3, the results are significantly improved with the GaussianNB classifier. Table 4.4 shows the performance of ML models in terms of Precision, Recall, and F1. In the 5-fold cross-validation process, the GaussianNB classifier outperformed all other models in terms of Precision and F1, irrespective of which event data served as the test set with consistent and high Precision scores ranging between $[87-97]\%$. A maximum F1 score of 68.9% was achieved with the GaussianNB classifier on Ottawa Shooting event data. In contrast, a maximum Recall score of 100% was obtained with the MLP classifier using event Germanwings Crash as the test set.

On the contrary, MLP classifier performance was drastically decreased with majority voting process in terms of Precision and F1 scores. Birch and K-Means models achieved good performance when events Germanwings Crash, Ottawa Shooting, and Sydney Siege were used as test sets. Logistic Regression, Multi-layer Perceptron, and Support Vector Machine classifiers performed poorly in which the SVM classifier achieved the poorest performance.

Finally, the researcher calculated micro-averaged Precision, Recall, and F1 using equations 4.8, 4.9, and 6.15.

$$Precision_{micro} = \frac{\sum_{i=1}^{5} |TP_i|}{\sum_{i=1}^{5} |TP_i| + \sum_{i=1}^{5} |FP_i|} \tag{4.8}$$

$$Recall_{micro} = \frac{\sum_{i=1}^{5} |TP_i|}{\sum_{i=1}^{5} |TP_i| + \sum_{i=1}^{5} |FN_i|} \tag{4.9}$$

$$F1^{micro} = 2 \times \frac{Precision_{micro} \times Recall_{micro}}{Precision_{micro} + Recall_{micro}} \tag{4.10}$$

where i refers to each fold in the 5-fold cross-validation process.

In Table 4.5, the micro-averaged results are shown. GaussianNB Classifier achieved outstanding performance in Precision and F1 metrics. Birch and K-Means models obtained almost similar results, but Birch was a little better. Decision Trees, Logistic Regression, Multi-layer perceptron, and Random Forest models achieved poor performances with their Recall scores better than that of their Precision and F1 scores. Support Vector Machines model achieved the poorest performance.

4.2.4 Discussions. The proposed time-series model performed well with some ML models and got poor performance results with a few ML models. In the case of the GaussianNB classifier, with its simplicity, fast computational capability, and

TABLE 4.3 Sample results for individual time intervals

Test event / Classifier	Time interval	Precision	Recall	F1
Charlie Hebdo / GNB	2min	0.565	**0.555**	0.357
	5min	0.575	0.553	0.332
	10min	0.570	0.544	0.313
	30min	0.566	0.534	0.289
	60min	0.565	0.530	0.279
Charlie Hebdo / MLP	2min	0.543	0.524	0.519
	5min	0.543	0.518	0.505
	10min	0.559	0.514	0.486
	30min	0.570	**0.533**	**0.527**
	60min	0.389	0.498	0.437
Ferguson / GNB	2min	0.537	0.536	0.397
	5min	0.534	0.527	0.360
	10min	0.535	0.525	0.340
	30min	0.525	0.515	0.317
	60min	0.528	0.518	0.321
Ferguson / MLP	2min	0.498	0.500	0.453
	5min	0.496	0.500	0.447
	10min	0.514	0.502	0.450
	30min	0.591	0.503	0.439
	60min	0.376	0.499	0.429
Germanwings Crash / GNB	2min	0.483	0.495	0.392
	5min	0.426	0.482	0.364
	10min	0.470	0.494	0.368
	30min	0.407	0.488	0.349
	60min	0.446	0.492	0.358
Germanwings Crash / MLP	2min	0.547	0.502	0.342
	5min	0.547	0.502	0.342
	10min	0.246	0.500	0.330
	30min	0.246	0.500	0.330
	60min	0.246	0.500	0.330
Ottawa Shooting / GNB	2min	0.508	0.502	0.407
	5min	0.543	0.509	0.398
	10min	0.566	0.509	0.384
	30min	**0.629**	0.506	0.365
	60min	0.599	0.505	0.364
Ottawa Shooting / MLP	2min	0.469	0.498	0.333
	5min	0.486	0.499	0.328
	10min	**0.595**	0.506	0.342
	30min	0.486	0.500	0.322
	60min	0.236	0.500	0.321
Sydney Siege / GNB	2min	0.581	0.536	**0.428**
	5min	0.596	0.530	0.398
	10min	0.612	0.526	0.378
	30min	0.616	0.516	0.349
	60min	0.606	0.512	0.339
Sydney Siege / MLP	2min	0.487	0.497	0.404
	5min	0.457	0.492	0.389
	10min	0.486	0.497	0.395
	30min	0.376	0.495	0.365
	60min	0.286	0.497	0.363

TABLE 4.4 Shows the experimental results with each event as a test set

Test event	Model	Precision	Recall	F1
Charlie Hebdo	Birch	0.662	0.197	0.304
	DT	0.371	0.240	0.291
	GNB	0.952	0.237	0.380
	KMeans	0.699	0.195	0.304
	LR	0.076	0.232	0.115
	MLP	0.046	0.304	0.080
	RF	0.332	0.303	0.317
	SVM	0.000	0.000	0.000
Ferguson	Birch	0.782	0.231	0.357
	DT	0.215	0.260	0.235
	GNB	0.870	0.258	0.398
	KMeans	0.641	0.218	0.325
	LR	0.035	0.213	0.060
	MLP	0.021	0.261	0.039
	RF	0.116	0.277	0.164
	SVM	0.000	0.000	0.000
Germanwings Crash	Birch	0.908	0.516	0.658
	DT	0.172	0.461	0.251
	GNB	0.937	0.502	0.654
	KMeans	0.870	0.506	0.640
	LR	0.021	0.625	0.041
	MLP	0.013	**1.000**	0.025
	RF	0.118	0.452	0.187
	SVM	0.000	0.000	0.000
Ottawa Shooting	Birch	0.747	0.523	0.615
	DT	0.170	0.533	0.258
	GNB	**0.974**	0.533	**0.689**
	KMeans	0.743	0.521	0.612
	LR	0.006	0.375	0.013
	MLP	0.002	0.500	0.004
	RF	0.132	0.614	0.217
	SVM	0.000	0.000	0.000
Sydney Siege	Birch	0.561	0.420	0.481
	DT	0.203	0.411	0.272
	GNB	0.971	0.440	0.605
	KMeans	0.381	0.431	0.404
	LR	0.048	0.373	0.085
	MLP	0.033	0.333	0.059
	RF	0.119	0.521	0.193
	SVM	0.000	0.000	0.000

TABLE 4.5 Micro-averaged results

Model	Precision	Recall	F1
BIRCH	0.702	0.323	0.443
DT	0.232	0.318	0.268
GNB	**0.949**	0.356	**0.518**
KMEANS	0.637	0.313	0.419
LR	0.040	0.278	0.069
MLP	0.024	0.324	0.045
RF	0.171	**0.374**	0.235
SVM	0.000		

ability to train well on a small dataset achieved the best performance with the proposed time-series model. MLP classifier had limitations with its training process. There is no guarantee it reached global minima during the training process. Thus, it needs to be trained several times to find the training step with the best RMS error. This makes the training process a time-consuming task. Another important limitation of MLP was the hidden layer setting, which is set by the user. A very less value of the number of hidden layer neurons may cause MLP underfitting issues. If the value is too high, it may result in MLP overfitting. MLP classifier performance with the majority voting process may be improved by tuning training and hidden layer parameters. SVM model achieved shocking results in the experimental analysis because it predicted each whole test event set as rumors even though it contained both rumors and non-rumors. According to Burges [41], kernel selection and discrete data limit SVM performance. Since the time-series data was of pure integer data type and the researcher tried only 'rbf' kernel; SVM may have achieved the poorest performance. Experimenting with different kernel options may improve its performance.

4.3 Concluding Remarks for the Chapter

Verifying news credibility in social media is a challenging task as information spreads rapidly. Fake news detection in social media is a well-known problem; many of the existing studies used various features of social media posts to achieve better fake news detection accuracy. Given a minimal amount of time to detect fake news before they proliferate, there is an on-demand need for models to detect fake news propagation in its early stages. In this study, the researcher proposed a multiple time-series data analysis model that relies only on tweets' temporal characteristics for detecting fake news on social media. With the proposed model, the researcher significantly reduced the time required for training and testing processes as well as reduced the computational complexity of ML models by taking advantage of numerical data. The experimental results showed that with the time-series approach, a high Precision score of 94% was achieved with the GaussianNB classifier.

CHAPTER 5

RUMOR DETECTION ON TIME-SERIES OF TWEETS VIA DEEP LEARNING

False information has become a weapon in cyberwarfare. How to detect false information effectively and efficiently on social media is a challenging problem. In this study, a novel method of rumor detection on Twitter tweets is proposed as a proof-of-concept for the fast detection of false information on social media. Specifically, the proposed method will use the tweets' propagation pattern to detect false information rather than the contents. As a result, the proposed method was very effective in reducing the dimensionality of the input feature set, and it required much less computational time compared to content-based methods. Extensive experiments on the PHEME dataset, a collection of Twitter rumors and non-rumors posted during five breaking news, were performed to demonstrate the effectiveness of the proposed method. The researcher also observed that deep learning models such as recurrent neural networks outperformed classical machine learning models in terms of micro-F score.

In this study, a novel rumor detection method was proposed by using the temporal features of the data. The PHEME dataset[1] was employed, which is a collection of Twitter conversations with two classes: rumor and non-rumor, to demonstrate the effectiveness of the proposed method. The temporal feature was built based on the tweet timestamp, which transforms all Twitter conversation samples into simple vectors representing the number of tweets/retweets along time (in different time

[1] https://figshare.com/articles/PHEME_dataset_of_rumours_and_non-rumours/4010619

intervals). The main features of the proposed method include:

- Since only temporal features are used, there was no need for the extraction and selection of complex features. This reduced the computational time dramatically, which is critical for timely rumor detection.

- The researcher generated the time-series data in pure numeric type, which was very favorable to the classification models and can be readily inputted into a model.

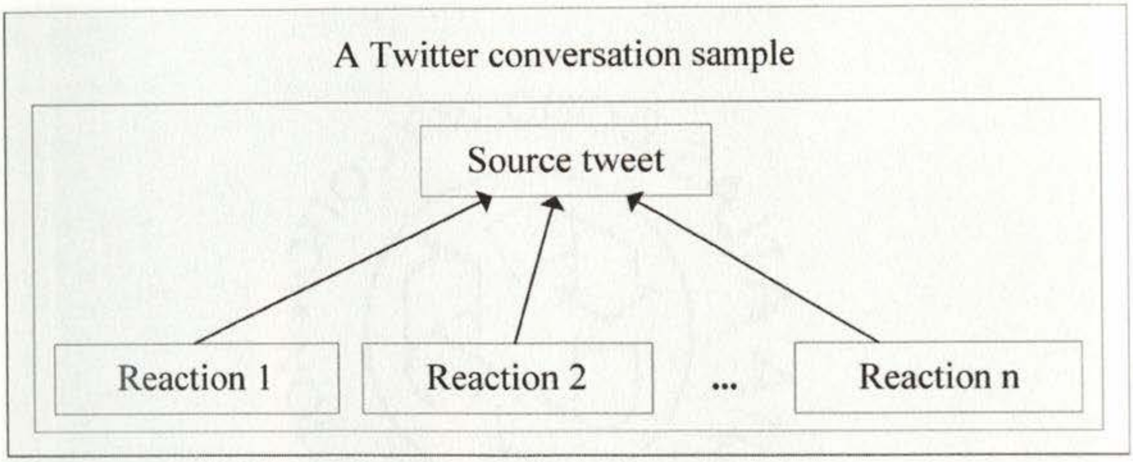
Extensive experiments were performed with both classical classifiers and deep learning models. It was observed that deep learning models such as recurrent neural networks outperformed classical machine learning models by about 4% in terms of micro-F score.

5.1 Rumor Detection Task

Definition to the rumor detection problem is provided here, which is followed by an introduction to the PHEME dataset.

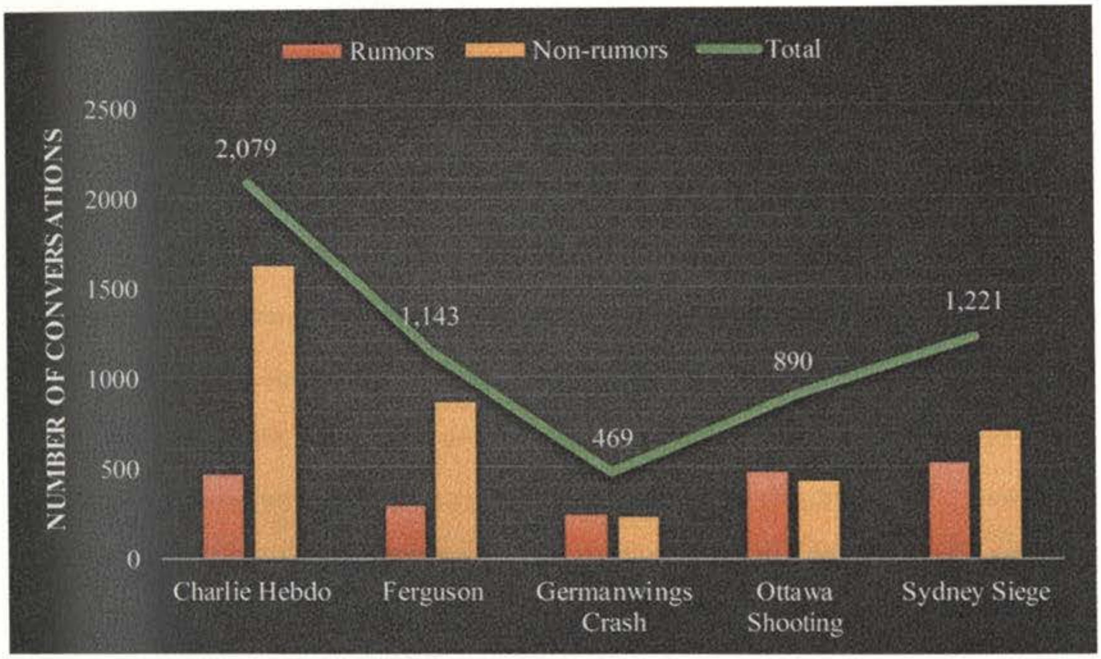
5.1.1 Problem Definition. Classification tasks in machine learning or deep learning typically involve predicting class label(s) $\hat{y}$ for supplied input samples. The current problem is a binary classification task, where the end goal is to predict whether a Twitter conversation sample is a rumor or not. Figure 5.1 shows the structure of Twitter conversation samples. Each Twitter conversation sample will have a source tweet and a set of reactions corresponding to that source tweet, in which the reactions would be retweets or comments. Equation $\hat{y} = f(X)$ defines the task, where $\hat{y} \in \{0, 1\}$ in which 0 represents a non-rumor sample, and 1 represents a rumor sample, f is a classification model, and $X$is a never before seen data sample, where the data sample is one Twitter conversation sample.

FIGURE 5.1 The structure of a Twitter conversation sample



5.1.2 Dataset. The PHEME [42] dataset is a collection of Twitter conversation samples categorized into two classes (i.e., rumors and non-rumors), which are related to five news events, namely, Charlie Hebdo, Ferguson, Germanwings Crash, Ottawa Shooting, and Sydney Siege. For each of the five events, rumor and non-rumor contents consists of the source-tweet and the reactions (a set of tweets corresponding to that source-tweet). Overall, the dataset contains $5,802$ conversation samples, in which the number of rumor samples is $1,972$, and the number of non-rumor samples is $3,830$. Figure 5.2 shows the data distribution of the PHEME dataset. The dataset had unbalanced nature both in terms of event-wise as well as class-distribution-wise. For event-wise, event Charlie Hebdo had got the lion's share of the dataset. Ferguson and Sydney Siege events had almost the same number of samples. Ottawa Shooting had a relatively decent number of samples, and the Germanwings Crash event was the smallest of all the five events with only 469 samples. In the case of class-distribution-wise, only events Germanwings Crash and Ottawa Shooting showed some decent balanced class nature, and other remaining

FIGURE 5.2 Shows the data distribution of the PHEME dataset



events exhibited unbalanced class nature, in which events Charlie Hebdo and Ferguson exhibited high class unbalance. The unbalanced nature has become the challenge to accomplish the classification task.

5.2 Deep Learning Models

To complete the classification task, the researcher employed different deep learning models. Three recurrent neural networks (RNN), namely Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Bi-directional Recurrent Neural Network (Bi-RNN), and one convolutional neural network (CNN) were used. Typically, an RNN consists of a hidden state **h**, and an optional output **y** for a given variable-length input sequence $\mathbf{x} = (x_1, \cdot\cdot\cdot, x_T)$. At each time *t*, the hidden state

$\mathbf{h}_{(t)}$ is given by [43]:

$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, x_t), \tag{5.1}$$

where f is a non-linear activation function.

5.2.1 LSTM. Hochreiter and Schmidhuber developed LSTM in 1997 [44]. The basis for the evolution of the LSTM network is finding a solution to the vanishing gradient problem in feedforward networks and learning long-term dependencies present in the input samples. It is a special type of Recurrent Neural Network (RNN), which consists of an information-carrying path across many time-steps to save information for later use, thus preventing older signals from gradually vanishing. The major components of an LSTM unit are cell, input gate, output gate, and forget gate. The function of the cell component is to remember values over arbitrary time intervals, and the function of the three gates is to regulate the flow of information into or out of the cell [45]. Each $j-th$ LSTM unit has a memory c_t^j at time t, and the output h_t^j is given by [46]:

$$h_t^j = o_t^j \tanh(c_t^j), \tag{5.2}$$

where o_t^j is an output gate.

5.2.2 GRU. This model was developed by Chung et al. in 2014 [46]. Its architecture is very similar to LSTM, but it is somewhat streamlined, making it cheaper to run. However, it may not have the same representational power as that of LSTM. It has a smaller number of parameters than LSTM due to the absence of an output gate. It uses the update and reset gates to control the flow of information, and the former is used in deciding how much of past information should be passed along to the future. The latter is used to determine how much of past information should be discarded [45]. The activation h^j_t is the linear interpolation between h^j_{t-1} and $\tilde{h}^j_t$, which are previous activation and candidate activation respectively at time t [46]:

$$h^j_t = (1 - z^j_t)h^j_{t-1} + z^j_t\tilde{h}^j_t, \tag{5.3}$$

where z^j_t is an update gate.

5.2.3 Bi-RNN. A traditional RNN is order dependent and processes the time-steps in order. Altering the time-steps in an input sequence can affect the representations extracted by RNNs. The Bi-RNN [47] exploits the order sensitivity present in RNN and processes the input sequence, both chronologically and antichronologically. In this way, the patterns which are overlooked by traditional RNN can be identified. It has twice the number of parameters of a traditional RNN, which makes it overfit quickly, but it can be controlled using some good regularization techniques. It is very popular in natural language processing applications [45]. RNN variants GRU and LSTM layers were used in the experiments. The forward and backward hidden sequences (i.e., $\overrightarrow{h}$ and $\overleftarrow{h}$) for Bi-RNNs are given by:

$$\overrightarrow{h}_t = \mathsf{H}(W_{x\overrightarrow{h}}x_t + W_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}) \tag{5.4}$$

$$\overleftarrow{h}_t = \mathsf{H}(W_{x\overleftarrow{h}}x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}), \tag{5.5}$$

where the W terms denote weight matrices, the b terms denote bias vectors, and H is the hidden layer function [48].

5.2.4 CNN. As one special CNN, 1D convnets are good alternatives to RNNs for simple tasks, for instance, text classification and time-series forecasting. In operation, they basically extract local 1D patches from sequences, which is similar to 2D convolution layers. Since the same input transformation is applied to every patch, once a pattern is learned at a position in a sequence, it can be identified later in a different position. 1D pooling of sequence data is also similar to 2D pooling, which is used to reduce the length of 1D inputs. 1D pooling involves identifying the 1D patches from the input and then outputting the values based on the chosen pooling type, for example, maximum or average [45].

5.3 Experiment

The workflow is divided into two components: generation of time-series data for each time interval (T), and training deep learning models to complete the classification task.

5.3.1 Time-series Data Generation. The PHEME dataset contains five events of rumor and non-rumor Twitter conversation samples. The researcher transformed each of those conversations into time-series vectors for each time interval T . Once transformed, each row in the time-series data structure represents one whole conversation, and each of its columns is the total reaction counts with respect to the chosen time interval step size. If $E = \{e_i\}$ is a set containing all the five events, then for each event data $c_{ij} \in e_i$ represents individual conversation samples. The PHEME dataset provides both rumor and non-rumor conversations separately for all events. The researcher iterated over all those events. For every conversation sample present in them, its source-tweet timestamp $timeSource$ and its $timeReactions = \{tr_1, tr_2, \cdots , tr_n\}$, which is a collection of timestamps of all the reactions corresponding to that source-tweet were extracted. The maximum length $N(c)$ of the vector representation for every conversation sample can be determined by,

$$N(c) = \frac{\max(timeReactions) - timeSource}{T} \tag{5.6}$$

For a conversation sample c, if $(a, b]$ is the time interval limit for the k-th interval, where $k = 1, 2, \cdot\cdot\cdot, N(c)$ then the total count of reactions falling into that time interval is given by,

$$count_k = \mathbf{card}(Q) \tag{5.7}$$

where $Q \subset timeReactions$ and $Q = \{x \mid x > a \wedge x \leq b\}$, and x is the timestamp of a reaction (tweet) and cardinality measures the size of set Q, and the transformed vector representation is as follows:

$$V(c) = [count_k \quad count_{k+1} \quad \cdots \quad count_N] \tag{5.8}$$

Then the final feature vector representation of conversation samples for each event is given by

$$e_i = \begin{bmatrix} V(c_1) \\ V(c_2) \\ \vdots \\ V(c_n) \end{bmatrix} \tag{5.9}$$

Since vector representations have variable lengths, the researcher padded all of them with $0s$ at their tail end. For the experimental analysis, non-rumors and rumor samples were labeled with $0s$ and $1s$, respectively. Algorithm 1 shows the pseudocode for generating the time-series data.

5.3.2 Training Deep Learning Models. Once the researcher completed generating the time-series datasets, some basic data pre-processing using *scikit-learn Machine Learning in Python library* [26] were performed. In the experimental analysis, 5-fold cross-validation was performed, which means for every fold, one event is

Algorithm 1: Time-series data generation

```
Input: $P$
/* $P$ is a data element containing file paths to all
   sub-directories of all events $E$                                      */
Output: stores generated time-series data into '$.csv$ 'files
initialize an empty dictionary $var$
forall $sub\ dir \in P$ do /* $sub\_dir$ is a sub directory path present in
   $P$                                                                    */
    forall $t \in T$ do
    └ (
    /* $t = 2, 5, 10, 30, 60$                                              */
    ) forall $c \in sub\ dir$ do
    └ (
    /* $c$ is a conversation sample present in $sub\_dir$                  */
    ) $var[c] \leftarrow emptylist$
    $a \leftarrow timeSource$
    /* lower time interval limit                                          */
    $b \leftarrow timeSource + t$
    /* upper time interval limit                                          */
    while $True$ do
        if $b \geq \max(timeReactions)$ then
            append $count_k$ to $var[c]$
            break
        else
            append $count_k$ to $var[c]$
            $a \leftarrow b$
            $b \leftarrow b + t$
    save $var$ into a '$.csv$ 'file and clear it
```

used as a test set, and other remaining events constitute a training set. The train and test sets proportions for all values of T are shown in Table 5.1.

TABLE 5.1 5-fold cross-validation train and test sets proportions for all values of T

Selected Test Event	Charlie Hebdo	Ferguson	Germanwings Crash	Ottawa Shooting	Sydney Siege
Train Set	3,723	4,659	5,333	4,912	4,581
Test Set	2,079	1,143	469	890	1,221

The cross-validation technique helped in mimicking the real-application scenario because the event to be predicted was completely unknown to the classification model. A subset (10%) of the training set was utilized for validation in the training procedure. The researcher trained deep learning models by iterating over T . Since the dataset had unbalanced class nature, class weights using sklearn's class_weight library with a *'balanced'* scheme were computed. Equation 5.10 [26] is used to calculate the class weights.

$$\textit{class weights} = \frac{n\_samples}{(n\_classes \times bincount(y))} \tag{5.10}$$

where y is the original class labels per sample, $n\_samples$ is the number of data samples. $n\_classes$ is the number of unique class label values present in the dataset, and $bincount(y)$ counts the number of occurrences of each value in y of non-negative integers. The researcher used the calculated class weights for weighting loss functions during the training process. Another challenge in the training process is data ambiguity. Some samples exist in the generated time-series data with the same time-series vector representation but different class labels. All of them were discarded to achieve unbiased deep learning models' training. The researcher saved the shapes of the generated train and test sets for each time interval and selected test event after removing the duplicate samples and plotted the data distributions in Figure 5.3. It is observed from Figure 5.3 that with the increase of time interval, the number of duplicate samples increased, which caused a significant loss of data samples. Before feeding the time-series data into deep learning models, the researcher had scaled the data using sklearn's MinMaxScaler to normalize the data. Table 5.2 shows the hyperparameter settings used for the NN models' training. All the NN models are designed using *Keras: The Python Deep Learning Library [49]*. They have only one special layer (i.e., Conv1D, LSTM, BiLSTM, GRU, BiGRU) as their first hidden layer, followed by Dropout, Flatten and output Dense layers. The dense output layer

FIGURE 5.3 Shows the time-series datasets distribution after removal of duplicates

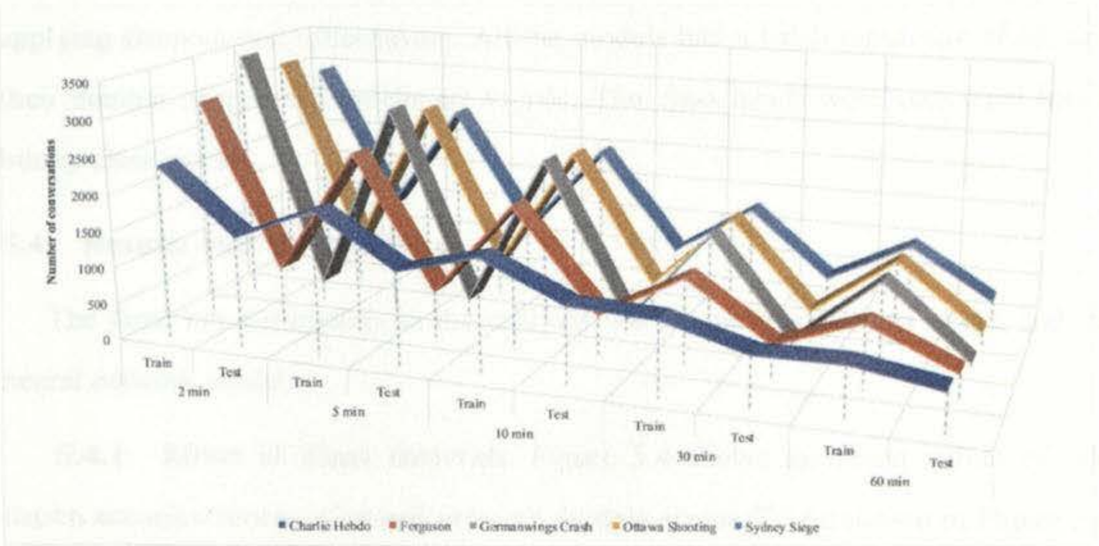


TABLE 5.2 NN models' hyperparameter settings

Model	Hidden layer units	Dropout	Optimizer	Loss function
Conv1D	64	0.3	Adam (0.0001)	Categorical Cross-entropy
LSTM	32		Adam (0.001)	Mean-Square Error
BiLSTM	48			
GRU	32	0.3		
BiGRU	48			

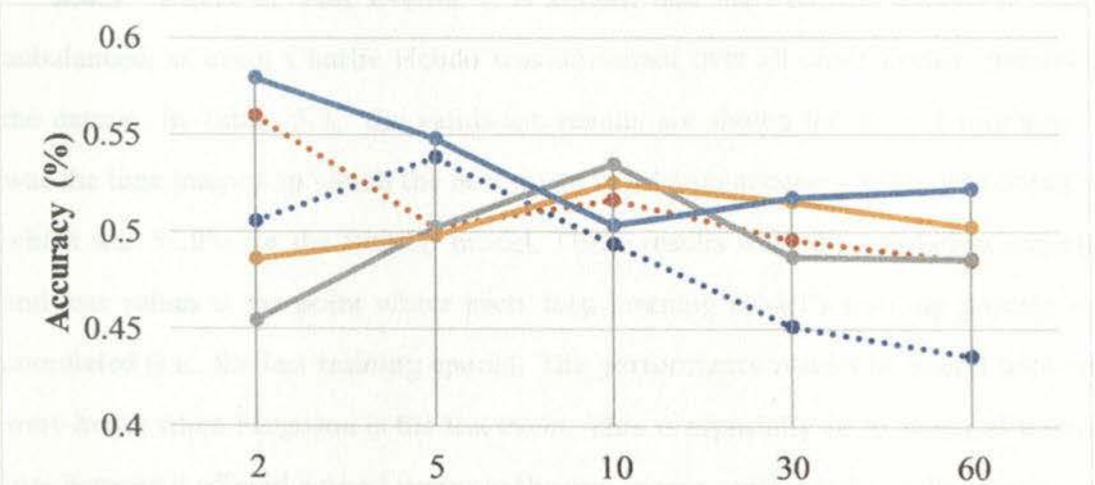
was activated using a *sigmoid* activation function. The Conv1D model's kernel size is set to 3, and it is activated using a *tanh* function. The MaxPooling1D layer was used with a poolsize of 2 only for the Conv1D model to downsample the data before applying Dropout and other layers. All the models had a batch input size of 64, and their number of training epochs set to 100. The class labels were converted into a binary class matrix.

5.4 Results and Discussions

The three key parameters in the analysis were time interval, test event, and the neural network model.

5.4.1 Effect of Time Intervals. Figure 5.4 shows the mean values of validation accuracy scores of neural network models across T . As shown in Figure 5.4, the performance of neural network models fluctuated up and down as the T value was increased. The main observation was that when T is small, models Conv1D, LSTM, and BiGRU performed well, which means they have made good use of the subtle variations present in the long propagation patterns. However, when T gets bigger, their performances decreased, and there was a significant performance drop between $T = 2$ and $T = 60$ minutes. In contrast, models BiLSTM and GRU models achieved good performances when T is large. That means they are powerful enough to overlook and identify the differences in the short propagation patterns, which is a good sign for improving NN models' training time as the dimensionality of the feature set gets reduced. Interestingly, their performances were improved by a decent margin between $T = 2$ and $T = 60$ minutes. Nonetheless, for $T = 10$, the performances of all the models were close to each other, which is not the case for other time intervals.

FIGURE 5.4 5-fold mean validation accuracy scores of neural network models across T



5.4.2 Effect of Neural Network Models. From Figure 5.4, even though T changes, the majority of the neural network models showed reasonable individual time interval performance consistency. Again, for higher values of T, the models were restricted to limited variations present in the propagation patterns of conversation samples, causing them to lose their power to perform better classification. Overall, neural networks that are designed using GRU and BiGRU layers achieved better performance than other models for most of T's values due to their parameter size, which was comparatively smaller than LSTM and BiLSTM layers. They have a good gating mechanism, too, to control the flow of information. Moreover, as there was a difference in performances among models for higher and lower values of T , using some ensemble techniques on the individual models, in which each model gets its best suitable time-series data may improve classification performance.

5.4.3 Effect of Test Events. It is known that the PHEME dataset is highly unbalanced, as event Charlie Hebdo was dominant over all other events present in the dataset. In Table 5.3, the validation results are shown for $T = 2$ min since it was the time interval in which the best mean validation accuracy score was obtained, which was 57.9% for the BiGRU model. These results were the validation accuracy and loss values at the point where each deep learning model's training process was completed (i.e., the last training epoch). The performance results of neural networks were better when Ferguson is the test event. This is especially so in terms of training loss because it offered a good rumor to the non-rumor sample ratio in the training set, which was $1,688 : 2,971$. However, when Charlie Hebdo is the test event, it offered even better rumor to the non-rumor sample ratio, i.e., $1,514 : 2,209$. On the flip side, compared to Ferguson being considered as a test event, the number of training samples was more than that of Charlie Hebdo. If Ferguson is the test event, then the total number of training samples is 4,659, and 3,723 total samples count in the case of Charlie Hebdo as the test event. It means significant data loss has occurred impacting NN models' training processes. Compared to events Charlie Hebdo and

Ferguson, other remaining events constituted to poor rumor to non-rumor sample ratio.

TABLE 5.3 NN models' validation results for $T = 2$ min (values are given in $[0 - 1]$)

Time	Event	Conv1D		LSTM		BiLSTM		GRU		BiGRU	
		Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss
2min	Charlie Hebdo	0.46186	0.81045	0.51271	0.32996	0.51271	0.33063	0.64407	0.35593	0.44492	0.55932
	Ferguson	0.57947	0.66224	0.54636	0.26432	0.51987	0.28071	0.29139	0.70861	0.56291	0.27162
	Germanwings Crash	0.5616	0.72854	0.54728	0.28441	0.48997	0.30552	0.51289	0.29355	0.69914	0.49854
	Ottawa Shooting	0.46894	0.78378	0.47205	0.31332	0.46584	0.3112	0.50311	0.29545	0.47826	0.31342
	Sydney Siege	0.45578	0.91026	0.71769	0.5	0.28231	0.71769	0.47619	0.33028	0.71088	0.5034
Mean		*0.5055*	*0.7791*	*0.5592*	*0.3384*	*0.4541*	*0.3892*	*0.4855*	*0.3968*	***0.5792***	*0.4293*

5.4.4 Other Observations. As the rumor detection problem is a typical binary classification task, the researcher considered the evaluation metric called F1 score, which is the weighted average of precision and recall scores for evaluating the classification models' performances. Calculations were done for the macro and micro averaged testing results for all combinations of T and events E. Tables 5.4 and 5.5 include the mean of 5-fold cross-validation micro and macro averaged testing results of neural network models with respect to T . The previous work results [28] are shown in Table 5.6, which were obtained using classification models: Decision Trees, Gaussian Naive Bayes, Logistic Regression, Multi-layer Perceptron, and Random Forests only by considering the micro averaging scheme. In this study, the researcher was also interested in how the macro averaging scheme would impact the classification models' performances. The researcher did not create separate instances for each classification model as in previous work [28] to operate on each time interval time-series data rather for all the neural network models used one static instance to work on all combinations of T and E time-series data. In Tables 5.4 and 5.5, the micro-averaged results were better than that of the macro-averaged results for all values of T . Micro-averaged testing results were high for lower values of T . In contrast, macro-averaged testing results were better in case of higher T values. For both the averaging schemes, the results did not vary too much for all of the models with respect to T , except in the case of the BiGRU model, where there was an almost 10% performance difference. Model LSTM stood out as the top performer in the case of a macro-averaging scheme by achieving a 49.4% accuracy score for $T = 30$; and when the micro-averaging scheme was used model BiGRU obtained a high accuracy score. It was clear from the results shown in Table 5.6 that the researcher improved the micro-F1 score by 4% roughly compared to a previous study.

In [5], content-based and social features were explored in the Twitter data contained in PHEME dataset, where content-based included feature extraction methods: Word Vectors, Part-of-speech Tags, Capital Ratio, Word Count, Use of Question Mark, Exclamation Mark, and Period; social features include: Tweet Count, Listed Count, Follow Ratio, users'age, and account verification status. Conditional Random Fields (CRF) classifier was the best model in their analysis, and its F1 scores were 0.606 and 0.339 for content-based and social features, respectively. When both these heavyweight feature sets were used together, the CRF model's F1 score was0.607. It is a little improvement compared to the F1 score obtained with only content-based features even after employing extensive feature engineering (i.e., social features extraction) that describes how difficult it was to obtain slight performance gain using this dataset.

On the other side, an increase in the complexity of the feature set may cause extra cost in terms of data pre-processing requirements, computational capabilities, and training time. The key observation was that as the T value increases, most of the models' performance showed gradual decay in the case of a micro-averaging scheme. A big difference exists in models' performances for $T = 2$ and $T = 60$. This observation shows that as the sequence length of the propagation patterns of Twitter conversations decreases, the subtle variations in the propagation patterns were not explored by the classification models to properly classify the input samples. In the case of a macro-averaging scheme, the performances of the classification models improved for medium time interval lengths, i.e., $T = 10$ and 30 than that of lesser values of T.

TABLE 5.4 Micro-averaged testing results in F1 scores

Time interval	Conv1D	LSTM	BiLSTM	GRU	BiGRU
2min	**0.498**	**0.522**	0.49	**0.512**	0.506
5min	0.496	0.474	0.478	0.478	**0.564**
10min	0.478	0.49	**0.512**	0.496	0.472
30min	0.454	0.514	0.496	0.498	0.502
60min	0.416	0.45	0.456	0.472	0.456

TABLE 5.5 Macro-averaged testing results in F1 scores

Time interval	Conv1D	LSTM	BiLSTM	GRU	BiGRU
2min	0.478	0.46	0.44	0.418	0.41
5min	**0.488**	0.464	0.458	0.468	0.41
10min	0.458	0.478	**0.486**	0.476	0.458
30min	0.436	**0.494**	0.462	**0.478**	**0.47**
60min	0.388	0.434	0.442	0.456	0.444

TABLE 5.6 Comparing current study with baselines

Previous study (baselines)						Current study
Model	DT	GNB	LR	MLP	RF	BiGRU
F1	0.268	**0.518**	0.069	0.045	0.235	**0.564**

5.5 Concluding Remarks for the Chapter

For fast rumor detection in social media, a multiple time-series data analysis approach was proposed. Compared to the literature's content-based methods, the proposed method used only the temporal features of tweets. Because information propagates fast on social media, the timely detection of false information using the proposed method could deter the proliferation of false information before any unwanted disturbances occur in society. This approach is simple but very effective in reducing the dimensionality of the input feature set, which helped improve training time and reduced the computational complexity of classification models because of the nature of the generated time-series data. By experimenting with advanced deep learning models, the researcher improved the micro-averaged F1 score by 4.6%, compared to the baselines [28].

CHAPTER 6

ENSEMBLE DEEP LEARNING ON TIME-SERIES REPRESENTATION OF TWEETS FOR RUMOR DETECTION IN SOCIAL MEDIA

Social media is a popular platform for information sharing. Any piece of information can be spread rapidly across the globe at lightning speed. The biggest challenge for social media platforms like Twitter is how to trust news shared on them when there is no systematic news verification process, which is the case for traditional media. False information, for example, detection of rumors, is a non-trivial task, given the fast-paced social media environment. In this study, the researcher proposed an ensemble model that performs a majority-voting scheme on a collection of neural networks' predictions using time-series vector representation of Twitter data for the fast detection of rumors. Experimental results showed that neural network models outperformed classical machine learning models in terms of a micro F1 score.

In this chapter, Twitter data's temporal features were explored for the timely detection of rumors in social media. Tweet creation timestamp can readily be extracted from tweets, and there is no time delay to collect timestamp features. No sophisticated data pre-processing is required to convert them into useful features to train a classification model. Based on this observation, an ensemble-based multiple time-series analysis model was proposed using deep learning models for the timely detection of rumors in social media. Specifically, time-series data were generated by transforming Twitter conversations, where each conversation contains a list of tweets, into times-series vectors that contain reaction counts as features, and fed as input to deep learning models. The contributions of the proposed method are:

- With the proposed method, computational complexity can be significantly reduced, as timestamps of tweets are needed rather than their contents or user social engagements to perform feature extraction. Moreover, the extracted feature set is of numeric type, which is amicable to classification models.
- The proposed ensemble model improves classification models' performances. It uses the majority-voting scheme on multiple neural networks that are part of the ensemble model and takes advantage of their strengths.
- The proposed method was validated on the PHEME[1] dataset, and the performance results demonstrate the effectiveness of the proposed scheme.

6.1 Problem Formulation

This section defines the rumor detection problem, provides an overview of tweets' general features, and discusses breifly about the feature extraction method for parsing Twitter data.

6.1.1 Rumor Detection. Rumor detection involved identifying whether a data sample is a rumor or not. In machine learning, this kind of problem is termed as a classification task, in which the classification model gets trained with an adequate number of training samples and tries to classify a never before seen testing sample as rumor or not. Therefore, the problem is given by $\hat{y} = f(X)$, where f is the classification model, and X is a completely new data sample (a Twitter conversation sample that is transformed into a time-series vector) to it. The $\hat{y}$ is the prediction of the classification model, and it has only two values since the PHEME dataset has

[1] https://figshare.com/articles/PHEME_dataset_for_Rumour_Detection_and_eracity_Classification/6392078

two classes. In this study, the researcher used 0's and 1's to represent non-rumor and rumor samples, respectively, i.e., $\hat{y} \in \{0, 1\}$.

6.1.2 General Features of Tweets. Typically, a classification task using machine learning or deep learning requires the extraction of useful features from the dataset. A variety of features can be extracted from Twitter data; for example, four types of features were extracted from Twitter data to study the spread of anomalous information in social media [50]. They are user profile features (users' friends and followers count), user network features (users' EgoNet features), temporal features (retweet count), and content features (e.g., whether a tweet has question mark). However, based on the theories of rumor propagation, authors in [36] considered temporal features as one of the key properties for studying the spread of rumors since, according to social psychologists, rumormongers have short attention. In this study, for the fast detection of rumors on social media, the researcher solely focused on the temporal features of Twitter data, which are the creation timestamps of tweets. These timestamps can be readily fetched. This study strictly relied on them for the generation of time-series data, which involved simple calculations, i.e., counting of the number of tweets for given time interval limits.

6.1.3 Feature Extraction. In general, for Twitter data, the researcher used a parser to read and extract the required information from it depending upon its data type. In this study, the Twitter data utilized was available in *JSON* format. The researcher used a suitable parser to read that information and extract the required features, which are the creation timestamps of tweets.

6.2 Ensemble Learning

An overview to the ensemble learning and the proposed model are discussed next.

6.2.1 Overview of Ensemble Learning. Ensemble learning is a concept in which many weak or base learners try to solve a single problem. An ensemble contains many base learners, and its generalization ability is more powerful than that of the base learners [51]. Ensemble methods work on a set of hypotheses derived from training data rather than relying on one hypothesis. Constructing ensembles is a two-step process. First, the required number of base learners are produced. Secondly, all the base learners are grouped, and typically majority voting is applied for classification problems, and weighted averaging combination schemes are used for regression problems. Popular ensemble methods are boosting [52], bagging [53], and stacking [54].

The Boosting method focuses on fitting multiple weak learners sequentially. Each model in a sequence emphasizes the data samples that were badly treated by its previous model. AdaBoost [52] algorithm is a good example of boosting, which is simple and can be applied to data that is numeric, textual, etc. In the bagging method, multiple bootstrap samples are generated from the training data, and a weak independent learner is fitted for each of these samples. Finally, all the predictions of weak learners are aggregated to determine the most-voted class. RandomForests [55] algorithm is a good example of the bagging method, one of the most accurate learning algorithms and runs efficiently on large databases. In the stacking method, using different learning algorithms, multiple first-level individual learners are created. These learners are grouped by a second-level learner (meta-learner) to output a prediction [54].

6.2.2 Bagging Learning. Bagging learning has been studied extensively in the literature. Bagging, also known as bootstrap aggregation, is a popular ensemble method that is useful in reducing the high variance of machine learning algorithms. In the bagging technique, several datasets are derived from the original training data set by employing sampling with replacement strategy. That means some observations in the derived datasets may be repeated. These datasets are used to train classification or regression models, and outputs are typically weighted average for regression cases, or majority voted for classification problems.

The majority voting grouping technique is used in [56, 57]. In [56], the ensemble's bagging method is used with REPTree as a base classifier for an intrusion detection system and compared to other traditional machine learning techniques. It was shown that the ensemble bagging method achieved high classification accuracy by employing the NSL_KDD dataset. Authors in [57], proposed to use dictionary learning with random subspace and bagging methods and introduced Random Subspace Dictionary Learning (RDL) and Bagging Dictionary Learning (BDL) algorithms. Their experimental analysis concluded that ensemble-based dictionary learning methods performed better than that of single dictionary learning.

The weighted averaging grouping technique is employed in [58, 59]. In [58], the Neural Network Ensemble (NNE) approach was proposed to improve the generalization ability of neural networks and to reduce the calculation errors of Density Functional Theory (DFT). It is shown that both simple averaging and weighted averaging grouping techniques helped in improving DFT calculation results. Authors in [59] proposed a method for improving image classification performance using SVM ensembles. Optimal weights for the base classifiers in the SVM ensemble are estimated by solving a quadratic programming problem. These weights are then used to combine the base classifiers to form an SVM ensemble.

The optimization of a generic bagging algorithm was studied in [60]. The authors added an optimization process into the bagging algorithm that focuses on selecting better classifiers, which are relatively efficient, and proposed Selecting Base Classifiers on Bagging (SBCB) algorithm. Experimental results proved that their SBCB algorithm performed better than the generic bagging approach.

6.2.3 Deep Bagging Learning. Because deep neural networks are nonlinear methods and have high variance, ensemble learning can combine the predictions of multiple neural network models to achieve less variance among the predictions and decrease the generalization error. An ensemble method is applied to neural networks mainly by (1) varying training data (data samples used to train models in the ensemble are varied), (2) varying choice of the models in the ensemble, and (3) varying the combination techniques that determine how outputs of ensemble members are combined.

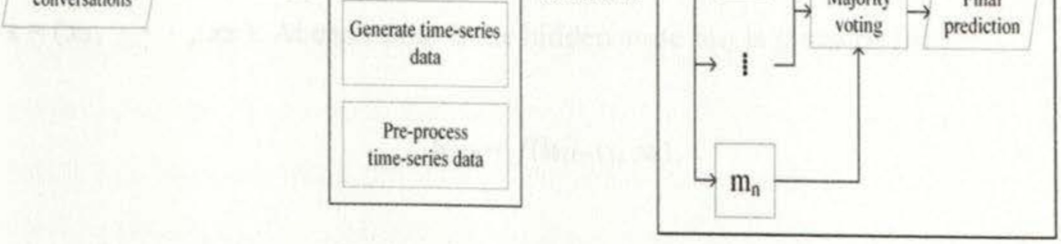
In [61], the authors proposed a method that used the Convolutional Neural Network (CNN) and the deep residual network (ResNET) ensemble-based classification methods for Hyperspectral Image (HSI) classification. Their proposed method used deep learning techniques, random feature selection, and majority voting strategy. Moreover, a transferring deep learning ensemble was proposed to make use of the learned weights of CNNs. In [62], two cooperative algorithms, namely NegBagg (bagging is used) and NegBoost (boosting is used), were proposed for designing a neural network (NN) ensembles. These algorithms used a negative correlation algorithm while training NNs in the ensemble. Applying these models to well-known problems in machine learning showed that with a lesser number of training epochs, compact NN ensembles with good generalization were produced.

In [63], a bagging ensemble was proposed to improve the prediction performance of artificial neural networks (ANN) to tackle the bankruptcy prediction problem. Experimental results showed that the proposed method improved the performance of ANNs. Bagging technique using an ANN is proposed to address imbalance datasets on clinical prediction in [64], and experimental results showed that this method improved the prediction performance.

6.2.4 Overview of the Proposed Model. The proposed model has two key components: a data pre-processing method and an ensemble model. First, raw Twitter conversations were processed to transform them into the required data format, and then the transformed data was supplied to the ensemble model to perform the classification task. The ensemble model consisted of six different neural networks (base learners) trained using the generated time-series data. Their predictions were grouped so that the majority voting scheme was applied to determine the outcome as rumor or non-rumor.

6.3 Methodology

FIGURE 6.1 Shows the proposed model for rumor classification taking Twitter conversations as input, which are cleaned in the data pre-processing block and fed as input to the ensemble model that performs the majority voting to determine the final prediction



The structure of the proposed model is shown in Figure 6.1. The model takes Twitter conversations as input, where each conversation is a stream of tweets that contains source-tweet and its corresponding reactions. In the data pre-processing stage, every tweet is parsed and its creation timestamp value is extracted. Once all tweets were parsed, time-series data for different time intervals were generated and conducted data cleaning. Then, the cleaned data were fed as input to the ensemble model. The ensemble model has n base learners, which are n different neural networks represented as $m_1, m_2, \cdots, m_n$, where each of them yields its prediction results (i.e., $r_1, r_2, \cdots, r_n$). Finally, the majority-voting process was performed on all the predictions of those base learners, i.e., summing up all the prediction results and deciding the final prediction result as 0 (non-rumor) if the total sum was less than $\lfloor n/2 \rfloor + 1$ or as 1 (rumor) otherwise.

6.3.1 Neural Networks Models Considered. The ensemble model constitutes base learners designed using Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Bi-directional Recurrent Neural Network (Bi-RNN). Six base learners were designed in this work: BiGRU, BiLSTM, GRU, LSTM, LG (a combination of LSTM and GRU layers), and RNN.

6.3.1.1 RNN. An RNN is a type of neural network that processes sequences by iterating through the sequence elements [45]. Typically, it consists of a hidden state **h**, and an optional output **y** for a given variable-length input sequence $\mathbf{x} = (x_1, \cdots, x_T)$. At each time t, the hidden state $\mathbf{h}_{(t)}$ is given by [65]:

$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, x_t), \tag{6.1}$$

where f is a non-linear activation function. The researcher used *Keras'* SimpleRNN [49] layer in the experiments.

6.3.1.2 LSTM. It is a special type of RNN and was developed by Hochreiter and Schmidhuber in 1997 [66]. It consists of four major components: cell, forget gate, input, and output gates. Component cell functions to memorize values over arbitrary time intervals and three gates regulate the flow of information into or out of the cell [45]. Each j^{th} LSTM unit has a memory c^j_t at time t, and the output h^j_t is given by [67]:

$$h^j_t = o^j_t \tanh(c^j_t), \tag{6.2}$$

where o^j_t is an output gate.

6.3.1.3 GRU. Chung et al. in 2014 [67] developed Gated Recurrent Unit, which has an architecture similar to LSTM. There is no output gate in GRU, which means it has a lesser number of parameters than LSTM. To control the flow of information, it uses the update and reset gates. These gates decide how much of past information should be passed along to the future or discarded [45]. The activation h^j_t is the linear interpolation between h^j_{t-1} and $\tilde{h}^j_t$, which are previous activation and candidate activation respectively at time t [67]:

$$h^j_t = (1 - z^j_t)h^j_{t-1} + z^j_t \tilde{h}^j_t, \tag{6.3}$$

where z^j_t is an update gate.

6.3.1.4 Bi-RNN. A traditional RNN processes the time-steps in order, whereas Bi-RNN [68] exploits the order sensitivity present in RNN. The input sequence can be processed in forward and reverse directions. It may have overfitting issues as it has twice the number of parameters of a traditional RNN. However, overfitting problems can be controlled by employing good regularization techniques [45]. The researcher employed RNN variants GRU and LSTM layers in the experiments.

The forward and backward hidden sequences (i.e., $\overrightarrow{h}$ and $\overleftarrow{h}$) for Bi-RNNs are given by:

$$\overrightarrow{h}_t = \mathsf{H}(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}) \qquad (6.4)$$

$$\overleftarrow{h}_t = \mathsf{H}(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}), \qquad (6.5)$$

where the W terms denote weight matrices, the b terms denote bias vectors, and H is the hidden layer function [69].

Once the base learners $(m_1, m_2, \cdot \cdot \cdot, m_n)$ complete their training procedures, the ensemble model combines all of their predictions and performs majority voting procedure on them to determine the ensemble model's evaluation metrics. First, the researcher created the proposed ensemble model that consisted of six base learners. Then, the researcher experimented on the proposed model by tuning its hyperparameters such as its batch input size and learning rate. New ensemble models were created using RNN, LSTM, and GRU layers to obtain a comprehensive set of results to efficiently analyze and determine the effectiveness of each ensemble model in detecting rumor Twitter conversations. Variants of the ensemble model will also have six base learners.

6.3.2 Implementation-I. In implementation-I, each of five base learners (BiGRU_1, BiLSTM_1, GRU_1, LSTM_1, and simple RNN_1) had one hidden layer, and the sixth based learner (LG _1) had two hidden layers, followed by one Dense output layer. For all the base learners, the number of hidden layer units was determined based on the integer value obtained from $(seq\ len + 2)/2$, where seq_len was the length of the feature set (i.e., vector length of the time-series data). Constant 2 was used because the number of classification outputs was two (rumor and non-rumor). The researcher considered this approach following one of the rule-of-thumb methods, which states that the number of hidden layer neurons should be between the input layer's size and the size of the output layer [70]. *RandUniform* kernel initializer was used for all the hidden layers with values $(-0.5, 0.5)$. *sigmoid* activation was applied only to the RNN model's hidden layer, and the Flatten layer was applied only to BiGRU and BiLSTM models to flatten the data before the final output Dense layer that was activated using *softmax* function. Adam optimizer was used with a learning rate of $1.00E-05$ along with a *categorical cross-entropy* loss function. The batch input size was set to 32, and the number of epochs was 300. The Dropout technique was not used with these models since their architectures were simple, and using it may have caused under-fitting issues. The variants of the proposed model followed the same neural network design except for the hyperparameter that are tuned, for example, the batch input size and learning rate.

TABLE 6.1 Configurations of NN models

NN model	# of hidden layers	Hidden layer units	Dropout
RNN_1	1	$(seq\_len+2)/2$	N/A
GRU_1			
LSTM_1			
RNN_2	3	16, 32, 64	0.25
GRU_2			
LSTM_2			
RNN_3	2	64, 32	
GRU_3			
LSTM_3			

6.3.3 Implementation-II. Six base learners (RNN_1, RNN_2, RNN_3,GRU_1, GRU_2, and GRU 3) were used in this implementation. To create new ensembles with new base learners, RNN, LSTM, and GRU layers were used. For instance, for base learners designed using the RNN layer, the researcher reused the RNN 1 base learner designed for implementation-I and new base learners were created by adding extra hidden layers with increasing (RNN_2) and decreasing (RNN_3) number of hidden layer units. The configurations of the base learners are shown in Table 6.1. All these base learners had the final output dense layer with $softmax$ activation and loss function as *categorical cross-entropy*. $RandUniform$ kernel initializer was applied with values $(-0.5, 0.5)$. The number of training epochs was set to 300. For RNN_1, GRU_1, and LSTM_1 base learners in Table 6.1, *seq_len* was the feature set's length.

6.3.4 Implementation-III. Similar to implementation-II, six base learners (RNN_1, RNN_2, RNN_3, LSTM_1, LSTM_2, and LSTM_3) were employed in implementation-III. The hyperparameters were set similarly.

6.4 Dataset

The PHEME dataset, time-series data generation, and data pre-processing on the time-series data are discussed next.

6.4.1 PHEME Dataset. In this study, the researcher used the PHEME [71] dataset of rumors and non-rumors, consisting of Twitter conversations for nine different newsworthy events. The distribution of the dataset is shown in Table 6.2. The basic structure of conversation samples is shown in Figure 6.2. Each conversation sample has a source-tweet and a set of reactions along time, where reactions express their opinions towards the claim contained in the source-tweet.

As shown in Table 6.2, this dataset exhibits severe event-wise and class-wise

TABLE 6.2 The PHEME dataset with nine events

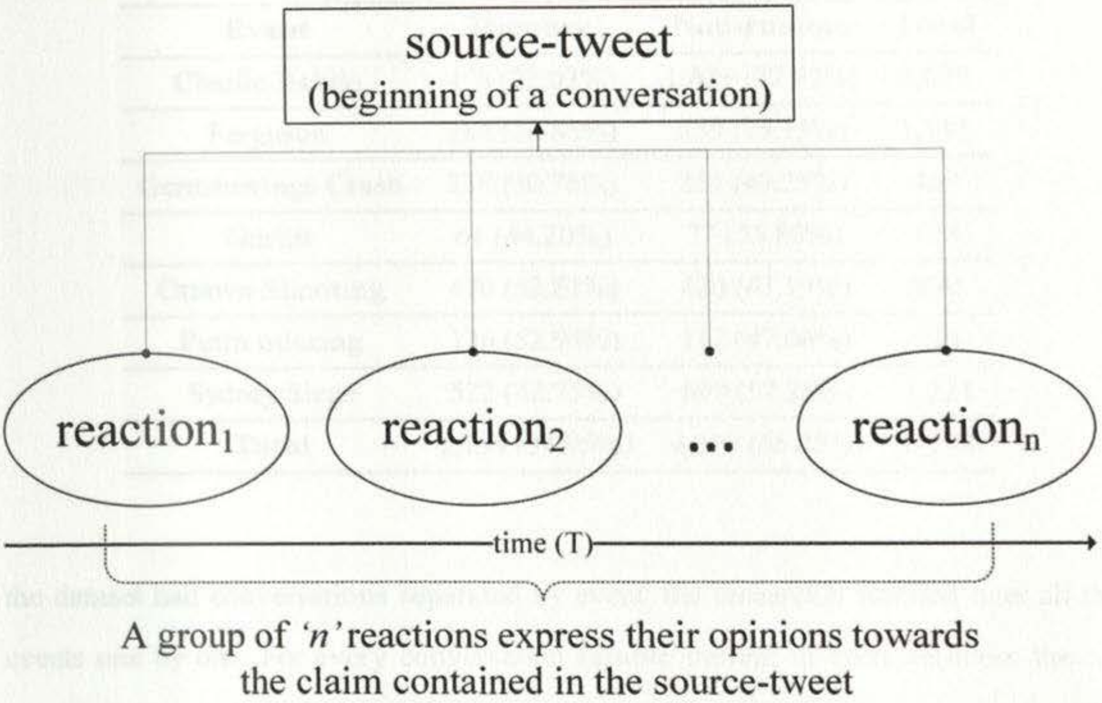
Event	Rumors	Non-rumors	Total
Charlie Hebdo	458	1,621	2,079
Ferguson	284	859	1,143
Germanwings-crash	238	231	469
Ottawa shooting	470	420	890
Sydney siege	522	699	1,221
Gurlitt	61	77	138
Putin missing	126	112	238
Prince Toronto	229	4	233
Ebola Essien	14	0	14
Total	2,402	4,023	6,425

unbalanced nature. For example, event Charlie Hebdo was dominant over all other events present in the dataset in terms of the number of samples causing event-wise unbalance. In general, the number of non-rumor class samples was way more than the number of rumor class samples, which was class-wise unbalance in the dataset.

The experimental analysis excluded events Prince Toronto and Ebola Essien as they had extremely unbalanced proportions of rumors and non-rumors and trimmed down the dataset to seven events. For example, the Ebola Essien event had zero number of non-rumor class samples. The basic statistics of the PHEME dataset with seven events are shown in Table 6.3. Overall, the PHEME seven events dataset had 6, 178 data samples, in which non-rumor class samples were almost double the number of rumor class samples.

6.4.2 Generation of Time-series Data. The researcher explored the temporal features of Twitter data for the timely detection of rumors in social media.

FIGURE 6.2 Structure of a Twitter conversation sample


source-tweet
(beginning of a conversation)
reaction$_1$
reaction$_2$
...
reaction$_n$
time (T)
A group of *'n'* reactions express their opinions towards the claim contained in the source-tweet

Specifically, time-series data were generated by transforming Twitter conversations, where each conversation contains a list of tweets, into time-series vectors that contain reaction counts as features, and fed as input to deep learning models. Each of the Twitter conversation samples present in the PHEME seven events dataset were transformed into a time-series vector for each time interval T, where $T = \{2, 5, 10, 30, 60\}$ minutes. After the successful transformation of all conversations into time-series data, each vector represented one whole conversation. Each of its values was the total reaction counts with respect to T.

Denote $E = \{e_i\}$ the set that contains data of seven events present in the dataset, then, for each event data e_i, c_{ij} is a conversation sample related to that event. As

TABLE 6.3 Distribution of the PHEME dataset with seven events

Event	**Rumors**	**Non-rumors**	**Total**
Charlie Hebdo	458 (22.03%)	1,621 (77.97%)	2,079
Ferguson	284 (24.85%)	859 (75.15%)	1,143
Germanwings Crash	238 (50.75%)	231 (49.25%)	469
Gurlitt	61 (44.20%)	77 (55.80%)	138
Ottawa Shooting	470 (52.81%)	420 (47.19%)	890
Putin missing	126 (52.94%)	112 (47.06%)	238
Sydney Siege	522 (42.75%)	699 (57.25%)	1,221
Total	2,159 (34.95%)	4,019 (65.05%)	6,178

the dataset had conversations separated by event, the researcher iterated over all the events one by one. For every conversation sample present in each iteration, the researcher extracted timestamps of its source-tweet *timeSource* (starting point of the conversation) and its $timeReactions = \{tr_1, tr_2, \cdots, tr_n\}$, which is a set of timestamps of all the reactions corresponding to that source-tweet. For a conversation sample, its length $N(c)$ is determined by,

$$N(c) = \frac{\max(timeReactions) - timeSource}{T} \tag{6.6}$$

Assume c represents a conversation sample, if $(a, b]$ is the time interval limit for the k-th interval, where $k = 1, 2, \cdots, N(c)$ then the total reactions count for that time interval is given by,

$$count_k = \mathbf{card}(Q) \tag{6.7}$$

where $Q \subset timeReactions$ and $Q = \{x \mid x > a \wedge x \leq b\}$, x is the timestamp of a reaction (tweet) and cardinality is the measure of the size of set Q, and the transformed vector representation is as follows:

$$V(c) = [count_k \quad count_{k+1} \quad \cdots \quad count_N] \tag{6.8}$$

The final vector representation of all conversation samples for each event is given by,

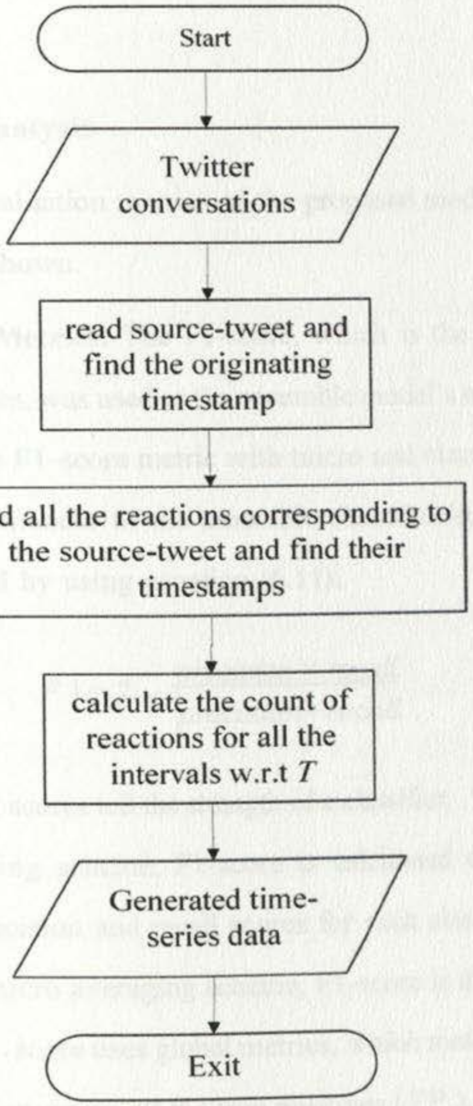
$$e_i = \begin{bmatrix} V(c_1) \\ V(c_2) \\ \vdots \\ V(c_n) \end{bmatrix} \tag{6.9}$$

The flow chart of transforming Twitter conversations into time-series vectors for all combinations of E and T is given in Figure 6.3.

6.4.3 Data Pre-processing. The second step in the data preparation process was to reduce the data sparsity of the time-series data since the vector length of all data samples were decided by the longest conversational sample with respect to T. To tackle this problem, the researcher applied sklearn's dimensionality reduction method called TruncatedSVD [72]. The time-series data were normalized using sklearn's MinMaxScaler [72] and removed duplicate data samples with the same features with different ground truth values.

Finally, The researcher calculated class weights by using sklearn's class _weight [72] library with a *balanced* scheme since the PHEME dataset exhibited an unbalance class nature. Class weights were used in weighting loss functions during the training process, which means the higher weight was given to minority class and lower weight

FIGURE 6.3 The flow chart for transforming Twitter conversations into time-series vectors


Start
Twitter conversations
read source-tweet and find the originating timestamp
read all the reactions corresponding to the source-tweet and find their timestamps
calculate the count of reactions for all the intervals w.r.t *T*
Generated time-series data
Exit

to the majority class. The class weights were computed using the equation below 6.10.

$$class\ weights = \frac{n\_samples}{(n\_classes \times bincount(y))} \tag{6.10}$$

where y represents the actual class labels per sample, $n\ classes$ is the count of unique class label values existing in the dataset, $n\_samples$ is the number of data samples, and *bincount*(y) counts the number of occurrences of each value in y of non-negative integers.

6.5 Experimental Analysis

In this section, the evaluation metrics of the proposed model are discussed, and experimental results are shown.

6.5.1 Evaluation Metrics. The F1-score, which is the weighted average of Precision and Recall scores, was used as the ensemble model's evaluation metric. The researcher considered the F1-score metric with micro and macro averaging schemes for evaluating the performances of the ensemble classification models. In general, the F1-score is calculated by using equation (6.11).

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \qquad (6.11)$$

where precision and recall scores tell the strength of a classifier.

In the macro averaging scheme, F1-score is calculated using equation (6.13). Macro F1-score uses precision and recall scores for each class label and finds their unweighted mean. In a micro averaging scheme, F1-score is determined using equation (6.15), and micro F1-score uses global metrics, which means precision and recall scores are calculated by counting all the true positives (TP), false positives (FP), and false negatives (FN) across all classes.

$$\begin{aligned} P_{macro} &= \frac{{}^{n}_{i=1} p_i}{n} \\ R_{macro} &= \frac{{}^{n}_{i=1} r_i}{n} \end{aligned} \qquad (6.12)$$

$$F1_{macro} = 2 \times \frac{P_{macro} \times R_{macro}}{P_{macro} + R_{macro}} \tag{6.13}$$

$$\begin{aligned} P_{micro} &= \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + FP_i} \\ R_{micro} &= \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + FN_i} \end{aligned} \tag{6.14}$$

$$F1_{micro} = 2 \times \frac{P_{micro} \times R_{micro}}{P_{micro} + R_{micro}} \tag{6.15}$$

In the above equations, P and R represent precision and recall values for a given averaging scheme (macro or micro), i represents a class label, p_i, and r_i are the precision and recall scores for the i^{th} class label. TP_i, FP_i, and FN_i are the true positives, false positives, and false negatives for the i^{th} class label. n is the total number of classes.

6.5.2 Experimental Results. In Table 6.4, the current chapter's best micro-averaged scores of Precision, Recall, and F1 were compared with the previous chapters' best micro-averaged results. The researcher improved the rumor classification performance by a decent margin with the proposed ensemble-based deep learning model in terms of micro-F1. The improvements were 12.5% and 7.9% for Kotteti et al., 2018 and Kotteti et al., 2019, respectively. The rest of this section discusses hyperparameters' influence, such as batch input size and learning rate on the classification model's performance.

6.5.2.1 Fixed Batch Input Size. The testing results when the batch input size is fixed are shown in Tables 6.5 and 6.6. These testing results were the mean micro and macro averaged F1 scores of all events obtained using leave-one-event-out cross-validation across T by varying learning rates for Tables 6.5 and 6.6, respectively.

In Table 6.5, for $T = 2$ and 5 min, the micro-F1 scores of the ensemble I-1

TABLE 6.4 Comparison of current study to Kotteti's previous studies

Metric	Previous studies		Current study
	Kotteti et al., 2018	Kotteti et al., 2019	
Micro-Precision	**0.949**	0.564	0.643
Micro-Recall	0.374	0.564	**0.643**
Micro-F1	0.518	0.564	**0.643**

are better than that of the ensembles I-2 and I-3 across the chosen learning rates. This may be because it had more ensemble diversity than other ensembles (i.e., the presence of base learners designed using Bi-directional RNNs and a model with hybrid architecture that contains a pair of LSTM and GRU layers). In these time intervals, the best scores for the ensemble I-1 were obtained for learning rate $1.50E - 05$. However, when $T = 5$ min, the ensemble I-3 performed poorly across all the chosen learning rates and T.

When $T = 10$ min, the ensemble I-1 outperformed ensembles I-2 and I-3 for learning rates $1.00E-05$ and $1.50E-05$. In the case of learning rate $5.00E-06$, the ensemble I-3 surpassed other ensembles. From this interval onward, the ensemble I-3 started to improve its performance for higher time intervals, w.r.t the chosen learning rates. In this case, the higher the time interval, the better the performance for the ensemble I-3. It is this time interval, where ensembles I-1 and I-2 performed weakly across T.

For $T = 30$ min, the ensemble I-3 achieved maximum micro-F1 score for learning rates $5.00E-06$ and $1.50E-05$. The ensemble I-1 obtained maximum micro-F1 score for learning rate $1.00E-05$. However, the micro-F1 scores of the ensemble I-2 across the learning rates were very low compared to the other ensemble implementations in

TABLE 6.5 Shows the mean micro averaged F1 testing results of all events that were obtained using leave-one-event-out cross-validation across *T* by varying learning rate

Time Interval	Learning Rate	Micro-F1		
		I-1	**I-2**	**I-3**
2 min	5.00E-06	0.53986	0.52801	0.52835
	1.00E-05	0.55231	0.53131	0.53537
	1.50E-05	0.56656	0.53399	0.50439
5 min	5.00E-06	0.43673	0.43128	0.3936
	1.00E-05	0.43809	0.4199	0.39489
	1.50E-05	0.44764	0.41844	0.40408
10 min	5.00E-06	0.43347	0.45515	0.46869
	1.00E-05	0.43594	0.4086	0.41396
	1.50E-05	0.42631	0.40358	0.41814
30 min	5.00E-06	0.55092	0.43766	0.5524
	1.00E-05	0.54492	0.42296	0.53995
	1.50E-05	0.53428	0.45717	0.5394
60 min	5.00E-06	0.55717	0.58966	0.6116
	1.00E-05	0.61769	0.56448	0.62146
	1.50E-05	0.619	0.55943	0.59565

TABLE 6.6 Shows the mean macro averaged F1 testing results of all events that were obtained using leave-one-event-out cross-validation across *T* by varying learning rate

Time Interval	Learning Rate	Macro-F1		
		I-1	**I-2**	**I-3**
2 min	5.00E-06	0.44878	0.38891	0.37119
	1.00E-05	0.46329	0.39504	0.38406
	1.50E-05	0.49849	0.38397	0.39108
5 min	5.00E-06	0.41544	0.35804	0.29844
	1.00E-05	0.42368	0.35859	0.3125
	1.50E-05	0.42362	0.371	0.34597
10 min	5.00E-06	0.34527	0.33345	0.33153
	1.00E-05	0.35471	0.31419	0.31294
	1.50E-05	0.34021	0.32128	0.32075
30 min	5.00E-06	0.37669	0.32084	0.34954
	1.00E-05	0.38528	0.31908	0.36389
	1.50E-05	0.38588	0.31528	0.38077
60 min	5.00E-06	0.42367	0.39506	0.45095
	1.00E-05	0.48757	0.39201	0.45083
	1.50E-05	0.48163	0.38864	0.43825

this time interval.

For $T =$ 60 min, the ensemble I-3 outperformed other ensembles in terms of the maximum micro-F1 score for learning rates $5.00E - 06$ and $1.00E - 05$. It was this time interval where all ensembles obtained their maximum micro-F1 scores across T for all chosen time intervals. The ensemble I-3 achieved the overall best micro-F1 score of 62.1% for the learning rate of $1.00E - 05$. In this time interval, ensembles I-1 and I-3 were better than that of the ensemble I-2. The more diversity of ensemble I-1 and the ensemble I-3 with its base learners having LSTM layers that have better representational power than GRU layers outplayed ensemble I-2 (base learners without LSTM layers). Nonetheless, ensemble I-1 and I-3 results are similar to each other.

From Table 6.6, for $T = 2, 5, 10$ and 30 min, the macro-F1 scores of the ensemble I-1 were better than that of the ensembles I-2 and I-3 across the chosen learning rates. And again, this may be due to the presence of more diversified base learners in the ensemble I-1 that helped to surpass other ensembles. The ensemble I-1 achieved top performance for learning rate $1.50E-05$ when $T = 2$ min, for learning rate $1.00E-05$ when $T = 5$ and 10 min, and for learning rate $1.50E-05$ when $T = 30$ min. However, when $T = 10$ and 30 min, the performance of the ensemble I-1 dropped down across the learning rates compared to $T = 2$ and 5 min. Since for higher time intervals, the lengths of time-series data sequences became shorter, thus maybe overlooking small propagation patterns present in the time-series data. The time interval $T = 10$ min contributed to the overall poor performance of the ensemble I-1. The ensembles I-2 and I-3 performed poorly when $T = 30$ and 5 min, respectively.

For $T = 60$ min, the ensemble I-1 outperformed others in terms of the best macro-F1 score for learning rates $1.00E - 05$ and $1.50E - 05$. When the learning rate was $5.00E - 06$, the ensemble I-3 surpassed other ensembles. Moreover, in this time interval, for ensembles I-1 and I-2, the results were almost on par with the results that they achieved when $T = 2$ min. In this time interval, the ensemble I-3 achieved its overall best performance across T. The overall best macro-F1 score was obtained by the ensemble I-1 when $T = 2$ min and a learning rate of $1.50E - 05$.

Furthermore, ensembles I-1, I-2, and I-3 performed better in terms of both micro-F1 and macro-F1 scores when $T = 60$ min over other time intervals with respect to the chosen learning rates. The only exceptional case was that the ensemble I-1 performed well in terms of the macro-F1 score when $T = 2$ min over other time intervals with respect to the chosen learning rates. In general, both the results showed the fact that the performances of ensembles are better when T is either too low (i.e., 2 min) or too high (i.e., 60 min). This presents us a chance to select a time interval based on the requirement. For example, if early detection is important, pick a lower time interval value; and in the case of effective prediction as a concern, than select a higher time interval value.

In addition to this, the 10 min time interval caused most of the ensemble implementations, particularly the ensemble I-1, to achieve low performance for both micro and macro averaging schemes. This may be due to the propagation patterns extracted based on this time interval value do not have necessary variations such that classification models can take advantage of them. On the other hand, the ensemble I-3 was clearly unhappy with 5 min time interval for both averaging schemes as a lower time interval value was subjected to have high data sparsity, which may be the cause for LSTM based ensemble I-3 to perform weakly in this time interval.

6.5.2.2 Fixed Learning Rate. In the case of a fixed learning rate, the testing results are shown in Tables 6.7 and 6.8. These testing results were the mean micro and macro averaged F1 scores of all events that were obtained using

TABLE 6.7 Shows the mean micro averaged F1 testing results of all events that were obtained using leave-one-event-out cross-validation across *T* by the varying batch input size

Time Interval	Batch Input Size	Micro-F1		
		I-1	I-2	I-3
2 min	16	0.51013	0.48588	0.50378
	32	0.55231	0.53131	0.53537
	64	0.54089	0.51473	0.52323
5 min	16	0.48062	0.4534	0.42757
	32	0.43809	0.4199	0.39489
	64	0.44371	0.46473	0.41045
10 min	16	0.44006	0.43332	0.48341
	32	0.43594	0.4086	0.41396
	64	0.43322	0.42206	0.42
30 min	16	0.51484	0.4542	0.50053
	32	0.54492	0.42296	0.53995
	64	0.55006	0.45109	0.54926
60 min	16	0.5789	0.5619	0.46469
	32	0.61769	0.56448	0.62146
	64	0.57902	0.58571	0.64331

TABLE 6.8 Shows the mean macro averaged F1 testing results of all events that were obtained using leave-one-event-out cross-validation across *T* by varying the batch input size

Time Interval	Batch Input Size	Macro-F1		
		I-1	I-2	I-3
2 min	16	0.44335	0.39693	0.38611
	32	0.46329	0.39504	0.38406
	64	0.43664	0.3692	0.36674
5 min	16	0.4367	0.37274	0.35665
	32	0.42368	0.35859	0.3125
	64	0.38173	0.32	0.32925
10 min	16	0.34352	0.32805	0.35568
	32	0.35471	0.31419	0.31294
	64	0.35004	0.31881	0.32688
30 min	16	0.37424	0.30853	0.3749
	32	0.38528	0.31908	0.36389
	64	0.38205	0.30122	0.34899
60 min	16	0.4266	0.37702	0.3486
	32	0.48757	0.39201	0.45083
	64	0.39252	0.38015	0.47661

leave-one-event-out cross-validation across *T* by varying batch input size for Tables 6.7 and 6.8, respectively.

From Table 6.7, for *T* = 2 and 30 min, the micro-F1 scores of ensembles I-1 and I-3 were very similar and better than that of the ensemble I-2. This may be due to the presence of LSTM layers in both ensembles I-1 and I-3. In ensemble I-2, there was no base learner with an LSTM layer. In these intervals, with respect to the chosen batch input sizes, the ensemble I-1 achieved the best performance.

When *T* = 5 min, the ensemble I-1 outperformed other ensembles for batch input sizes 16 and 32. In this time interval, the ensemble I-2 performed better than that of other ensembles for batch size 64. This time interval is where the ensemble I-3 achieved its least micro-F1 scores across all the batch input sizes and T, which is the same when the batch input size is fixed under a micro-averaging scheme. For *T* = 10 min, the ensemble I-1 obtained the best micro-F1 scores for batch input sizes 32 and 64. The ensemble I-3 achieved a better micro-F1 score over other ensembles for batch input size 16. In this time interval, the ensembles I-1 and I-2 obtained their least micro-F1 scores across all the batch input sizes and *T*.

When *T* = 60 min, the ensemble I-3 outperformed other ensembles for batch input sizes 32 and 64, and the ensemble I-1 performed better for batch input size of 16. It was this time interval where all ensembles obtained their maximum micro-F1 scores. The ensemble I-3 achieved the overall best micro-F1 score of 64.3% for batch input size of 64. In this case, higher time interval helped the ensembles to surpass their lower time interval micro-F1 scores for almost all of the combinations of batch input size and *T* . Again, the results show that LSTM backed ensemble I-3 outplayed other ensembles given the advantages of LSTM such as its good gating mechanism and ability to learn long-term dependencies.

From Table 6.8, for *T* = 2 min, the ensemble I-1 achieved better macro-F1 scores than that of ensembles I-2 and I-3 across all the batch input sizes. In this time interval, the ensemble I-2 obtained its maximum macro-F1 score. When $T = 5$ min, the ensemble I-1 outperformed other ensembles in terms of the macro-F1 score. Lower time intervals had longer time-series sequences that can better represent variations in propagation patterns of rumors and non-rumors than for higher time interval values. However, lower time intervals may have more data sparsity.

For $T = 10$ and 30 min, the ensemble I-1 achieved better performance than that of other ensembles for batch input sizes 32 and 64. However, its performance significantly dropped compared to lower time interval values, and the ensemble I-3 obtained better performance for batch input size 16. The ensemble I-2 became weak when $T = 30$ min, and ensembles I-1 and I-3 started to show some improvement in their performances compared to $T = 10$ min.

In the time interval $T = 60$, the ensemble I-1 better performed over other ensembles for batch input sizes 16 and 32, and the ensemble I-3 obtained the best macro-F1 score for batch input size 64. In this time interval, the ensembles I-1 and I-3 obtained their overall maximum macro-F1 scores (i.e., 48.7% and 47.6% respectively) across T. Overall, the ensembles supported extreme time intervals such as $T = 2$ min and $T = 60$ min to achieve good performance.

In the case of the micro-F1 score, the ensembles I-1, I-2, and I-3 obtained their best micro-F1 scores for $T = 60$ min with respect to the chosen batch input sizes. The only exception was where the micro-F1 score of the ensemble I-3 was lower than its micro-F1 scores when $T = 2$, 10 and 30 min when batch input size was set to 16. This means that $T = 60$ min was appropriate for the effective detection of rumors. In the case of the macro-F1 score, the best performances of the ensembles I-1, I-2, and I-3 were varied for each batch input size across T, which means based on the need, a suitable ensemble model can be selected along with its appropriate time interval value. For instance, if early detection is required, a researcher can pick a lower time interval value; and for effective prediction, the researcher can choose a higher time interval value.

As discussed earlier, the same behavior for 10 min time interval value was seen, which caused most of the ensemble implementations to perform weakly for both micro and macro averaging schemes. In addition to that, the ensemble I-3 again showed low performance in the 5 min time interval under both averaging schemes.

By observing the above results, varying the hyperparameters batch input size and learning rate resulted in producing similar kinds of behavior of the ensembles. In general, when micro-averaging was used, both hyperparameter variations supported higher time interval value for better performance. In the case of macro-averaging scheme was employed, time intervals 2 and 60 min helped ensembles I-1 and I-2 to perform strong. However, the ensemble I-3 still achieved better performance when T = 60 min case only as all ensembles were comfortable with the 60 min time interval. It was best used to achieve better performance regardless of variations in chosen batch input sizes and learning rates. For T = 60, the generated time-series data will have lesser data sparsity than that of other values of T that makes the feature space short for the conversation samples. This may be the reason that all ensembles performed better at higher time intervals. Especially, the ensembles with base learners designed using LSTM layers.

Another key observation was that, for all ensembles, 2 and 60 min time intervals have shown good performance. However, there is no sweet spot for the ensembles for other values of T . This observation was critical in applying the proposed model depending upon the end goal. For instance, if early detection is needed, the researcher can pick small time interval values such as T = 2 min by sacrificing a little amount of prediction performance. If the effective prediction is important, the researcher can easily set the time interval to a higher value, for example, $T = 60$ min.

6.5.3 Discussions. As the PHEME dataset exhibits non-rumor chauvinism (i.e., the dataset contains non-rumor samples almost double the number of rumor samples), adding more rumor samples to the dataset will help in improving its class balance. It may help classification models to perform better classification. When compared to [73], it was noticed that increase in maximum micro and macro averaged F1 scores with the addition of two extra events (Gurlitt and Putin Missing events) to the PHEME five event dataset. In the case of fixed batch input size, improvement was 5.7% and 0.4% for micro and macro averaging schemes. When the learning rate was constant, the improvement was 7.9% for the micro averaging scheme. However, the maximum macro F1 score was dropped by 0.7%. Moreover, even though events Gurlitt and Putin Missing were included in the seven events PHEME dataset, only the Putin Missing event contributed to adding a greater number of rumor samples slightly to the dataset than Gurlitt event, which was also a supporter of the non-rumor group.

In addition to this, the data pre-processing method combined with the proposed model helped in improving Kotteti's previous best score in [73] and achieved a 64.3% micro F1 score, which is almost an 8% improvement. The performance improvement may seem small, but it is non-trivial to gain huge performances using this dataset. For instance, in [74], extensive feature engineering was conducted for a rumor detection problem on social media using the PHEME dataset with five events. The authors focused on extracting complex features such as content-based and social features, and their best F1 scores were 0.606 and 0.339 for content-based and social features, respectively. When both feature sets are jointly used, the F1 score reached 0.607, which was a 0.1% improvement. Again, extensive feature engineering needs a long time to be completed as some of the features may not be readily available. Having complex feature sets challenge hardware resources, which also increases computational complexity that directly impacts training times of classification models. Nevertheless, given the condition that information spreads rapidly on social media, time-taking labor-intensive feature engineering may not be appropriate.

6.6 Concluding Remarks for the Chapter

In this chapter, a data pre-processing method and an ensemble model were proposed for the timely detection of rumors on social media. The proposed data pre-processing method transformed Twitter conversations into time-series vectors based on the tweet creation timestamps, which can be extracted and processed without delay. Furthermore, the generated time-series data was of pure numeric type, which reduced feature set complexity and, in turn, helped in reducing the computational complexity of classification models during their training process. The proposed ensemble model contained several classification models with simplistic yet effective architectures designed using deep learning techniques. By combining the proposed data pre-processing method with the ensemble model, better performance of rumor detection was demonstrated in the experiments using the PHEME dataset. For instance, the researcher improved the classification performance by 7.9% in terms of micro F1 score compared to the baselines.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

In this study, a data imputation preprocessing method was proposed for enhancing fake news detection using machine learning. The proposed method aimed to mitigate the missing values problem in the raw data using data imputation methods. The researcher utilized scikit-learn's Imputer with a "mean" strategy for missing numerical values [26], and the CategoricalImputer[1] method was applied to categorical missing data. Experimental results showed that traditional machine learning models combined with the proposed data preprocessing method outperformed baselines.

Given the condition that information spreads rapidly in social media, verifying news credibility becomes a challenging task. Fake news detection in social media is a well-known problem; many of the existing works explored various features from social media posts to improve fake news detection accuracy. Fake news in social media disseminates very fast. This created a necessity for developing novel methods/techniques to detect fake news in its early stages of proliferation. In this study, a multiple time-series data analysis model was also proposed. It relied only on the temporal characteristics of social media (Twitter) data for the early detection of fake news in social media.

With the proposed time-series model, the researcher simplified the feature extraction process, which significantly reduced the time required for ML models' training and testing processes as well as reduced the computational complexity of ML models by taking advantage of pure numerical time-series data. The experimental results showed that the generated time-series data used with the GaussianNB classifier achieved a high Precision score of 94%.

Furthermore, deep learning techniques combined with time-series data generated by using only the temporal features of tweets showed better performance results than traditional machine learning models. The ensemble-based deep learning rumor detection model achieved top performance, especially.

In summary, the main contributions of this study are:

- A data imputation preprocessing method was proposed for mitigating missing values in the raw data and used with traditional ML models improved fake news detection accuracy and outperformed state-of-the-art methods [27].
- A multiple time-series data analysis model for the early detection of fake news in social media was proposed.
- The experimental results showed that using the generated time-series data, simplified the feature extraction process, reduced ML models' computational complexity and their training and testing times, and achieved a high Precision score of 94% with GaussianNB classifier.
- The researcher proposed a deep learning-based classification model that relied entirely on tweets' propagation patterns for the detection of false information in social media.
- Experimental results showed improvement in the micro-averaged F1 score by 4.6%, compared to baselines [28].
- The researcher proposed an ensemble-based deep learning classification model for rumor detection on social media. With that, the classification performance was improved by 7.9% in terms of a micro F1 score compared to the baselines.

7.2 Future Work

This study addressed issues related to handling missing data to enhance fake news detection and early detection of false information on social media. These results can be enhanced by employing good quality dataset(s) than that of the datasets used in this study. However, a dataset with good quality in terms of size and ground-truth balance was not currently available. It would be beneficial to collect a high-quality dataset for fake news detection in the future.

REFERENCES

[1] N. Kshetri and J. Voas, "The economics of "fake news"," *IT Professional*, vol. 19, pp. 8–12, November 2017.

[2] M. Granik and V. Mesyura, "Fake news detection using naive bayes classifier," in *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, pp. 900–903, May 2017.

[3] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *CoRR*, vol. abs/1708.01967, 2017.

[4] A. Zubiaga, M. Liakata, R. Procter, K. Bontcheva, and P. Tolmie, "Towards detecting rumours in social media," in *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[5] A. Zubiaga, M. Liakata, and R. Procter, "Learning reporting dynamics during breaking news for rumour detection in social media," *CoRR*, vol. abs/1610.07363, 2016.

[6] V. Qazvinian, E. Rosengren, D. R. Radev, and Q. Mei, "Rumor has it: Identifying misinformation in microblogs," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, (Stroudsburg, PA, USA), pp. 1589–1599, Association for Computational Linguistics, 2011.

[7] A. Zubiaga, A. Aker, K. Bontcheva, M. Liakata, and R. Procter, "Detection and resolution of rumours in social media: A survey," *CoRR*, vol. abs/1704.00656, 2017.

[8] T. Lan, C. Li, and J. Li, "Mining semantic variation in time series for rumor detection via recurrent neural networks," in *HPCC/SmartCity/DSS*, pp. 282–

289, IEEE, 2018.

[9] T. Hashimoto, T. Kuboyama, and Y. Shirota, "Rumor analysis framework in social media," in *TENCON 2011-2011 IEEE Region 10Conference*, pp. 133–137, IEEE, 2011.

[10] N. Ruchansky, S. Seo, and Y. Liu, "Csi: A hybrid deep model for fake news detection," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 797–806, ACM, 2017.

[11] N. Xu, G. Chen, and W. Mao, "Mnrd: A merged neural model for rumor detection in social media," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, IEEE, 2018.

[12] Z. Zhao, P. Resnick, and Q. Mei, "Enquiring minds: Early detection of rumors in social media from enquiry posts," in *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, (Republic and Canton of Geneva, Switzerland), pp. 1395–1405, International World Wide Web Conferences Steering Committee, 2015.

[13] C. Buntain and J. Golbeck, "Automatically identifying fake news in popular twitter threads," in *2017 IEEE International Conference on Smart Cloud (SmartCloud)*, pp. 208–215, Nov 2017.

[14] L. Wu, F. Morstatter, X. Hu, and H. Liu, "Mining misinformation in social media," *Big Data in Complex and Social Networks*, pp. 123–152, 2016.

[15] J. Ma, W. Gao, and K.-F. Wong, "Detect rumors in microblog posts using propagation structure via kernel learning," in *ACL*, 2017.

[16] J. Ma, W. Gao, P. Mitra, S. Kwon, B. J. Jansen, K.-F. Wong, and M. Cha, "Detecting rumors from microblogs with recurrent neural networks.," in *IJCAI*,

pp. 3818–3824, 2016.

[17] T. Hashimoto, T. Kuboyama, and Y. Shirota, "Rumor analysis framework in social media," in *TENCON 2011 - 2011 IEEE Region 10 Conference*, pp. 133–137, Nov 2011.

[18] Y. Chang, M. Yamada, A. Ortega, and Y. Liu, "Ups and downs in buzzes: Life cycle modeling for temporal pattern discovery," in *2014 IEEE International Conference on Data Mining*, pp. 749–754, IEEE, 2014.

[19] Y. Chang, M. Yamada, A. Ortega, and Y. Liu, "Lifecycle modeling for buzz temporal pattern discovery," *ACM Transactions on Knowledge Discovery from Data*, vol. 11, no. 2, 2016.

[20] S. Kwon, M. Cha, K. Jung, W. Chen, and Y. Wang, "Prominent features of rumor propagation in online social media," in *2013 IEEE 13th International Conference on Data Mining*, pp. 1103–1108, Dec 2013.

[21] S. Kwon and M. Cha, "Initial small data reveal rumor traits via recurrent neural networks," *Journal of KIISE*, vol. 44, no. 7, pp. 680 – 5, 2017/07/.

[22] T. N. Nguyen, C. Li, and C. Niederee, "On early-stage debunking rumors on twitter: Leveraging the wisdom of weak learners," vol. pt.II, (Cham, Switzerland), pp. 141 – 58, 2017//.

[23] J. Ma, W. Gao, Z. Wei, Y. Lu, and K.-F. Wong, "Detect rumors using time series of social context information on microblogging websites," vol. 19-23-Oct-2015, (Melbourne, VIC, Australia), pp. 1751 – 1754, 2015.

[24] J. Poulos and R. Valle, "Missing Data Imputation for Supervised Learning," *ArXiv e-prints*, Oct. 2016.

[25] R. Procter, J. Crump, S. Karstedt, A. Voss, and M. Cantijoch, "Reading the riots: what were the police doing on twitter?," *Policing and Society*, vol. 23, no. 4, pp. 413–436, 2013.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[27] W. Y. Wang, ""liar, liar pants on fire": A new benchmark dataset for fake news detection," *CoRR*, vol. abs/1705.00648, 2017.

[28] C. M. M. Kotteti, X. Dong, and L. Qian, "Multiple time-series data analysis for rumor detection on social media," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 4413–4419, IEEE, 2018.

[29] Z. Zhao, P. Resnick, and Q. Mei, "Enquiring minds: Early detection of rumors in social media from enquiry posts," in *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, (Republic and Canton of Geneva, Switzerland), pp. 1395–1405, International World Wide Web Conferences Steering Committee, 2015.

[30] J. Ma, W. Gao, Z. Wei, Y. Lu, and K.-F. Wong, "Detect rumors using time series of social context information on microblogging websites," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 1751–1754, ACM, 2015.

[31] T. Chen, X. Li, H. Yin, and J. 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*, pp. 40–52, Springer, 2018.

[32] Y. Liu and S. Xu, "Detecting rumors through modeling information propagation networks in a social media environment," *IEEE Transactions on computational social systems*, vol. 3, no. 2, pp. 46–62, 2016.

[33] Y. Qin, W. Dominik, and C. Tang, "Predicting future rumours," *Chinese Journal of Electronics*, vol. 27, no. 3, pp. 514–520, 2018.

[34] Z. Wang, Y. Guo, J. Wang, Z. Li, and M. Tang, "Rumor events detection from chinese microblogs via sentiments enhancement," *IEEE Access*, 2019.

[35] G. Liang, W. He, C. Xu, L. Chen, and J. Zeng, "Rumor identification in microblogging systems based on users' behavior," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 3, pp. 99–108, 2015.

[36] S. Kwon, M. Cha, K. Jung, W. Chen, and Y. Wang, "Prominent features of rumor propagation in online social media," in *2013 IEEE 13th International Conference on Data Mining*, pp. 1103–1108, IEEE, 2013.

[37] K. Wu, S. Yang, and K. Q. Zhu, "False rumors detection on sina weibo by propagation structures," in *2015 IEEE 31st international conference on data engineering*, pp. 651–662, IEEE, 2015.

[38] L. Wu and H. Liu, "Tracing fake-news footprints: Characterizing social media messages by how they propagate," 2018.

[39] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[40] J. Kaiser, "Dealing with missing values in data," vol. 5, pp. 42–51, 01 2014.

[41] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min. Knowl. Discov.*, vol. 2, pp. 121–167, June 1998.

[42] A. Zubiaga, G. Wong Sak Hoi, M. Liakata, and R. Procter, "Pheme dataset of rumours and non-rumours," Oct 2016.

[43] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[44] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[45] F. Chollet, *Deep Learning with Python*. Greenwich, CT, USA: Manning Publications Co., 1st ed., 2017.

[46] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[47] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[48] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *2013 IEEE workshop on automatic speech recognition and understanding*, pp. 273–278, IEEE, 2013.

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

[50] J. Zhao, N. Cao, Z. Wen, Y. Song, Y.-R. Lin, and C. Collins, "# fluxflow: Visual analysis of anomalous information spreading on social media," *IEEE transactions on visualization and computer graphics*, vol. 20, no. 12, pp. 1773–1782, 2014.

[51] Z.-H. Zhou, "Ensemble learning," *Encyclopedia of biometrics*, pp. 411–416, 2015.

[52] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[53] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[54] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[55] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[56] D. Gaikwad and R. C. Thool, "Intrusion detection system using bagging ensemble method of machine learning," in *2015 International Conference on Computing Communication Control and Automation*, pp. 291–295, IEEE, 2015.

[57] G. Tuysuzoglu, N. Moarref, and Y. Yaslan, "Ensemble based classifiers using dictionary learning," in *2016 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 1–4, IEEE, 2016.

[58] H. Li, J. Wang, T. Gao, Y. Lu, and Z. Su, "Accurate prediction of the optical absorption energies by neural network ensemble approach," in *2010 Fifth International Conference on Frontier of Computer Science and Technology*, pp. 503–507, IEEE, 2010.

[59] B. Linghu and B. Sun, "Constructing effective svm ensembles for image classification," in *2010 Third International Symposium on Knowledge Acquisition and Modeling*, pp. 80–83, IEEE, 2010.

[60] X.-D. Zeng, S. Chao, and F. Wong, "Optimization of bagging classifiers based on sbcb algorithm," in *2010 International Conference on Machine Learning and Cybernetics*, vol. 1, pp. 262–267, IEEE, 2010.

[61] Y. Chen, Y. Wang, Y. Gu, X. He, P. Ghamisi, and X. Jia, "Deep learning ensemble for hyperspectral image classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

[62] M. M. Islam, X. Yao, S. S. Nirjon, M. A. Islam, and K. Murase, "Bagging and boosting negatively correlated neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 3, pp. 771–784, 2008.

[63] L. Shi, L. Xi, X. Ma, and X. Hu, "Bagging of artificial neural networks for bankruptcy prediction," in *2009 International Conference on Information and Financial Engineering*, pp. 154–156, IEEE, 2009.

[64] I. Fakhruzi, "An artificial neural network with bagging to address imbalance datasets on clinical prediction," in *2018 International Conference on Information and Communications Technology (ICOIACT)*, pp. 895–898, IEEE, 2018.

[65] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[66] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[67] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[68] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[69] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *2013 IEEE workshop on automatic speech recognition and understanding*, pp. 273–278, IEEE, 2013.

[70] J. Heaton, *Introduction to neural networks with Java.* Heaton Research, Inc., 2008.

[71] E. Kochkina, M. Liakata, and A. Zubiaga, "All-in-one: Multi-task learning for rumour verification," *arXiv preprint arXiv:1806.03713*, 2018.

[72] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[73] C. M. M. Kotteti, X. Dong, and L. Qian, "Rumor detection on time-series of tweets via deep learning," in *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*, pp. 1–7, IEEE, 2019.

[74] A. Zubiaga, M. Liakata, and R. Procter, "Learning reporting dynamics during breaking news for rumour detection in social media," *arXiv preprint arXiv:1610.07363*, 2016.

CURRICULUM VITAE

CHANDRA MOULI MADHAV KOTTETI

E-mail: chandra.mankatha@gmail.com

EDUCATION

Ph.D. in Electrical Engineering, Prairie View A&M University, Prairie View, Texas, USA, August 2020.

M.S. in Applied Computer Science, Northwest Missouri State University, Maryville, Missouri, USA, December 2014.

B.Tech. in Electrical and Electronics Engineering, Koneru Lakshmaiah College of Engineering, Guntur, Andhra Pradesh, India, April 2012.

WORK EXPERIENCE

Graduate Research Assistant, CREDIT Center, Prairie View A&M University, Prairie View, Texas 77446, USA, October 2016 - May 2020.

Test Automation Developer, TEKSYSTEMS GLOBAL SERVICE, LLC, Arizona, USA, April 2015 - January 2016.

Graduate Teaching Assistant, Northwest Missouri State University, Missouri, USA, May 2014 - December 2014.

PUBLICATIONS

C. Kotteti, X. Dong and L. Qian, "Ensemble Deep Learning on Time-Series Representation of Tweets for Rumor Detection in Social Media", *to be submitted*, 2020.

C. Kotteti, X. Dong and L. Qian, "Rumor Detection on Time-Series of Tweets via Deep Learning", *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*, 2019.

C. Kotteti, X. Dong and L. Qian, "Multiple Time-Series Data Analysis for Rumor Detection on Social Media", *2018 IEEE International Conference on Big Data (Big Data)*, 2018.

C. Kotteti, X. Dong, N. Li and L. Qian, "Fake News Detection Enhancement with Data Imputation", *2018 IEEE (DASC/ PiCom/ DataCom/ CyberSciTech)*, 2018.