A Mini Project on

“RUMOR DETECTION ON ONLINE SOCIAL NETWORKS “

A Project Report submitted in partial fulfillment of the requirements for

The award of the degree of

###### **BACHELOR OF TECHNOLOGY**

###### In

**COMPUTER SCIENCE AND ENGINEERING**

**(AL&ML)**

By

**T. SAI SREENATH 21861A6647**

**D. DIVYA 21861A6614**

**M. AKSHITHA 21861A6634**

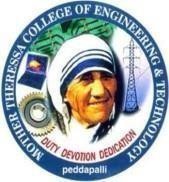
**K. SAHITHI 21861A6628**

**K. SHIVACHARAN 21861A6627**

Under the Guidance of

###### **Mr. BURLA SRINIVAS**

**Assistant Professor**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**MOTHER THERESSA COLLEGE OF ENGINEERING & TECHNOLOGY**

**(Approved by AICTE, New Delhi & Affiliated to JNTUH Hyderabad)**

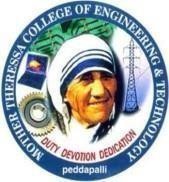
**Peddapalli (Dist.), Telangana-505174**

**2021-2025**

**DEPARTMENT OF COMUPTER SCIENCE ENGINEERING (AI&ML)**

**MOTHER THERESSA COLLEGE OF ENGINEERING &TECHNOLOGY**

**(Affiliated to JNTU Hyderabad Approved by AICTE)**

 PP colony, Peddapally-505172

**CERTIFICATE**

This is to certified to that the project report entitled **“Rumor Detection On Social Media”** is being submitted by team **T. SAI SREENATH (21861A6647), D. DIVYA (21861A6614), M. AKSHITHA (21861A6634),K.SAHTHI(21861A6628),K.SHIVACHARAN(21861A6627)**,in partial fulfillment of the requirements for the award of the Degree of Master of Technology in CSE with specialization “**COMPUTER SCIENCE AND ENGINEERING(AL&ML)**” is Bonafide work performed by Team. The work embodied in this project report has not been submitted to any other institution for the award of any degree.

**INTERNAL GUIDE HEAD OF THE DEPT.**

**BURLA SRINIVAS Mrs. SHABHANA**

Assistant professor. Assistant Professor

**PRINCIPAL EXTERNAL EXAMINER**

**Dr.T. SRINIVAS**

**PhD, MTech, MBA, MISTE, MIAENG**

**DECLARATION**

I am hereby declaring that the entire work embodied in this project entitled **“Rumor Detection On Social Media”** has been carried out by Team. No part of it has been submitted for the award of any Degree or Diploma at any other University or Institution. I further declare that this project dissertation is based on my work carried out at “**MOTHER THERESSA COLLEGE OF ENGINEERING & TECHNOLOGY”** in the final year B. Tech course.

DATE: T. SAI SREENATH 21861A6647

D. DIVYA 21861A6614

M.AKSHITHA 21861A6634

K. SAHITHI 21861A6628

K. SHIVA CHARAN 21861A6627

# ACKNOWLEDGEMENT

The Satisfaction that accomplishes the successful completion of any task would be in complete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts with success.

It is my privilege and pleasure to express my profound sense of respect, gratitude and indebtedness of my guide **Mr. BURLA SRINIVAS** Assistant Professor, Department of CSE(AI&ML), MOTHER THERESSA COLLEGE OF ENGG. & TECH, for his constant guidance, inspiration, and constant encouragement throughout this project work.

I wish to express my deep gratitude to **Mr. BURLA SRINIVAS**, Assistant Professor & Head of CSE Department, **MOTHER THERESSA COLLEGE OF ENGG &TECH,** for his cooperation and encouragement, in addition to providing necessary facilities throughout the project work.

I sincerely extend my thanks to **Dr.T. SRINIVAS,** Principal of project work. **“MOTHER THERESSA COLLEGE OF ENGG & TECH”**, for his valuable suggestions in completing this project.

I would like to thank all the staff and all my friends for their good wishes, their helping and constructive criticism, which led the successful completion of this project.

Finally, I thank all those who directly and indirectly helped to me in this

regard; I apologize for not list.

T. SAI SREENATH 21861A6647

D. DIVYA 21861A6614

M.AKSHITHA 21861A6634

K. SAHITHI 21861A6628

K. SHIVA CHARAN 21861A6627

|  |  |
| --- | --- |
| **INDEX** |  |
| **TITLE** | **PAGE NO** |
| **ACKNOWLEDGEMENT** | **I** |
| **ABSTRACT** | **II** |
| **LIST OF FIGURES** | **III-IV** |
| **Chapter 1: Introduction** | **1-3** |
| 1.1About Project | 1 |
| 1.2 Existing System with Drawbacks | 2 |
| 1.3 Proposed System with Features | 3 |
| **Chapter 2: Literature Survey** | **4-7** |
| 2.1 Literature Review | 4-6 |
| 2.2 Problem Definition | 7 |
| **Chapter 3: Analysis** | **8-10** |
| 3.1 Hardware & Software Requirements | 8 |
| 3.2 Functional Requirements | 9 |
| 3.2 Non-Functional Requirements | 9 |
| 3.3 Module Description | 10 |
| **Chapter 4: Design** | **11-15** |
| 4.1 Architecture Diagram | 11 |
| 4.2 Data Flow Diagram | 12-15 |
| **Chapter 5: Implementation** | **16-21** |
| 5.1 Technologies Used | 16-21 |

[Chapter 6: Testing 22-26](#_TOC_250006)

* 1. : Model Evaluation 22
  2. [: Deep Learning Model Testing (LSTM) 24](#_TOC_250005)
  3. [: Performance Testing 25](#_TOC_250004)
  4. [: Model Evaluation Metrics 26](#_TOC_250003)

[Chapter 7: Results 27-38](#_TOC_250002)

[Chapter 8: Future Scope & Enhancements 39](#_TOC_250001)

Chapter 9: Conclusion 40

[References 41](#_TOC_250000)

# ABSTRACT

The proliferation of online social networks has led to an unprecedented spread of information, including rumors. These rumors can have significant social and economic consequences. This paper presents a comprehensive review of machine learning and deep learning techniques applied to rumor detection in online social networks. We discuss traditional machine learning algorithms such as Naive Bayes, Support Vector Machines, and Random Forests, as well as deep learning models like Recurrent Neural Networks (RNNs). The paper also explores various feature engineering techniques used to extract relevant information from social media data, including text content, user behavior, and network structure. Furthermore, we highlight the challenges and limitations associated with rumor detection and discuss potential future research directions. By providing a systematic overview of existing methods and identifying emerging trends, this paper aims to contribute to the development of more effective rumor detection

|  |  |  |
| --- | --- | --- |
| **TITLE** |  | **PAGE NO** |
| Fig 4.1 Block Diagram |  | 11 |
| Fig 4.2.1 Dataflow Diagram |  | 12 |
| Fig 4.2.2 Data flow Diagram |  | 14 |
| Fig 7.1: Sample Data |  | 27 |
| Fig 7.2: Data Frame |  | 28 |
| Fig 7.3: Columns |  | 28 |
| Fig 7.4: Dataset Dimensions |  | 28 |
| Fig 7.5: Numerical Data |  | 28 |
| Fig 7.6: Test Results |  | 29 |
| Fig 7.7: Logistic Regression |  | 29 |
| Fig 7.8: decision Tree |  | 29 |
| Fig 7.9: Random Forest |  | 29 |
| Fig 7.1: Confusion Matrix |  | 30 |
| Fig 7.2: Line Plot |  | 30 |
| Fig 7.3: Confusion Matrix |  | 31 |
| Fig 7.4: ROC Curve |  | 31 |
| Fig 7.5: Distribution Plot |  | 32 |
| Fig 7.6: Box Plot |  | 32 |
| Fig 7.7: Correlation Heatmap |  | 33 |
| Fig 7.8: Pair Plot |  | 34 |
| Fig 7.9: Histogram Plot |  | 35 |

Fig 7.10: Scatter Plot 36

Fig 7.11: PCA Plot 36

Fig 7.12: Box Plot 37

Fig 7.13: 3D Scatter Plot 37

Fig 7.14: Violin Plot 38

# CHAPTER 1 INTRODUCTION

* 1. **ABOUT PROJECT:**

Rumor detection on social networks has emerged as a critical area of research and development. The rapid dissemination of information, coupled with the vast volume of content generated daily, makes it increasingly difficult to identify and mitigate the impact of rumors effectively. Misleading information can undermine public trust, influence elections, incite violence, and lead to a range of other negative consequences. This project aims to address these challenges by developing advanced techniques for detecting rumors on social networks. We will explore various methods, including machine learning and deep learning techniques to identify and classify rumors with greater accuracy. By leveraging these technologies, our goal is to enhance the reliability of information on social networks and contribute to a more informed and responsible online community.

Machine learning and deep learning techniques have emerged as promising approaches for addressing the challenge of rumor detection in online social networks. These techniques leverage the power of algorithms and models to analyze vast amounts of data, identify patterns, and make predictions. By automating the process of rumor detection, these methods can help to improve the accuracy and efficiency of identifying and mitigating the spread of false information.

This project report provides a comprehensive overview of machine learning and deep learning techniques applied to rumor detection in online social networks. We explore various methods, including traditional machine learning algorithms and state-of-the-art deep learning models, and discuss their strengths and limitations. Additionally, we delve into feature engineering techniques used to extract relevant information from social media data and highlight the challenges and future directions in rumor detection research.

# EXISTING SYSTEM WITH DRAWBACKS:

While several systems have been proposed for rumor detection in online social networks, they often face challenges and limitations. These systems typically rely on machine learning and deep learning techniques to analyze various features extracted from social media data, such as text content, user behavior, and network structure.

# LIMITATIONS:

* + - **Data limitations:** Insufficient labelled data and class imbalance can hinder the training and performance of these systems.
    - **Feature engineering:** feature relevance can be challenging to determine.
    - **Model complexity:** Complex models, especially deep neural networks, can be prone to overfitting and may lack interpretability.
    - **Real-time detection:** Many systems struggle to detect rumors in real time due to latency and scalability issues.
    - **Contextual understanding:** Existing systems may not adequately consider contextual factors, such as the source of the information, user reputation, and broader social and political context.

# PROPOSED SYSTEM WITH FEATURES:

Our proposed system seeks to overcome the shortcomings of existing rumor detection systems by employing sophisticated machine learning and deep learning algorithms, coupled with innovative feature engineering and contextual understanding. This system is designed to enhance accuracy by more precisely categorizing rumors and truthful information, improve efficiency by reducing computational burden and latency, enhance interpretability by making predictions easier to comprehend and transparent, and consider contextual factors to better understand rumor propagation.

# FEATURES:

* + - **Hybrid Deep Learning Model:** The system will use a combination of RNNs and CNNs with an attention mechanism and a hierarchical architecture.
    - **Enhanced Feature Engineering:** extract contextual and temporal features, and use a knowledge graph for contextual understanding.
    - **Real-time Detection:** The system will use a streaming architecture, distributed processing, and incremental learning for real-time detection.
    - **Interpretability:** The system will employ explainable ML and DL techniques and visualization tools for better understanding.

# CHAPTER 2 LITERATURE SURVEY

* 1. **LITERATURE REVIEW:**

Rumor detection in social networks has garnered significant attention, particularly with the rapid growth of user-generated content. Early models largely relied on traditional **Machine Learning (ML) Techniques**, such as **Naive Bayes**, **Random Forest**, and **Support Vector Machines (SVM)**. These models typically used **handcrafted features** like text content, user behaviors, and network structures to detect rumors. However, these systems face challenges such as scalability, interpretability, and reliance on feature engineering.

Deep learning models, including **Convolutional Neural Networks (CNNs)** and **Recurrent Neural Networks (RNNs)**, have emerged as more promising approaches, automatically learning features from data without extensive manual intervention. For instance, CNNs are effective in identifying textual patterns, while RNNs and **Long Short-Term Memory (LSTM)** networks excel in handling sequential data, such as conversations or retweets. Despite these improvements, several limitations persist in both ML and DL approaches.

# Drawbacks of Existing Systems:

* + 1. **Data Limitations**: One significant drawback is the lack of **sufficient labelled data** for training accurate models. Many social media platforms face **class imbalance**, where there are more non-rumor instances than rumors, making it difficult to detect the latter efficiently Additionally, **insufficient training data** can negatively impact model generalization, especially for rare rumors.
    2. **Over-Reliance on Feature Engineering**: In earlier ML models, the selection of features, such as **user credibility**, post characteristics, and **propagation features**, is often manual, which can result in the exclusion of important hidden features Moreover, these models lack the flexibility to adapt to different types of rumors or evolving patterns in social media.
    3. **Real-Time Detection Challenges**: Many systems fail to deliver **real-time detection** due to computational complexity and latency issues. The necessity for **batch processing** and high-volume data limits.

4

* + 1. **Lack of Contextual Understanding**: A significant limitation is the inability to account for **contextual factors**, such as the source's credibility, historical reputation, and external events. Current models do not adequately understand broader social or political contexts, which are essential for accurately detecting rumors [5]
    2. **Model Complexity and Overfitting**: While deep neural networks, especially **LSTMs and CNNs**, have achieved success, they are often prone to **overfitting** due to model complexity. Furthermore, these models can be less interpretable, making it difficult to explain why a certain piece of information was classified as a rumor.[8]

# Features Missing in Existing Systems:

* **Multilingual Capability**: Most existing models are designed to detect rumors in English, overlooking other languages and **multilingual rumors**.
* **Cross-Platform Scalability**: Models are often **platform-specific** (e.g., designed for Twitter) and fail to generalize to other social media platforms, which have different user behaviours and post structures.
* **Temporal and Contextual Features**: Limited focus on temporal patterns, such as the evolution of rumors over time, and contextual understanding (e.g., **user influence**, **external events**).
* **Explainability**: Many deep learning models, while effective, lack **interpretability**, making it difficult for users to understand how the model arrived at a particular classification.

# Proposed Features and Advancements:

1. **Hybrid Deep Learning Models**: The proposed system will employ **hybrid models**, combining **CNNs and RNNs** with an **attention mechanism** to capture complex, nonlinear patterns in rumor propagation

The **hierarchical architecture** will allow for **multimodal learning**, where both team and network features are processed together for more accurate detection.

5

1. **Temporal feature extraction**: This includes leveraging **knowledge graphs** to enhance **contextual understanding** and incorporating **propagation patterns** to detect rumors in the early stages, even before they gather significant traction
2. **Real-Time Detection**: To address latency issues, the system will implement **streaming architectures** and **distributed processing** techniques, enabling it to process large-scale social media data in near real-time

**Incremental learning** will allow models to update continuously as new data arrives, enhancing real-time capabilities.

1. **Explainable ML and DL**: By integrating **explainable AI techniques**, the proposed system will provide **transparency** in its decision-making process. Users will have access to **visualization tools** that show how the model classifies information as rumor or non-rumor, improving **interpretability** and trust.

# PROBLEM DEFINITION:

In the age of social media, rumors spread rapidly, often leading to significant societal and economic consequences. The timely and accurate detection of rumors is crucial to mitigate their impact. This research aims to develop robust machine learning and deep learning algorithms that can effectively identify and classify rumors in real-time based on the content and context of posts on online social networks. By analyzing large datasets of social media posts, including text, images, and metadata, the goal is to create models that can accurately distinguish between factual and false information. The research will explore various machine learning and deep learning techniques, such as natural language processing, sentiment analysis, graph analysis, and deep neural networks, to capture the complex patterns and characteristics of rumor propagation. The developed models will be evaluated using relevant metrics like accuracy, precision, recall, and F1-score to assess their performance in detecting rumors. Additionally, the research will address the challenges posed by the dynamic nature of rumors, the noisy nature of social media data, the computational complexity of processing large datasets, and the ethical implications of rumor detection systems. Ultimately, the goal is to create a reliable and efficient rumor detection system that can help combat the spread of misinformation and promote fact-based information dissemination on social media platforms.

# CHAPTER 3 ANALYSIS

* + **HARDWARE AND SOFTWARE REQUIREMENTS Hardware Requirements:**

Processor: Intel Core i5 or higher

RAM: 16 GB or more for efficient real-time data processing

Storage: 500 GB SSD to manage large datasets

GPU: NVIDIA GTX 1080 or better for deep learning model training

# Software Requirements:

Operating System: Windows 10 or Ubuntu 20.04

Programming Languages: Python, TensorFlow, PyTorch for deep learning models

# FUNCTIONAL REQUIREMENTS AND NON-FUNCTIONAL REQUIREMENTS Functional Requirements:

# Rumor Detection:

# The system must classify social media posts into rumors or non-rumors using machine learning and deep learning algorithms.

# Real-Time Processing:

# Capable of processing social media data streams in real-time for early rumor detection.

# Multilingual Support:

# Support rumor detection in multiple languages.

# Visualization:

# Present results via dashboards with explanations of detection results.

# Non-Functional Requirements:

# Scalability: Must handle large volumes of social media data from various platforms.

# Performance: Ensure low latency for real-time rumor detection.

# Reliability: Must consistently provide accurate results across different datasets.

# Usability: The system should be user-friendly, providing easily interpretable results and visualizations.

# Security: Ensure data privacy, especially for personal user information.

9

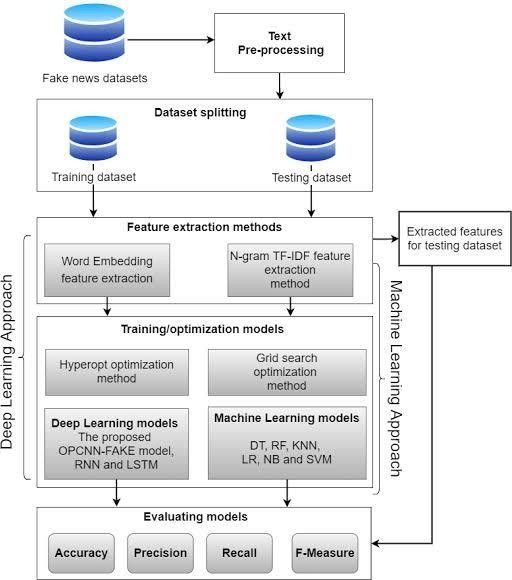
# MODULE DESCRIPTION:

* **Data Collection Module:** This module is responsible for continuously gathering real-time data from various social media platforms. It captures various types of data including text (tweets, posts), images, and metadata such as user behavior (likes, shares, retweets, comments).
* **Feature Engineering Module:** This module is responsible for extracting relevant features that improve rumor detection. Key features include textual content (e.g., length, keywords), user credibility (e.g., account age, follower count, posting frequency), and propagation features (e.g., how widely and quickly a post spreads).
* **Rumor Detection Engine:** This is the core module of the system, where the actual classification of content into rumor and non-rumor takes place. A hybrid model combining Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and an attention mechanism ensures that the system captures both the temporal sequence of rumors as well as spatial patterns.
* **Real-Time Processing Module:** Handles the continuous stream of incoming social media data and processes it in real-time, ensuring that rumors are detected promptly. This module integrates streaming technologies such as **Apache Kafka** and **Spark Streaming** for processing large-scale, fast-moving data.
* **Visualization and Reporting Module:** Provides interactive dashboards and reports for visualizing detection results, accuracy, false-positive and false-negative rates, and rumour propagation patterns. It also offers explainability by presenting insights on why a post is classified as a rumor.

# CHAPTER 4 DESIGN

* 1. **BLOCK DIAGRAM**

The block diagram is typically used for a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of Implementation. Main operations are to add, view, update and delete the details of the faculty and students The following figure represents the block diagram.



# Fig 4.1: Block Diagram

* 1. **DATA FLOW DIAGRAM**

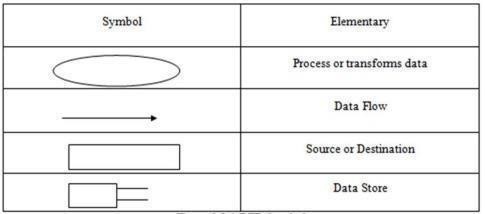
A Data Flow Diagram (DFD) visually represents the flow of data through a system. In the context of rumor detection on online social networks, a DFD can illustrate the key processes involved in collecting, analyzing, and classifying posts as rumors or not **.**

In the DFD, there are four symbols:

* Square defines a source (originator) or destination of system data.
* An arrow identifies data flow. It is the pipeline through which the information flows. Data move in a specific direction from an origin to a destination.
* A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
* An Open Rectangle is a data store, data at rest or a temporary repository of data.

# Constructing a DFD:

Several rules of thumb are used in drawing DFD’S:

* Process should be named and numbered for an easy interface. Each name should be representative of the process.
* The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source.
* An alternative way is to repeat the source symbol as the destination.
* The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized.

# Fig 4.2.1: Data Flow Diagram

**DFD Symbols and Their Meanings:**

# External Entities:

Social Media Platform: The source of posts.

Fact-Checker: The destination of verified information.

# Processes:

Data Collection: Collects posts from the social media platform. Preprocessing: Cleans and normalizes the collected data.

Feature Extraction: Extracts relevant features from the posts (e.g., sentiment, keywords, network

structure).

Model Training: Trains a machine learning or deep learning model using labelled data. Rumor Classification: Classifies new posts as rumors or not based on the trained model. Fact-Checking: Sends verified information to the fact-checker.

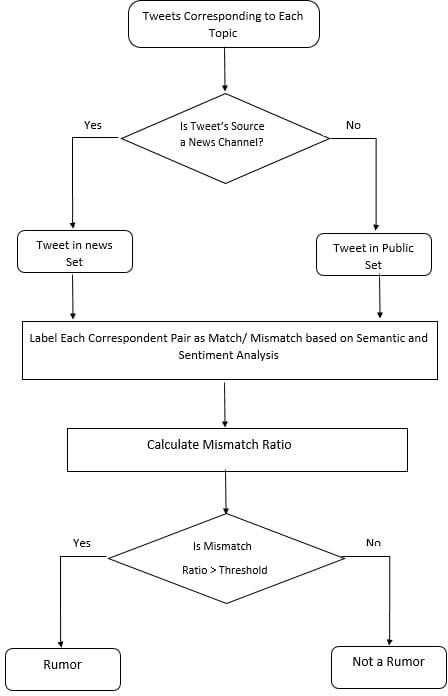
# Data Stores:

Raw Data: Stores the collected posts.

Processed Data: Stores the pre-processed and feature-extracted data. Model: Stores the trained machine learning or deep learning model. **Data Flows:**

Posts from the social media platform to the data collection process.

Pre-processed data from the preprocessing process to the feature extraction process.



# Fig 4.2.2: Data Flow Diagram

**Context level DFD:**

Context-Level Data Flow Diagram (DFD) is a high-level, simplified representation of a system that highlights its primary function and how it interacts with external entities. Often referred to as a **Level 0 DFD**, it presents the system as a single process (represented by a circle or bubble) with external entities, such as users, systems, or organizations, shown around it. These external entities interact with the system by either providing input or receiving output. The context- level DFD doesn't delve into the internal workings of the system but instead focuses on defining the boundaries of the system, showcasing data flow between the system and its external environment, thus providing an overview of the system's interactions.

# Top level DFD:

Top-level Data Flow Diagram (DFD), also known as a Context Diagram, provides a high-level overview of a system by depicting the system as a single process, representing its boundaries and external interactions. In this diagram, external entities (such as users, systems, or organizations) are shown interacting with the system via data flows, which represent the input and output of information. The top-level DFD helps stakeholders understand the overall system's scope and context without diving into internal processes or details. It serves as a starting point for more detailed levels of DFDs, breaking down the system's internal components in subsequent levels.

# Detailed Level Diagram:

Detailed level diagram, often referred to as a low-level design (LLD), breaks down a system into its finer components, showing how individual elements work together to achieve the system's goals. It provides an in-depth view of modules, subsystems, interfaces, data flows, and interdependencies between components. Unlike high-level diagrams, which give a broad overview, a detailed level diagram outlines specifics like data structures, algorithms, database schemas, and communication protocols. It includes details such as class diagrams, sequence diagrams, and state machines in software engineering or wiring and circuit layouts in hardware systems.

# INTRODUCTION

**CHAPTER 5 IMPLEMENTATION**

First, a dataset of social media posts is collected and labelled as either rumors or credible information. Next, relevant features like textual content, user information, and social context are extracted. Then, machine learning algorithms like Logistic Regression, Decision Tree Classifier, Random Forest Classifier, and LSTM are trained on the prepared dataset to learn patterns and relationships between features and rumours labels. The trained models are evaluated using metrics like accuracy, precision, recall, and F1-score to select the best- performing one. Finally, the chosen model is integrated into a real-time system that can process new social media posts and classify them as rumours or credible information, helping to mitigate the spread of misinformation on platforms like Twitter or Facebook.

# 5.1 TECHNOLOGIES USED What is Python?

Python is a high-level, interpreted programming language known for its simplicity

and readability. Created by “Guido van Rossum” and first released in 1991, Python emphasizes code readability, allowing developers to express ideas in fewer lines of code compared to other languages like C++ or Java. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Python is widely used for web development, data analysis, artificial intelligence, machine learning, automation, scientific computing, and more. Its extensive standard library and vast ecosystem of third-party packages make it versatile for various tasks. Popular frameworks and libraries include Django (for web development), Pandas (for data analysis), TensorFlow (for machine learning), and Flask (for lightweight web applications).

Its syntax is clean and straightforward, making Python an excellent language for beginners while still being powerful for experts. It is cross-platform, running on Windows, macOS, and Linux, and its flexibility has made it a top choice for developers worldwide.

# Libraries of Python:

Here is a list of popular Python libraries commonly used for machine learning (ML) and deep learning (DL) tasks:

1. Core Python Libraries for ML and DL:

These libraries form the foundation for machine learning and deep learning tasks:

* + NumPy: Used for numerical computations, handling arrays and matrices.
  + Pandas: Data manipulation and analysis, used for handling structured data.
  + Matplotlib: Visualization library for plotting graphs, histograms, etc.
  + Seaborn: Built on Matplotlib, it provides more aesthetically pleasing statistical graphics.
  + SciPy: Scientific computing, used for mathematical algorithms and functions.
  + Scikit-learn: Most popular ML library for classical machine learning algorithms like linear regression, decision trees, clustering, etc.
  + Stats models: Statistical modelling and hypothesis testing.

These libraries are mainly used for implementing machine learning models:

* + Scikit-learn: As mentioned, it’s a go-to library for traditional machine learning algorithms (classification, regression, clustering).
  + XGBoost: Optimized gradient boosting framework for large-scale datasets.[7]
  + LightGBM: A gradient boosting framework that is more efficient for large datasets, especially with categorical features.
  + CatBoost: Gradient boosting framework with support for categorical data.
* PyCaret: Low-code ML library that automates machine learning pipelines and is easy to use.

17

* + TPOT: Automated machine learning library that uses genetic algorithms to optimize pipelines.

# Deep Learning Libraries:

These libraries provide tools for building deep neural networks:

* + TensorFlow: A comprehensive open-source deep learning framework developed by Google. Supports building and training deep neural networks.
  + Keras: A high-level neural networks API, running on top of TensorFlow, which simplifies the model building process**.**
  + PyTorch: Developed by Facebook, it's one of the most popular libraries for deep learning research, offering dynamic computational graphs.
  + Theano: Another deep learning library that allows for fast computations but is now less used in Favor of TensorFlow and PyTorch**.**
  + MXNet: Deep learning framework that supports dynamic and symbolic programming for neural networks, often used for large-scale training.
  + Chainer: Flexible deep learning library that supports dynamic neural networks (though it's more niche compared to PyTorch and TensorFlow).
  + Caffe: Primarily used for deep learning in computer vision tasks.
  + DeepPy: Library for building deep neural networks in a flexible manner.
  + CNTK (Microsoft Cognitive Toolkit): Deep learning library developed by Microsoft for training deep neural networks.
  + DL4J (Deeplearning4j): Deep learning library in Java with Python API support, mainly used for integrating with enterprise systems

# Modules in Python:

**Modules**

In Python, the term "modules" refers to individual files containing Python code (functions, classes, variables) that can be imported and used in other Python scripts. Many Python libraries for machine learning and deep learning are also made up of multiple modules, each with a specific purpose.

Here’s a breakdown of the core modules (submodules) that are commonly used in Python, machine learning, and deep learning libraries:

# Core Python Modules (General Purpose)

These modules are part of the Python Standard Library and are useful for everyday programming tasks:

* + OS: Interacts with the operating system (file system operations, environment variables).
  + sys: Provides access to system-specific parameters and functions.
  + math: Provides mathematical functions (trigonometry, logarithms, etc.).
  + random: Generates random numbers, shuffles lists, etc.
  + datetime: Handles date and time manipulation.
  + collections: Provides specialized data structures like deque, Counter, defaultdict, etc.
  + itertools: Efficient looping constructs like permutations, combinations, etc.
  + function tools: Higher-order functions for working with other functions (e.g., reduce, lru\_cache).
  + Json: JSON parsing and serialization.

# Modules in Deep Learning Libraries TensorFlow:

* + tensorflow.keras: High-level API for building neural networks (used to define models, layers, activations). o tensorflow.keras.models: For defining, training, and evaluating models.
  + o tensorflow.keras.layers: Predefined layers like Dense, Convolutional, LSTM, etc.
    - tensorflow.keras.optimizers: Optimizers like SGD, Adam, RMSProp.

o tensorflow.keras.losses: Common loss functions like mean\_squared\_error, binary\_crossentropy.

* + - tensorflow.keras.metrics: Metrics like accuracy, AUC, Precision.
  + tensorflow.data: Tools for building efficient input pipelines for large datasets.
  + tensorflow.lite: Module for deploying models to mobile and IoT devices.
  + tensorflow.distribute: Tools for distributing training across multiple GPUs or machines.

# PyTorch

* + torch.nn: Core module for building neural networks (e.g., Linear, Conv2d, LSTM, etc.).
  + torch.optim: Optimizers like Adam, SGD, RMSProp.
  + torch.autograd: Automatic differentiation and backpropagation.
  + torch. utils.data: Utilities for loading datasets (e.g., Data Loader, Dataset).
  + torchvision: Tools for computer vision, including image datasets, transforms, and pretrained models (e.g., torchvision.models, torchvision.transforms).
  + torch.cuda: Provides CUDA support for running computations on GPUs.

# Keras (often integrated with TensorFlow)

* + keras.models: Functions to build models (Sequential, Model API).
  + keras.layers: Layers like Dense, Conv2D, LSTM, Dropout, etc.
  + keras.callbacks: Functions like EarlyStopping, ModelCheckpoint.
  + keras.preprocessing: Preprocessing functions for image, text, and sequence data.
  + keras.optimizers: Optimizers like Adam, SGD, etc.

# MXNet

* + mxnet.ndarray: NDArray operations for manipulating matrices.
  + mxnet.gluon: High-level API for defining and training models.
  + mxnet.autograd: For automatic differentiation in MXNet.
  + mxnet.lr\_scheduler: Learning rate scheduling strategies.

# FastAI

* + fastai.vision: Contains functions and models specific to computer vision tasks.
  + fastai.tabular: Tools for working with tabular data (e.g., TabularDataBunch, TabularLearner).
  + fastai.text: Modules for NLP tasks (e.g., TextDataBunch, language\_model\_learner).
  + fastai.callbacks: Useful callbacks such as SaveModelCallback, LearningRateScheduler.

# CHAPTER 6 TESTING

In any machine learning and deep learning project, testing is a critical phase that ensures that

the models and the overall system function correctly, accurately, and efficiently. Testing also helps in evaluating how well the models generalize to unseen data and identify problems that might affect the performance.

For your rumor detection project, various tests were performed during different phases, including model evaluation, error identification, and system robustness checks. Below is a detailed breakdown of the types of tests, problems encountered, and the countermeasures adopted:

Model Evaluation

Deep Learning Model Testing (LSTM) Performance Testing

Model Evaluation Metrics **6.1 Model Evaluation**

1. Train-Test Split
   * Description: A common practice in machine learning is to split the dataset into training and testing sets (e.g., 80% training and 20% testing). This allows the model to be trained on one part of the data and tested on another to evaluate its generalization capabilities.
   * Tests Performed:
     + Logistic Regression, Random Forest, Decision Tree, and LSTM models were trained on the training set. o Accuracy, Precision, Recall, F1 Score, and ROC AUC score were computed on the test set to evaluate the model's performance. o Tools: scikit-learn for calculating metrics and performance measures.

* Problems Encountered:
  + Imbalanced dataset:
  + Rumor detection datasets often have more non-rumor tweets than rumor tweets, leading to a bias where the model tends to classify most tweets as non-rumors.
* Countermeasures:
  + Resampling techniques: Methods like oversampling the minority class (rumors) or under sampling the majority class (non-rumors) were considered to balance the dataset.
  + Evaluation metrics: Instead of relying solely on accuracy (which can be misleading with imbalanced data), metrics like precision, recall, F1-score, and ROC-AUC were prioritized.

1. Cross-Validation
   * Description: Cross-validation was used to ensure that the model’s performance was consistent across different subsets of the data.
   * Tests Performed:
     + k-Fold Cross-Validation (where k=5 or 10) was applied to assess how the model performs across different folds of data.

* Problems Encountered:
  + Overfitting: Some models, like Decision Trees, performed well on the training data but failed to generalize to the test data.
* Countermeasures:
  + Regularization: Techniques like limiting tree depth or pruning were used to prevent overfitting for Decision Trees.

# Deep Learning Model Testing (LSTM)

a. Sequential Data Processing (LSTM)

* Description: LSTM models were used to handle the sequential nature of text data. Given the model's complexity, its training and testing involved more sophisticated testing procedures.
* Tests Performed:

o Model accuracy was checked after each epoch during training, and a validation set was used to monitor model performance. o A confusion matrix was used to evaluate the number of correctly and incorrectly classified tweets.

* Problems Encountered:
  + Vanishing Gradient Problem: During LSTM training, gradients became too small, preventing the network from learning effectively, especially for long sequences of text. o Overfitting: The LSTM model tended to overfit to the training data after a few epochs.
* Countermeasures:
  + Dropout Layers: A dropout layer with a 20% dropout rate was introduced to prevent overfitting by randomly dropping some neurons during training. o

Batch Normalization: Applied to stabilize and accelerate the training process.

* + Early Stopping: Implemented to stop training when the validation loss stops improving, preventing the model from overfitting the training data.
  + Hyperparameter Tuning: Batch size, learning rate, and number of LSTM units were tuned to optimize performance without overfitting

.

# Performance Testing

1. Scalability and Efficiency
   * Description: Given the large volume of data in social media, the system’s ability to scale and efficiently process data was tested.
   * Tests Performed:
     + The pipeline was tested on larger datasets to assess if the system could handle the increased load.

* Problems Encountered:

o Memory Issues: High-dimensional data from the TF-IDF vectorizer caused memory reduce the feature space before feeding the data into the models. o Sparse Matrices: Leveraged sparse matrices for the TF-IDF representation, which helped significantly reduce memory usage.

1. Speed and Computational Efficiency
   * Description: Model training and prediction times were measured to ensure that the models could be deployed in real-time or near real-time environments.
   * Problems Encountered:
     + Slow Training Times: Training deep learning models like LSTM on large datasets took a significant amount of time.

* Countermeasures:
  + Optimization of Model Parameters: Reduced the complexity of the LSTM model by lowering the number of LSTM units and using a smaller max vocabulary size in the tokenizer.
  + Batch Processing: Implemented batch processing for large datasets to reduce memory consumption and speed up processing. Issues during the training of models.

# Model Evaluation Metrics

* + - Accuracy: Measures the percentage of correctly predicted instances. However, due to the imbalanced nature of the dataset, accuracy alone wasn't sufficient.
    - Precision: Measures the percentage of true positive predictions out of all positive predictions made by the model. It was crucial for ensuring that the model did not incorrectly classify non-rumors as rumors.
    - Recall: Measures the percentage of actual positive instances that were correctly identified by the model. Important for ensuring that the model catches as many actual rumors as possible.
    - F1 Score: The harmonic mean of precision and recall, offering a balanced metric to account for both false positives and false negatives.
    - ROC AUC: The ROC AUC score measures the model's ability to distinguish between rumors and non-rumors. A higher AUC indicates a better performing model.

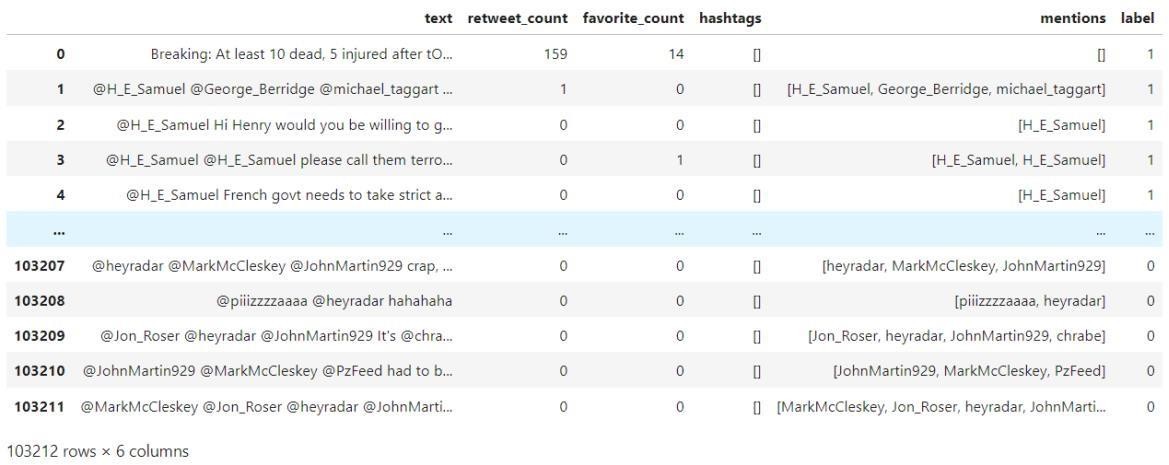
Through rigorous testing, various issues related to model performance, overfitting, class imbalance, and scalability were identified and addressed. By using techniques like cross validation, resampling, dropout layers, and dimensionality reduction, the project ensured robust performance across both machine learning and deep learning models. These tests ultimately improved the reliability of the rumor detection system.

# CHAPTER 7 RESULTS

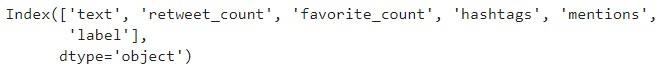
**Sample Data from PHEME Rumor Dataset:**



# Data Frame:



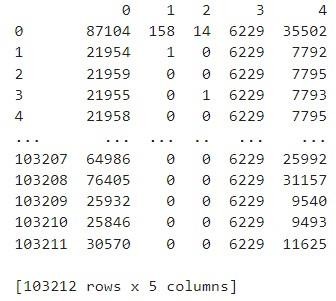
**Columns:**



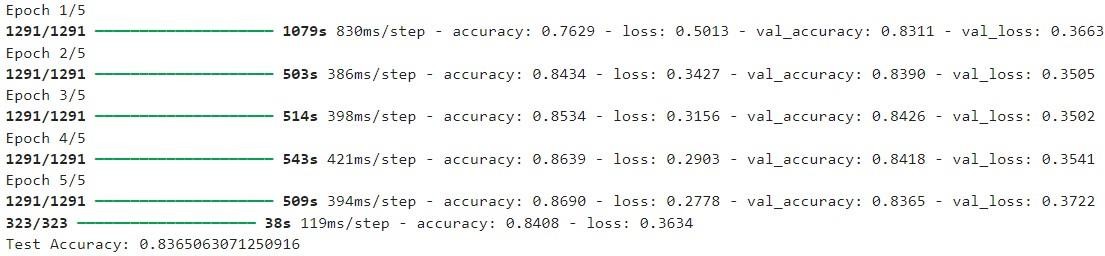
# Dataset Dimensions: Feature Set and Labels:



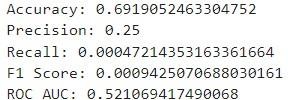
**Initial Numerical Data:**



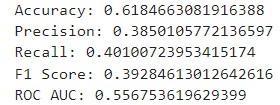
# LSTM Model Training and Test Results (Accuracy: 83.65%):



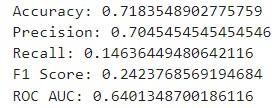
**Logistic Regression Model Evaluation (Accuracy: 69%, Poor Recall):**



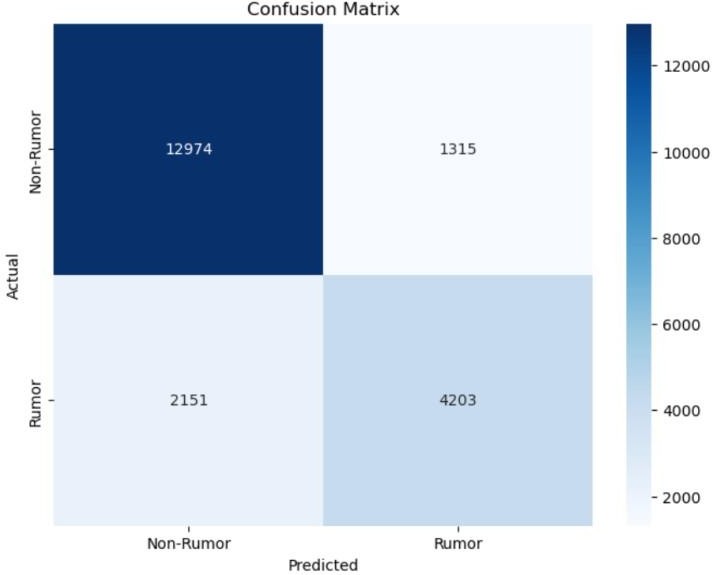
# Decision Tree Model Evaluation (Moderate Performance, F1 Score: 0.39):



**Random Forest Model Evaluation (Accuracy: 71.84%, Low Recall: 14.64%):**

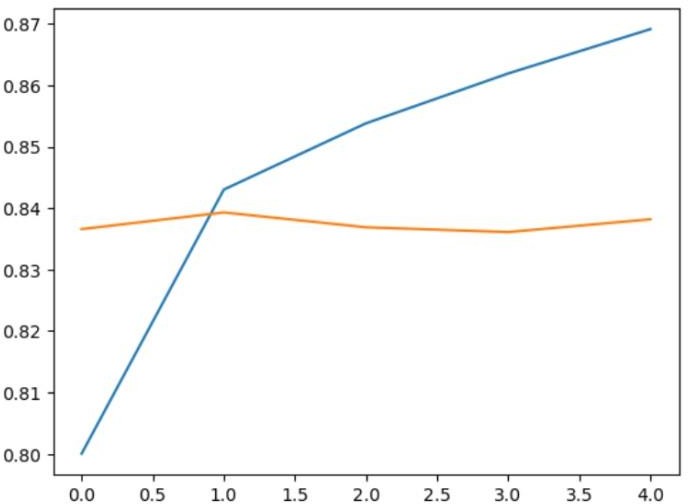


The confusion matrix shows the model's performance in detecting rumors by comparing actual and predicted labels. It helps calculate accuracy, precision, and recall, offering insights into the model's effectiveness in distinguishing rumors from non-rumors shown in below

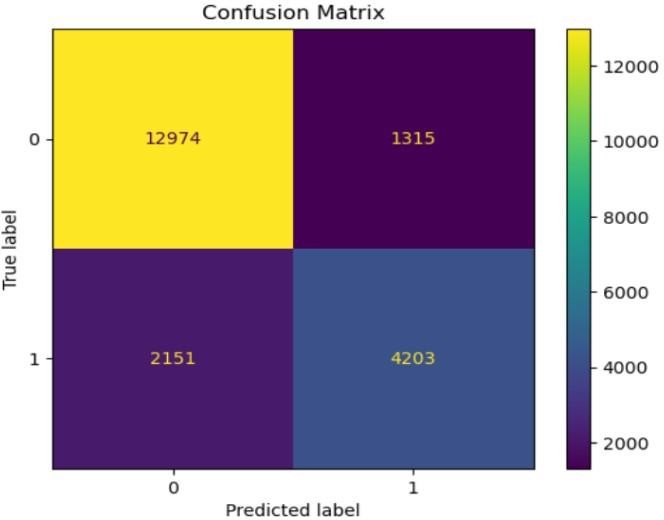


# Fig 7.1: Confusion Matrix

This plot shows the performance of a deep learning model for rumor detection on social networks, comparing training (blue) and validation (orange) accuracy. It highlights learning progress and potential overfitting shown in below**.**



# Fig 7.2: Line Plot

This confusion matrix shows the performance of a deep learning model for rumor detection on social networks. It displays true positives, true negatives, false positives, and false negatives. Including it helps evaluate the model's accuracy, precision, and recall in classifying rumors and non-rumors shown in below.

# Fig 7.3: Confusion Matrix

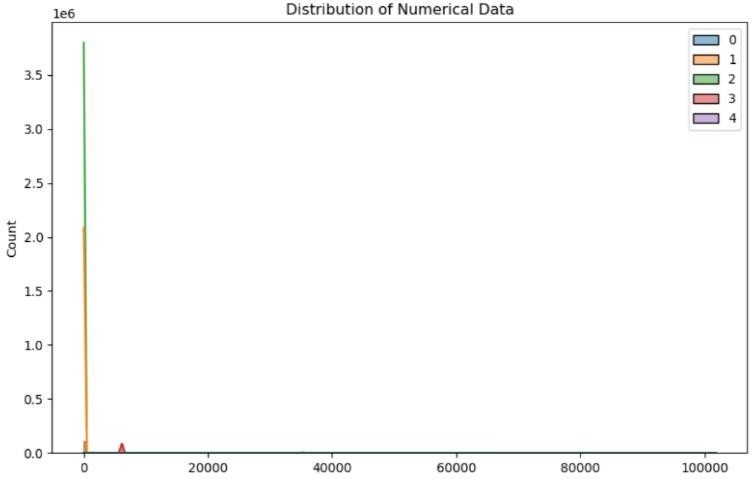
The ROC curve in this deep learning-based rumor detection project evaluates the model's ability to classify rumors and non-rumors by plotting the true positive rate against the false positive rate. The diagonal line and AUC of 0.50 suggest the model performs no better than random guessing. Improving the model architecture or fine-tuning is necessary to enhance classification accuracy shown in below.

# 

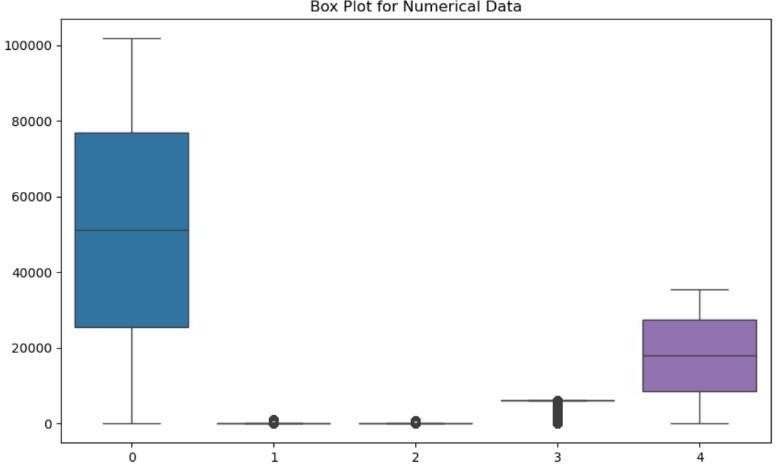
# 

# Fig7.4: ROC Curve

The distribution plot shows the frequency of numerical data in the rumor detection dataset, with most values concentrated near zero. This indicates a class imbalance, where non-rumors dominate, potentially affecting model performance. Addressing this issue with resampling or weighted loss functions is essential for better accuracy shown in below.

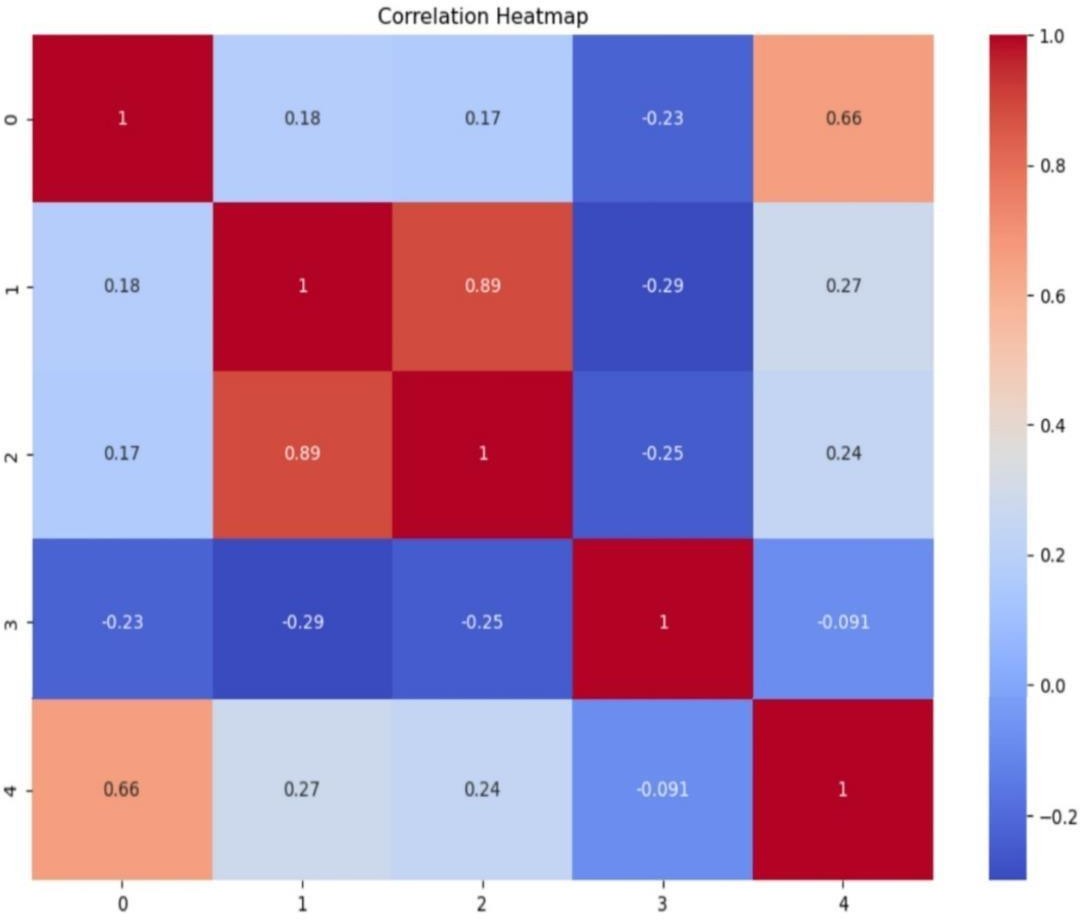


# Fig 7.5: Distribution Plot

Box plots are a valuable tool for visualizing data distribution in rumor detection. They can help identify differences between groups, outliers, and informative features shown in below

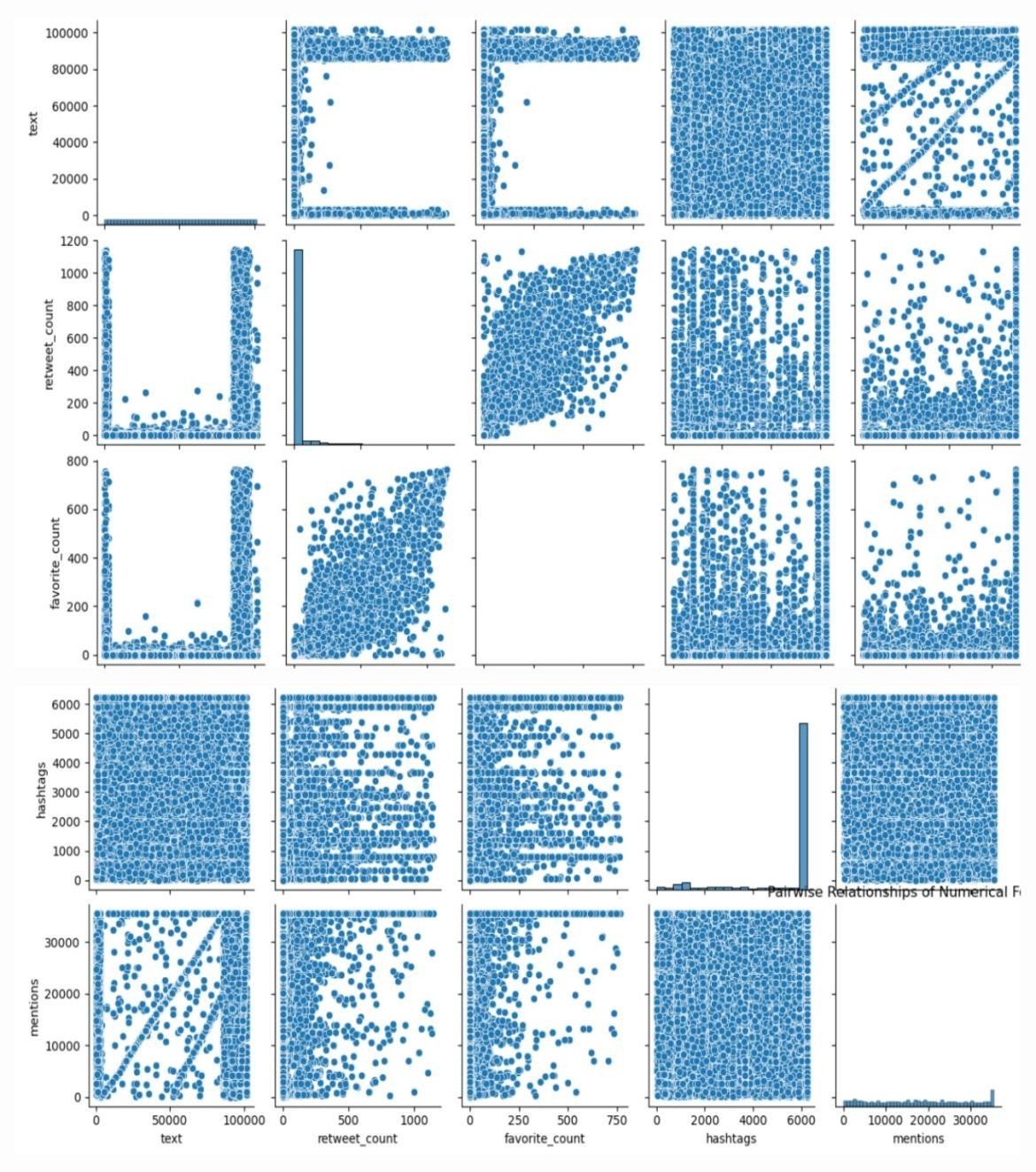
# Fig 7.6: Box Plot

The correlation heatmap visualizes relationships between features, highlighting which ones are strongly correlated. This helps identify redundant features, reducing dimensionality and improving model efficiency. It also reveals key variables that could impact the accuracy of rumor classification shown in below**.**



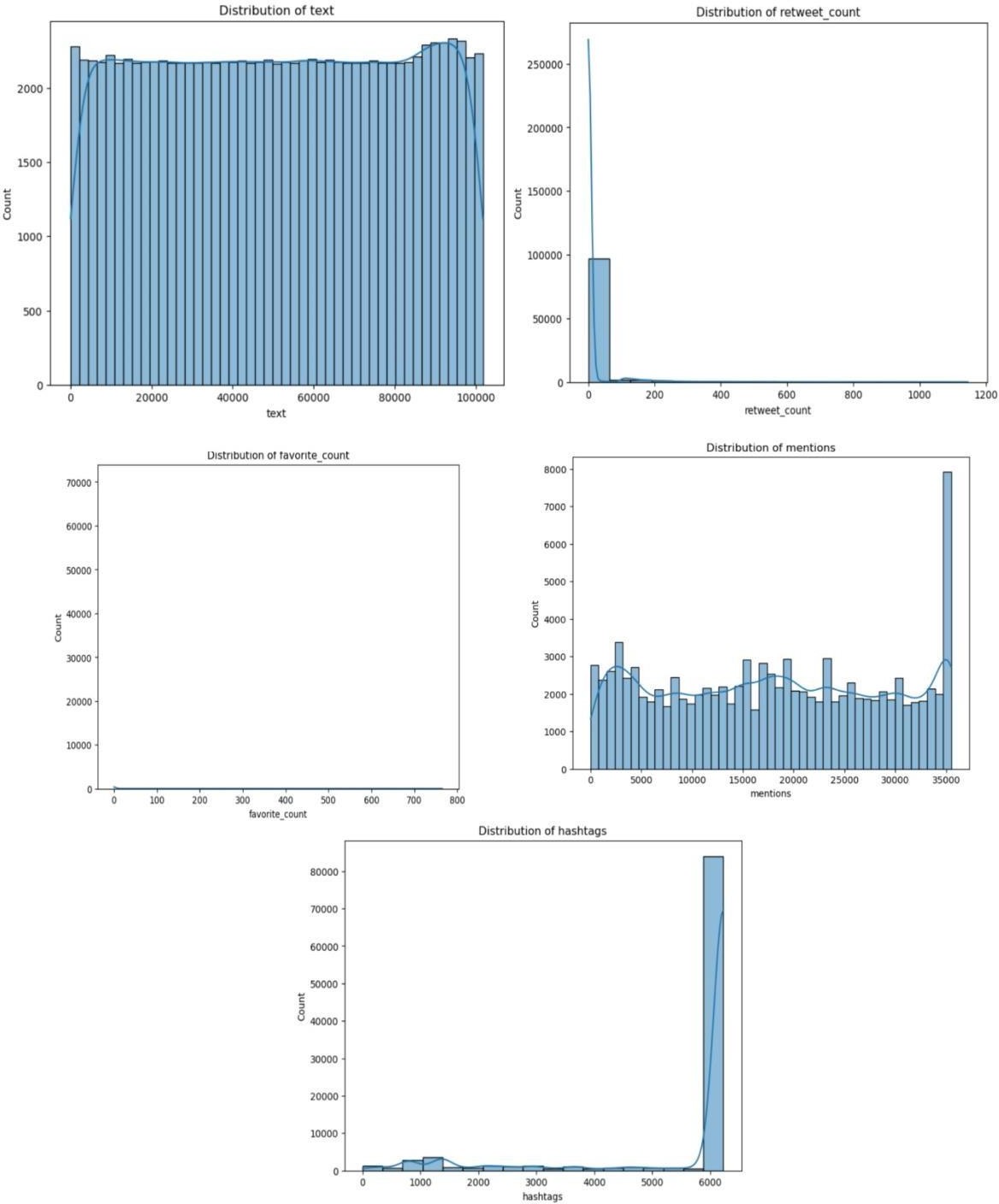
# Fig 7.7: Correlation Heatmap

pair plot that visualizes the relationships between different text features like length, hashtags, retweets, mentions, and favorites. Understanding these relationships can help identify potential correlations and patterns that are relevant for rumor detection shown in below.



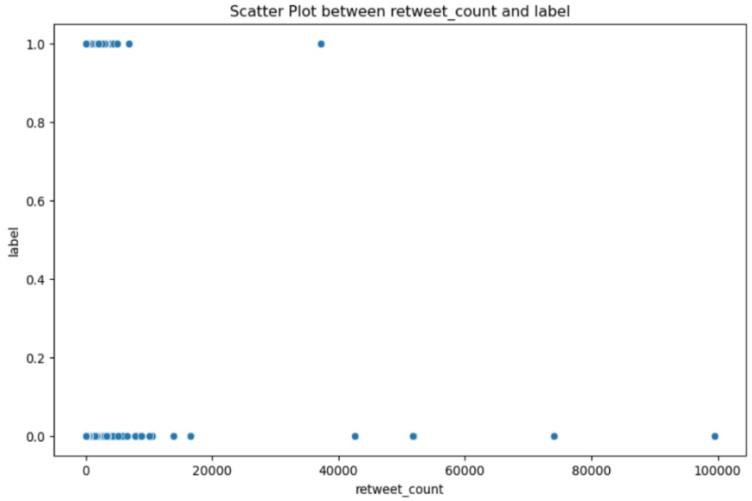
# Fig 7.8: Pair Plot

The images are histograms that show the distribution of text features like length, hashtags, retweets, mentions, and favorites. Understanding these distributions is important for rumor detection as they can help identify outliers, select informative features, preprocess data, and evaluate model performance shown in below.



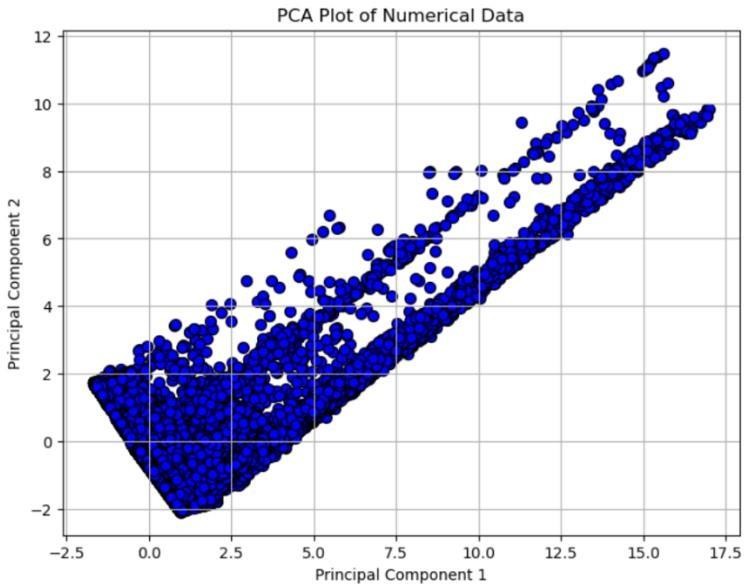
# Fig 7.9: Histogram Plot

35

Scatter plots visually represent the relationship between numerical features and the target variable in rumor detection, helping identify correlations, outliers, and non-linear patterns. They are crucial for feature selection, outlier detection, and improving model accuracy shown in below.

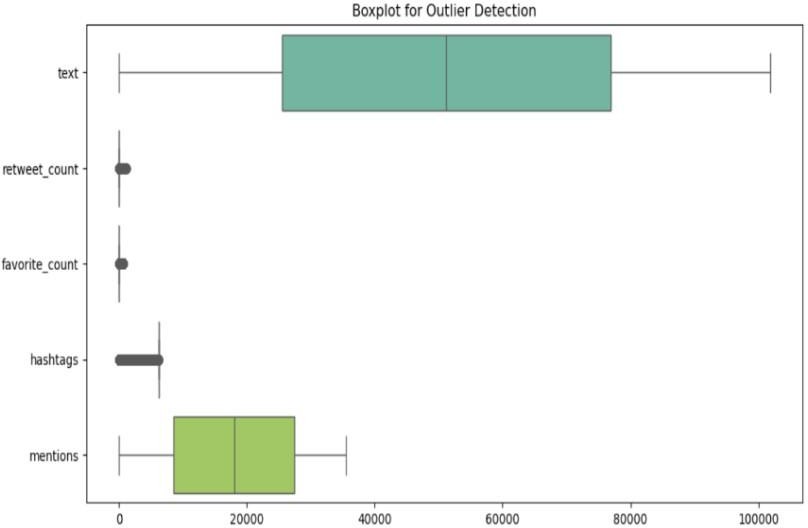
# Fig 7.10: Scatter Plot

PCA plots reduce high-dimensional data into 2D or 3D, revealing patterns, clusters, and outliers in rumor detection datasets. They help assess feature importance, simplify data analysis, and enhance model performance by focusing on key variance shown in below.



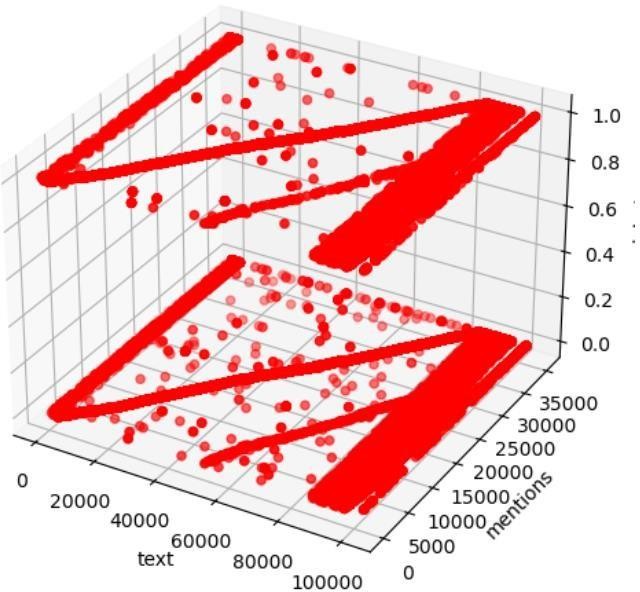
# Fig 7.11: PCA Plot

Box plots help detect outliers and assess data distribution in rumor detection by visualizing the spread, quartiles, and outliers of features. They aid in identifying unusual posts, comparing distributions between rumor and non-rumor groups, and selecting important features shown in below.



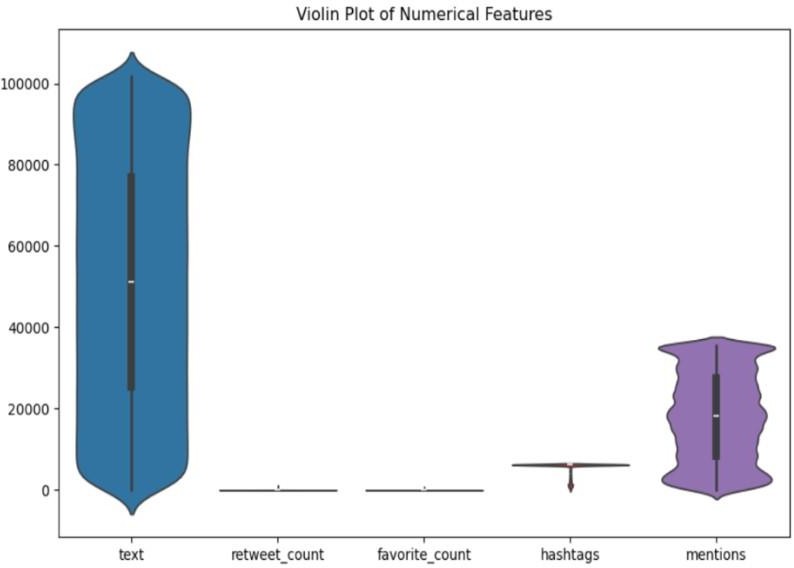
# Fig 7.12: Box Plot

3D scatter plots are a valuable tool for visualizing relationships between three numerical features in rumor detection. They can help identify patterns, outliers, and non-linear relationships shown in below.



# Fig 7.13: 3D Scatter Plot

Violin plots are a valuable tool for visualizing data distributions in rumor detection. They can reveal multimodality, skewness, and outliers shown in below.



# Fig 7.14: Violin Plot

# CHAPTER 8

**FUTURE SCOPE AND ENHANCEMENTS**

Building upon the success of LSTM and Random Forest in this project, several future directions can be explored to further enhance rumor detection on social networks:

* + - Hybrid Models: Combining the strengths of LSTM and Random Forest could lead to even more robust performance. For example, LSTM could be used to extract deep features from text data, which could then be fed into a Random Forest classifier for final prediction.
    - Ensemble Methods: Experimenting with different ensemble techniques, such as stacking or bagging, could improve generalization and reduce overfitting.
    - Multimodal Approaches: Incorporating additional modalities, such as images or videos, could provide valuable contextual information and improve detection accuracy.
    - Real-time Detection: Developing real-time rumor detection systems would enable timely intervention and mitigation of the spread of misinformation. This would require efficient algorithms and scalable architectures.
    - Contextual Understanding: Enhancing the models' ability to understand the context of rumors, including the source, topic, and social context, could improve their accuracy and reliability.
    - Explainability: Developing techniques to explain the predictions made by the models would increase transparency and trust in the system.
    - Cross-lingual Rumor Detection: Extending the models to handle multiple languages would enable rumor detection in a global context.
    - User Interaction: Incorporating user feedback and interaction could help refine the models and improve their performance over time.

# CHAPTER 9

# CONCLUSION

In this project, we conducted a comprehensive evaluation of various machine learning and deep learning models for rumor detection on online social networks. Our experiments demonstrated that the deep learning approach using Long Short-Term Memory (LSTM) networks significantly outperformed traditional machine learning models like Logistic Regression, Random Forest, and Decision Tree.

LSTM's ability to effectively capture the sequential nature of text data proved to be a critical factor in its superior performance. Unlike traditional models that treat text as a bag of words, LSTM networks can understand the context and relationships between words within a sequence. This enables them to better identify subtle patterns and nuances in rumor-related text, leading to more accurate and reliable predictions.

While Random Forest also exhibited strong performance, LSTM's advantages were particularly evident in terms of recall and F1 score. LSTM demonstrated a higher ability to correctly identify true rumors while minimizing false negatives, making it a more suitable choice for applications where detecting all potential rumors is crucial.

Our findings highlight the potential of deep learning methods for rumor detection in online social networks. As these networks continue to grow in size and complexity, deep learning models like LSTM are likely to play an increasingly important role in addressing the challenges posed by the rapid spread of misinformation. Future research could explore the integration of other deep learning techniques, such as attention mechanisms and generative models, to further enhance the accuracy and efficiency of rumor detection systems.

# REFERENCES

1. Buntain, C., & Golbeck, J. (2017). Automatically identifying fake news in popular Twitter threads. Proceedings of the IEEE International Conference on Smart Cloud (pp. 208-215).
2. Kumar, S., & Shah, N. (2018). False information on web and social media: A survey. Social Media Analytics: Advances and Applications, 15(4), 35-55.
3. Zubiaga, A., Aker, A., Bontcheva, K., Liakata, M., & Procter, R. (2018). Detection and resolution of rumors in social media: A survey. ACM Computing Surveys, 51(2), 1-36.
4. Ma, J., Gao, W., Wei, Z., Lu, Y., & Wong, K. F. (2016). Detect rumors using time series of social context information on microblogging websites. Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (pp. 17511754).
5. Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. ACM SIGKDD Explorations Newsletter, 19(1), 22- 36.
6. Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. Science, 359(6380), 1146-1151.
7. Chen, T. Guestrin, C. (2016). XG Boost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794).
8. Hochreiter.S & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780.

41