ML BASED MISINFORMATION DETECTION & ANALYSIS USING SOCIAL NETWORK ANALYSIS

Enroll. No. - 23103067, 23103078,23103096

Name of Students- Ritika Singh, Shireen Kachroo, Paakhi Somani

Name of Supervisor-Dr. Ankita Verma

November 2025

Submitted in partial fulfillment of the Degree of Bachelors of Technology

In

Computer Science Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY

JAYPEE INSTITUTION OF INFORMATION TECHNOLOGY, NOIDA

TABLE OF CONTENTS

DECLARATION BY STUDENTS

We **Ritika Singh, Shireen Kachroo and Paakhi Somani** declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

1. List of Pre Built Libraries: NetworkX, pandas, matplotlib, scikit-learn, transformers/torch
2. List of Pre Built Features: TF-IDF text vectorization, NetworkX graph generation, Centrality Calculation, ML classifiers
3. Percentage of pre written source code and code written by us: 20%, 80%

Place: Noida, India

Date: Nov 22,2025

Name	Enrolment No.	Signature
Ritika Singh	23103067	
Shireen Kachroo	23103078	
Paakhi Somani	23103096	

DECLARATION BY SUPERVISOR

I, **Dr. Ankita Verma,** declare that the above-submitted project titled "**ML Based Misinformation Detection & Analysis Using Social Network Analysis**" was conducted under my supervision. The project is original, not copied from any external source, and has not been submitted earlier in JIIT.

Remarks (if any):

Signature (Supervisor)

CERTIFICATE

This is to certify that the work titled '**ML Based Misinformation Detection and Spread Analysis using Social Network Analysis'** submitted by **Ritika Singh, Shireen Kachroo** and **Paakhi Somani** in partial fulfilment for the award of degree of **B.Tech** in **Computer Science** of Jaypee Institute of Information Technology, Noida has been carried out in my supervision. This work has not been submitted partially or wholly to another University or Institute for the award of this or other degree or diploma.

Signature of Supervisor:

Name of Supervisor: Dr. Ankita Verma

Designation: Assistant Professor(Senior Grade)

Date: Nov 22,2025

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who have contributed to the successful completion of this project. First and foremost, we extend our heartfelt thanks to Dr. Ankita Verma, whose guidance, support and encouragement played a pivotal role in the execution of this project. We are deeply appreciative of the opportunity she provided us to work on this meaningful endeavor, which has significantly contributed to our personal and professional growth.

We are particularly grateful for Dr. Ankita's constant assistance throughout the entire process. From the initial stages of conceptualization to the final implementation, her insightful suggestions and hands-on support helped us overcome challenges and stay on track. Her expertise, particularly in working with social network analysis, was invaluable. We learned not only theoretical aspects of the project but also gained practical experience in coding, debugging, training ML models, understanding social network analysis and utilizing key libraries like networkX. The knowledge we have gained from this project will undoubtedly be crucial for our future endeavours in Machine Learning and Social Network Analysis.

Additionally, we would like to thank our peers, mentors, and anyone else who has contributed indirectly to this project, whether through brainstorming ideas, providing feedback, or helping us troubleshoot complex problems. Their support made a significant difference in ensuring the smooth progress of the project. We are grateful to have had the chance to work on such a rewarding project that has enhanced our technical skills and broadened our understanding of the field.

Date: Nov 22,2025

Name	Enrolment Number	Signature
Ritika Singh	23103067	
Shireen Kachroo	23103078	
Paakhi Somani	23103096	

SUMMARY

This project develops a machine learning based misinformation detection system combined with Social Network Analysis (SNA), designed to study how misinformation propagates across online social platforms. The system performs two core functions: (1) classifying information as real or misinformation using NLP and ML, and (2) simulating the spread of misinformation using an SIR epidemic model to analyze network-level diffusion patterns.

The detection pipeline uses comprehensive NLP preprocessing, including HTML cleaning with BeautifulSoup, regular expressions, stopword filtering, sentiment extraction with TextBlob, and feature engineering through TF-IDF and BERT embeddings. Traditional machine learning models(Logistic Regression, Linear SVM, Passive Aggressive Classifier, and Random Forest) are trained alongside a deep learning BERT architecture implemented with PyTorch and AdamW optimization. This ensures robust classification performance even in the presence of noisy or rapidly evolving misinformation.

To analyze spread behavior, the project integrates Social Network Analysis with epidemic modeling. The misinformation diffusion process is simulated using the SIR (Susceptible–Infected–Recovered) model, where each user/node transitions from being vulnerable to misinformation, to sharing it, and eventually recovering. Network properties such as hubs (high-degree nodes), bridges (critical connectors), community clusters, and information bottlenecks are examined to understand how structural positions within a network accelerate or slow down the spread.

By combining ML-based detection with SIR-based diffusion simulation, the project functions as an analytical tool that visualizes how misinformation originates, which nodes act as super-spreaders, how quickly it travels, and what structural interventions can reduce its impact. Despite challenges like dynamic user behavior and unpredictable online interactions, the system demonstrates the potential of integrating NLP, ML, and network science to study and mitigate misinformation in a realistic and data-driven manner.

Chapter 1: Introduction

1. General Introduction

Misinformation has become one of the most critical challenges in the digital era, with false content spreading rapidly across social media platforms. As online networks grow in size and complexity, understanding how misinformation originates, propagates, and influences users is essential for building safer digital ecosystems.

This project focuses on developing a dual-component system: a machine learning–based misinformation detection model, and a social network simulation tool that studies how misinformation spreads using the SIR (Susceptible–Infected–Recovered) model.

Using NLP techniques and transformer-based models like BERT, the system classifies online articles as real or misinformation. In parallel, the SIR-based simulation demonstrates how misinformation infects users, how it travels across hubs and bridges in a social graph, and how recovery or mitigation affects its diffusion. This integration highlights the combined power of machine learning and network science in combating misinformation.

2. Problem Statement

Although misinformation detection has improved with ML and NLP, the spread of misinformation remains poorly understood due to the dynamic nature of social networks. Key issues include:

1. **High Spread Rate:** False news spreads faster than real news due to emotional appeal and rapid sharing behavior.

2. **Lack of Awareness:** Most users cannot distinguish between credible and misleading information, increasing susceptibility.

3. **Complex Network Behavior:** Online communities contain hubs (highly connected users) and bridges (connecting communities) that amplify misinformation.

4. **Inadequate Tools:** Existing solutions detect fake news but do not simulate how it spreads or identify which users accelerate its propagation.

This project addresses these limitations by combining ML-based detection and network-based spread modeling to provide a complete analytical framework.

3. <u>Significance / Novelty of the Problem</u>

This project introduces a unique, unified approach by merging misinformation detection with spread simulation:

Understanding Spread Dynamics

Using the SIR epidemic model, the system demonstrates how misinformation behaves like a digital virus: infecting users, traveling across communities, and eventually fading out.

Network Awareness

By analyzing hubs, bridges, and community clusters, the tool identifies super-spreaders and vulnerable regions within the network.

Practical Real-World Application

Such models can help governments, platforms, and researchers plan interventions, deploy fact-checking, or reduce virality.

Innovative Learning Approach

The integration of machine learning classification with network diffusion modeling offers students and researchers insights into both NLP and SNA domains.

Scalability and Future Potential

The system can be extended to real-time misinformation monitoring, bot detection, or platform-level alert systems.

4. Empirical Study

A detailed study was performed to evaluate both the misinformation classification model and the spread simulation:

1. **Dataset Exploration**

- Collected misinformation dataset via Kaggle and online repository.
- Labeled samples as real or fake.
- Performed text cleaning with BeautifulSoup, regex, and stopword removal.

2. **Model Comparison**

- **Traditional ML Models**: Logistic Regression, Linear SVM, Passive Aggressive Classifier, Random Forest.
- **Deep Learning Model:** BERT-based transformer using PyTorch, achieving contextual understanding.
- **Feature Techniques:** TF-IDF and BERT embeddings compared for effectiveness.

3. **Performance Metrics**

- Accuracy, precision, recall, and F1-score were used.
- Confusion matrix visualization for misclassification analysis.
- BERT embeddings provided strong, stable performance.

4. **SNA & SIR Simulation Tests**

- Random and scale-free networks were generated to imitate real social platforms.
- Nodes were assigned S, I, or R states.
- Simulations tracked infection spread, peak misinformation spread, and recovery phases.
- Results showed how hubs amplify spread while bridges connect isolated communities, allowing misinformation to leak across clusters.

5. <u>Brief Description of Solution Approach</u>

1. **Text Preprocessing**

- Cleaned text using BeautifulSoup and regex.
- Performed tokenization, sentiment extraction, and vectorization (TF-IDF + BERT embeddings).

2. **Machine Learning Classification**

- Implemented multiple classifiers to identify the most efficient model.
- Fine-tuned thresholds, hyperparameters, and feature sizes.
- Used BERT with AdamW optimization for deep contextual learning.

3. **Network Graph Construction**

- Modeled users as nodes and interactions/shares as edges using Barabasi Albert Network.
- Identified hubs (high-degree nodes) and bridges (high betweenness centrality).

4. **SIR Spread Simulation**

- Initialized a fraction of nodes as infected with misinformation.
- Applied infection and recovery probabilities.
- Simulated how misinformation flows through the network over time.
- Tracked number of susceptible, infected, and recovered nodes at each time step.

6. <u>Comparison with Existing Approaches</u>

1. **Traditional Fake News Classifiers**

- Earlier systems only classify misinformation.
- Our project additionally simulates spread, offering more actionable insight.

2. **Rule-Based or Keyword Models**

- Fast but inaccurate.
- Our BERT-based approach understands context, sarcasm, and language nuance.

3. **Standard SNA Tools**

- Many tools study networks but don’t combine ML detection with SIR modeling.
- Our approach bridges both domains in one framework.

4. **Basic Graph Models**

- Simple models don’t account for hubs and bridges.
- Here, network structure heavily influences spread, making results more realistic.

This combined approach ensures a more powerful and comprehensive solution detecting misinformation and studying how it spreads, bringing together NLP, machine learning, and Social Network Analysis.

Chapter 2: Literature Survey

1. Introduction and Summary

In the era of rapidly growing digital communication, the spread of information, both accurate and misleading, has become a critical global concern. With the widespread use of social media platforms, misinformation can propagate at unprecedented speed, influencing public opinion, decision-making, and societal stability. As a result, researchers have increasingly focused on computational methods for identifying and preventing the spread of fake news. Machine learning and natural language processing have proved to be powerful tools for analyzing linguistic features, detecting deceptive patterns, and classifying news content into Fake or Real.

Extensive studies demonstrate that traditional ML models such as Logistic Regression, SVM, Random Forest, and ensemble techniques can effectively identify fake news when combined with text preprocessing methods like TF-IDF, stemming, and stopword removal. Research also highlights the importance of explainability, feature selection, and dataset quality in enhancing classification accuracy and robustness. Several works emphasize that fake news tends to follow recognizable patterns, including emotional polarity, exaggerated phrasing, and structural inconsistencies, making automated detection feasible.

Beyond classification, another crucial research area is the modeling of misinformation spread within social networks. Prior studies show that fake news spreads differently than real news; typically faster, deeper, and to a wider audience. Graph theory concepts such as degree centrality, closeness centrality, and betweenness centrality have been widely applied to identify influential nodes ("hubs") and bridge-like nodes responsible for connecting separate communities. These concepts form the foundation for understanding how misinformation propagates through interconnected groups.

Additionally, epidemic-inspired models, especially the SIR (Susceptible–Infected–Recovered) model, have been adapted to simulate the spread of misinformation. Literature suggests that SIR-based simulations help visualize infection dynamics, predict outbreak magnitude, and test interventions by altering network structures or reducing node influence.

For this project, an extensive literature review provided essential insights into three key domains:
(1) Fake news classification using machine learning,
(2) Network analysis for identifying influential spreaders, and
(3) SIR modeling to simulate misinformation propagation.

These studies guided our decisions regarding preprocessing pipelines, model selection, graph algorithms, and simulation design. The literature also highlighted potential challenges such as classification ambiguity, biased datasets, noisy network structures, and dynamic spreading behavior. By critically analyzing prior research, we were able to refine our project direction, define realistic goals, and implement a system grounded in validated theoretical principles.

This thorough engagement with existing work strengthened the project's foundation and ensured that the implemented solution aligns with current advancements in fake news detection, social network analytics, and misinformation modeling.

2.Integrated Summary of the Literature Studied

S.No	Article	Description	Metrics Used	Gaps / Shortcomings	Remarks
[1]	Disinformation Detection on Social Media: An Integrated Approach (Rastogi & Bansal, 2022)	Presents an integrated method combining linguistic features, sentiment, and propagation cues to classify misinformation on social	Accuracy, Precision, Recall, F1-Score	Limited network-level diffusion analysis; predominantly text-focused.	Strong foundation for the ML classification part of the project.

		media.			
[2]	IEEE Access: New Techniques for Limiting Misinformation Propagation (2023)	Introduces interventions such as edge-removal and influence-limiting strategies to reduce misinformation spread.	Spread reduction %, cascade size	Focuses on mitigation, not detection; lacks SIR-based modeling.	Useful for designing intervention strategies in the SNA simulation.
[3]	ScienceDirect: Fake News Detection Using Machine Learning (2022)	Provides ML-based approaches for detecting fake news using text classification pipelines.	Accuracy, ROC-AUC	Limited use of network propagation features.	Supports the model training and evaluation methodology
[4]	NetMax (GitHub Repository)	A graph-ranking library useful for scoring node influence in networks.	Influence scores, rank distribution	Not specifically designed for misinformation-related tasks.	Useful for identifying influential spreaders in SNA.

[5]	ArXiv: Social Network Analysis for Misinformation Tracking (2021)	Explores graph-based methods to detect and track misinformation diffusion patterns.	Graph density, modularity, diffusion depth	Fewer real-world datasets; mainly simulated environments.	Good support for the SNA component and propagation tracking.

Chapter 3: Requirement Analysis and Solution Approach

1. Overall Description of Project

The primary goal of this project is to build a complete misinformation analysis tool that combines machine learning based fake news detection with social network driven spread simulation. While many existing systems focus only on classifying content as real or fake, they often fail to explain how misinformation actually spreads across social media networks and which users contribute most to its propagation. This project bridges that gap by providing both detection and diffusion modeling in one integrated framework.

The system processes news articles using NLP techniques to classify them as real or fake. It employs text cleaning, TF-IDF, sentiment extraction, and transformer-based embeddings such as BERT to extract meaningful features. Multiple machine learning models(Logistic Regression, Linear SVM, Random Forest) as well as a deep-learning BERT classifier are trained to achieve high accuracy and stable performance across diverse datasets.

Beyond detection, the project introduces a Social Network Analysis (SNA) module to study how misinformation spreads in a network of users. Using the SIR (Susceptible–Infected–Recovered) model, the system simulates digital infection, showing how misinformation travels through hubs, bridges, and community clusters, and how interventions can reduce its reach. Nodes (users) dynamically change states based on infection and recovery probabilities, allowing realistic observation of viral spread behavior.

The project leverages Python, scikit-learn, PyTorch, and network science libraries to create an accessible and educational framework for misinformation research. Its applications extend beyond academic analysis to areas such as policy design, platform moderation, bot detection, and public awareness studies, where understanding misinformation diffusion is increasingly crucial.

2. Requirement Analysis

Functional Requirements

1. **Data Acquisition**

- Import misinformation datasets from Kaggle or online repositories.
- Load CSV/text data for preprocessing and model training.

2. **Text Preprocessing**

- Clean text using HTML removal, regex, stopword filtering, and sentiment extraction.
- Convert text into numerical form using TF-IDF and BERT embeddings.

3. **Misinformation Classification**

- Train ML models (Logistic Regression, SVM, Passive Aggressive, Random Forest).
- Implement a BERT-based classification pipeline using PyTorch and Transformers.

4. **Network Construction**

- Generate or import a social network graph representing user interactions.
- Identify nodes, edges, hubs, and bridges.

5. **SIR Spread Simulation**

- Define infection and recovery probabilities.
- Simulate how misinformation spreads across the network.
- Track Susceptible, Infected, and Recovered states over time.

6. **Visualization & Reporting**

- Plot confusion matrices, accuracy charts, and SIR curves.
- Display network graphs showing infection flow.

<u>Non-Functional Requirements</u>

1. **Performance**

- Ensure fast preprocessing and reliable model training.
- Maintain efficient spread simulation even on medium-sized networks.

2. **Accuracy & Reliability**

- Achieve high accuracy on misinformation classification using robust ML models.
- Ensure consistency in SIR simulation outcomes with deterministic parameters.

3. **Scalability**

- Support multiple datasets and network sizes.
- Allow addition of new ML models or new diffusion rules.

4. **Usability**

- Provide clear configuration options for model parameters and simulation settings.
- Keep the workflow modular so users can easily modify preprocessing or spread logic.

<u>Hardware Requirements</u>

I. **Computer System**

- A laptop or PC with Python installed.
- Minimum 8GB RAM recommended for BERT-based training.

II. GPU (Optional but beneficial)

- For faster BERT fine-tuning and large-scale experiments.

<u>Software Requirements</u>

I. **Python 3.x**

- Primary programming environment.

II. **NLP & ML Libraries**

- scikit-learn (ML models & evaluation)
- Transformers (BERT tokenizer + model)
- PyTorch (BERT training & dataloaders)
- TextBlob (sentiment extraction)

III. **Network Science Libraries**

- NetworkX (graph creation, hubs, bridges, SIR simulation support)

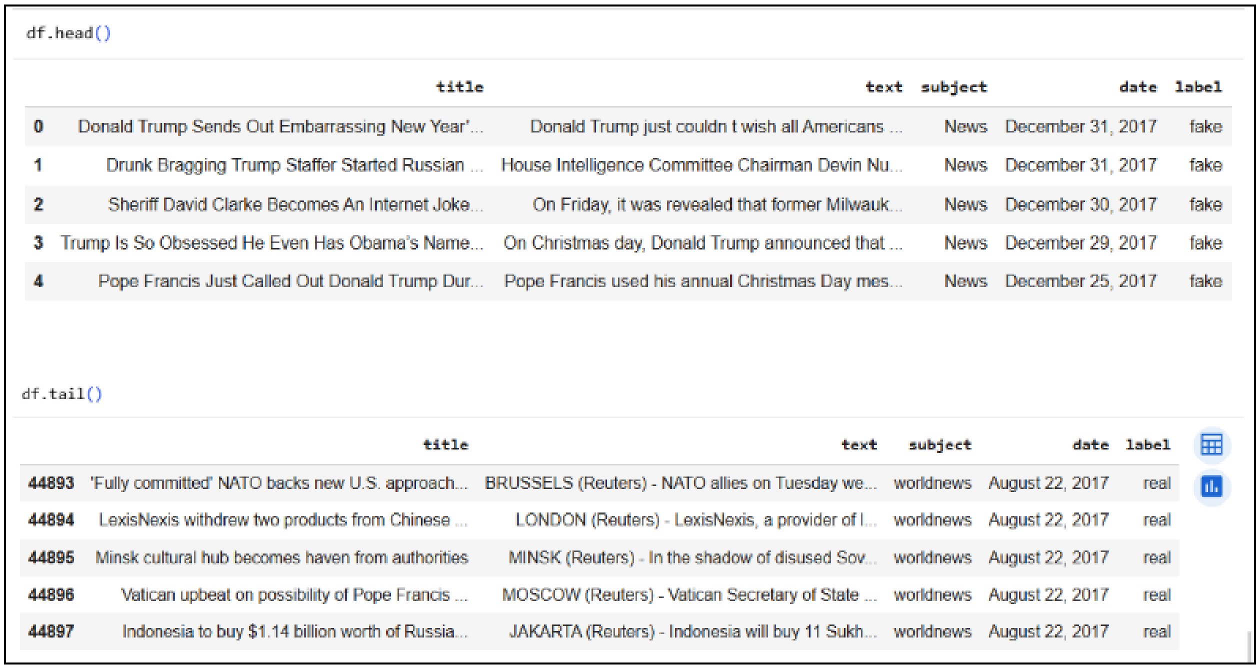
IV. **Data Handling & Visualization**

- Pandas, NumPy (data manipulation)
- Matplotlib, Seaborn (plots & graphs)
- WordCloud (visual analysis)

3. <u>Solution Approach</u>

Step 1: Loading and Preparing Input Data

The system begins by loading the dataset containing fake and real news articles. The text data is standardized by combining titles and body content into a single field. This ensures that the model receives complete contextual information for classification. The dataset is cleaned by removing duplicates, handling null values, and creating a unified structure for further preprocessing and model training.



```
df.head()
```

	title	text	subject	date	label
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017	fake
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017	fake
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017	fake
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017	fake
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017	fake

```
df.tail()
```

	title	text	subject	date	label
44893	'Fully committed' NATO backs new U.S. approach...	BRUSSELS (Reuters) - NATO allies on Tuesday we...	worldnews	August 22, 2017	real
44894	LexisNexis withdrew two products from Chinese ...	LONDON (Reuters) - LexisNexis, a provider of l...	worldnews	August 22, 2017	real
44895	Minsk cultural hub becomes haven from authorities	MINSK (Reuters) - In the shadow of disused Sov...	worldnews	August 22, 2017	real
44896	Vatican upbeat on possibility of Pope Francis ...	MOSCOW (Reuters) - Vatican Secretary of State ...	worldnews	August 22, 2017	real
44897	Indonesia to buy $1.14 billion worth of Russia...	JAKARTA (Reuters) - Indonesia will buy 11 Sukh...	worldnews	August 22, 2017	real

Fig 1: Overview of the dataset before cleaning

Step 2: Text Preprocessing and Cleaning

The news articles undergo a multi-stage preprocessing pipeline designed to remove noise, extract meaningful tokens, and prepare the text for feature extraction.
This includes operations such as:

a) HTML Tag Removal:

HTML fragments embedded within the text are removed using BeautifulSoup to clean the content and avoid irregular token patterns.

b) Lowercasing:

All text is converted to lowercase for consistency and to avoid treating the same word differently (e.g., “President” vs. “president”).

c) URL, Number, and Punctuation Removal:

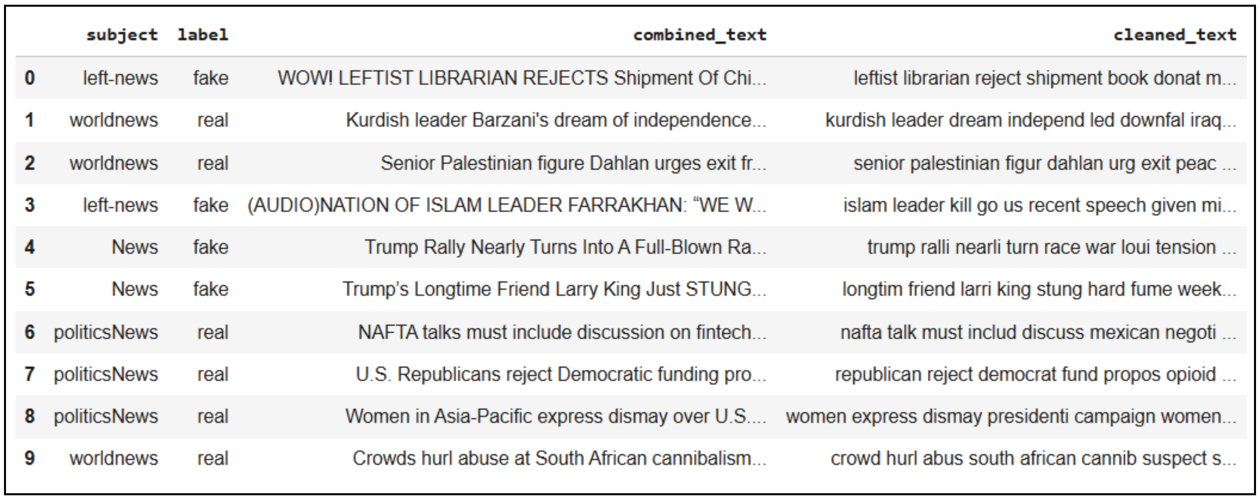
Regular expressions (regex) are used to remove links, special characters, and numeric sequences that do not contribute to classification.

d) Stopword Removal:

Non-informative words such as “the”, “is”, “at”, etc., are removed using NLTK to improve classifier focus.

e) Stemming:

Tokens are reduced to their root forms to normalize variations(e.g., “running”, “runs”, “ran” → “run”). This ensures that the cleaned text is uniform, meaningful, and optimized for vectorization.



	subject	label	combined_text	cleaned_text
0	left-news	fake	WOW! LEFTIST LIBRARIAN REJECTS Shipment Of Chi...	leftist librarian reject shipment book donat m...
1	worldnews	real	Kurdish leader Barzani's dream of independence...	kurdish leader dream independ led downfal iraq...
2	worldnews	real	Senior Palestinian figure Dahlan urges exit fr...	senior palestinian figur dahlan urg exit peac ...
3	left-news	fake	(AUDIO)NATION OF ISLAM LEADER FARRAKHAN: “WE W...	islam leader kill go us recent speech given mi...
4	News	fake	Trump Rally Nearly Turns Into A Full-Blown Ra...	trump ralli nearli turn race war loui tension ...
5	News	fake	Trump’s Longtime Friend Larry King Just STUNG...	longtim friend larri king stung hard fume week...
6	politicsNews	real	NAFTA talks must include discussion on fintech...	nafta talk must includ discuss mexican negoti ...
7	politicsNews	real	U.S. Republicans reject Democratic funding pro...	republican reject democrat fund propos opioid ...
8	politicsNews	real	Women in Asia-Pacific express dismay over U.S....	women express dismay presidenti campaign women...
9	worldnews	real	Crowds hurl abuse at South African cannibalism...	crowd hurl abus south african cannib suspect s...

Fig 2: Overview of the dataset after cleaning and preprocessing

Step 3: Feature Extraction Using TF–IDF

The cleaned text is transformed into numerical vectors using TF–IDF, which assigns weights based on the frequency and importance of words. This produces high-dimensional feature vectors that serve as input for machine learning models. The vectorizer is fitted on the training set and then applied consistently to both training and testing datasets.

Step 4: Machine Learning Model Training

To develop a robust misinformation detection system, multiple supervised learning models were implemented and evaluated. The initial baseline models included:

- Logistic Regression – a linear classifier suitable for high-dimensional sparse TF–IDF features.
- Random Forest – a nonlinear ensemble capable of capturing complex decision boundaries.

In addition to the baselines, two ensemble strategies were explored to further enhance predictive performance:

- Hard Voting Ensemble *(Logistic Regression + Support Vector Machine + Passive Aggressive Classifier):* This ensemble aggregates discrete class labels from heterogeneous classifiers. While it improved robustness to individual model errors, its performance was constrained by the non-probabilistic nature of several participating models.

- Soft Voting Ensemble *(Logistic Regression + Random Forest):*This ensemble combines class probability distributions generated by the constituent models. The probabilistic fusion enables smoother decision boundaries, improved calibration, and better handling of ambiguous samples.

Consequently, the Soft Ensemble was selected as the final deployed classifier for the misinformation detection module.

Step 5: Fake/Real News Prediction

Once trained, the model predicts whether an input article is misinformation. Users provide a text sample through the interface, which is preprocessed and passed to the model. The output label (0 = Fake, 1 = Real) is displayed along with model confidence.

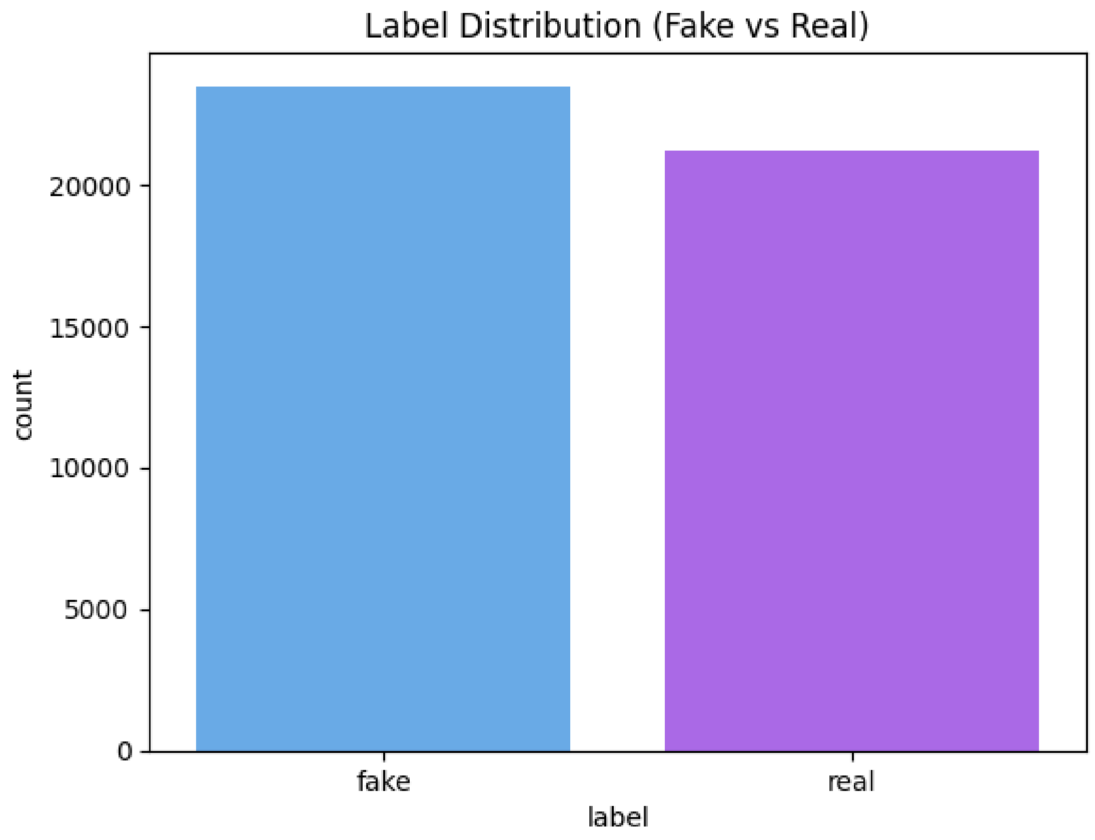


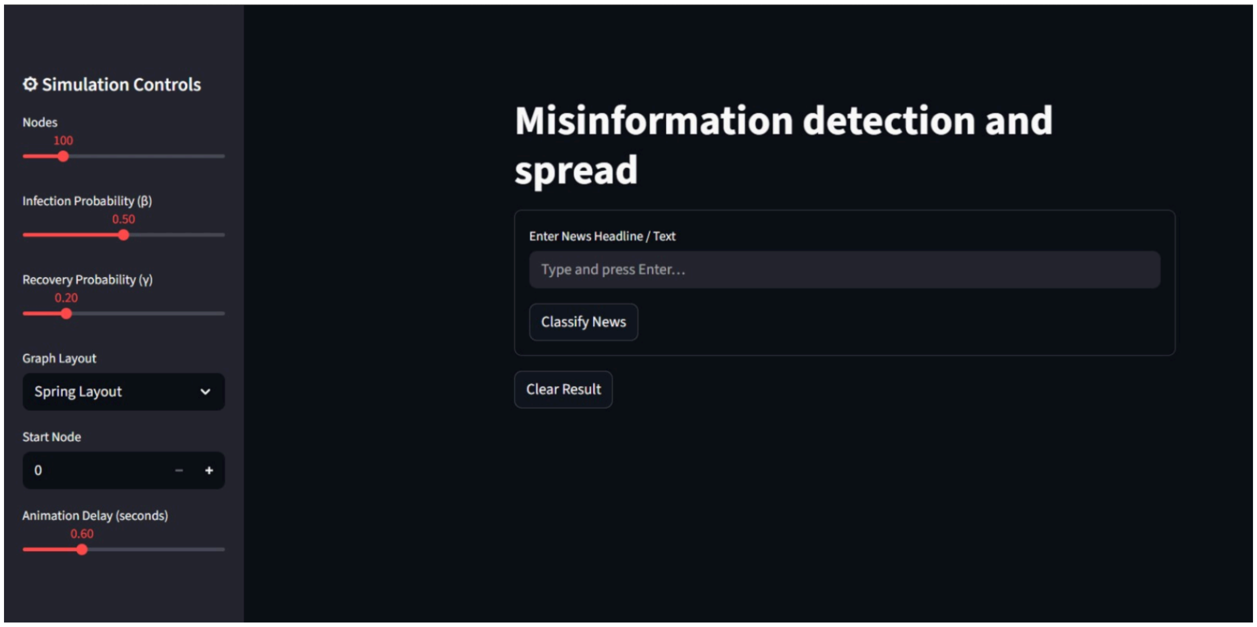
Fig 3: The figure above shows the overall distribution of labels in our cleaned dataset

Step 6: User Selection for Misinformation Spread Simulation

If the news is classified as fake (0), the system prompts the user to choose how they want to view misinformation propagation:

● Hub-based spread – misinformation starts from the most connected nodes
● Bridge-based spread – misinformation begins from nodes acting as connectors between communities.

This choice influences which node becomes the initial infected source in the network.



⚙ Simulation Controls

Nodes: 100

Infection Probability (β): 0.50

Recovery Probability (γ): 0.20

Graph Layout: Spring Layout

Start Node: 0

Animation Delay (seconds): 0.60

Misinformation detection and spread

Enter News Headline / Text

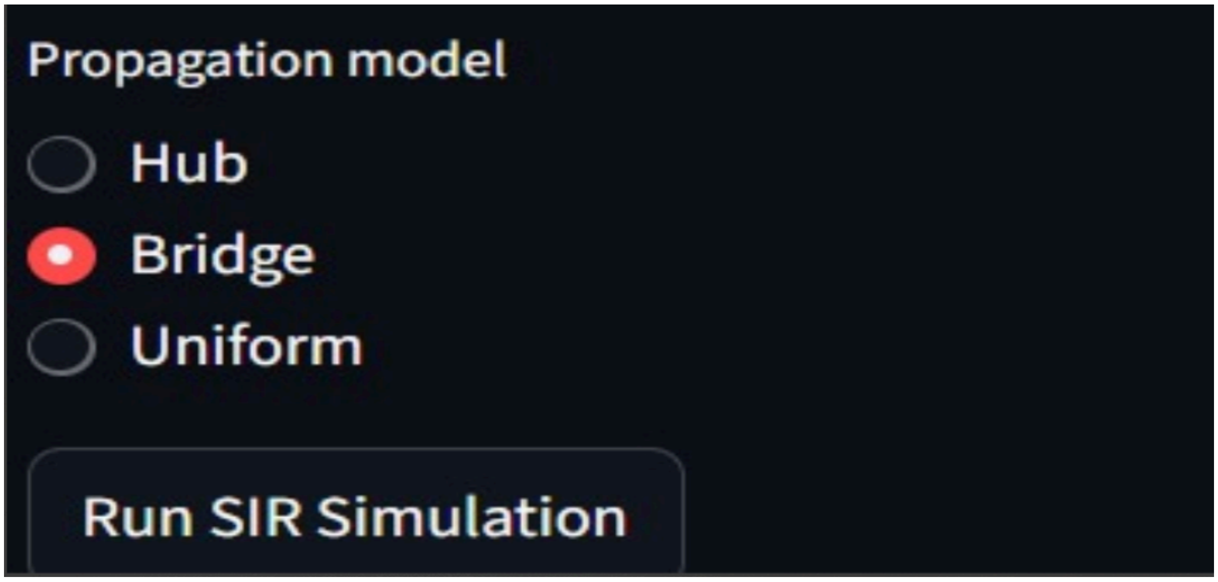
Fig 4: The figure above shows the user-interface for studying the misinformation flow

Step 7: Graph Construction and Centrality Analysis

A scale-free Barabási-Albert (BA) graph is generated using NetworkX to simulate a realistic social media network where few nodes have very high degrees. Next, the system calculates:

- Degree Centrality – identifies hubs
- Betweenness Centrality – identifies bridges

Based on the user's selection, the corresponding node with the highest centrality score is marked as the initial misinformation source.


Propagation model

Hub [✓ Bridge] Uniform

Fig 5: User options for analysing misinformation flow

The image above shows the various options available to user for analysing the flow of misinformation

Step 8: SIR Model Simulation

The misinformation propagation is simulated using the classical SIR model with three states:

- S – Susceptible (unexposed)
- I – Infected (misinformed)
- R – Recovered (corrected / no longer spreading misinformation)

During each iteration:

- Infected nodes attempt to spread misinformation to neighbors based on infection probability.
- Nodes transition to a recovered state based on recovery probability.
- The simulation continues until no infected nodes remain.

Node colors change dynamically (initially blue, infected-red, recovered-green) to visualize the evolving states.

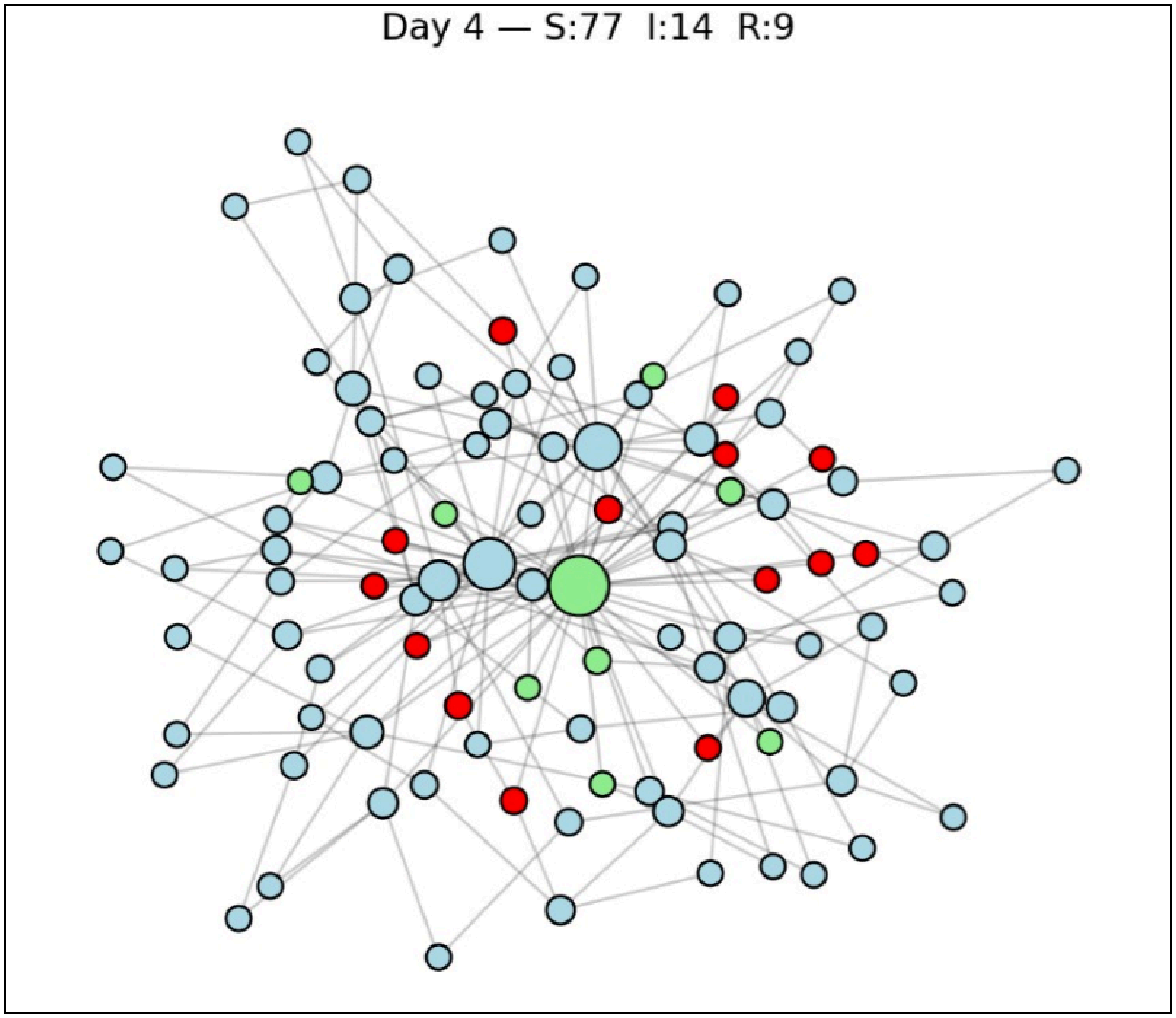


Fig 6: The figure above shows dynamic color changes to visualise evolving states

Step 9: Real-Time Graph Visualization

The graph visualization updates at each simulation step.The user sees misinformation spreading across hubs/bridges, stabilizing, and eventually dying out. The visualization provides intuitive insight into how misinformation travels through networks.

<u>Key Features of the Solution</u>

1. Interactive User Input

▪ Users can enter real news text and immediately see classification results.

▪ If misinformation is detected, users choose how they want to view its spread (hub vs bridge).

2. Real-Time SIR Visualization

▪ Graph updates dynamically to represent misinformation propagation.

▪ Node colors provide clear feedback on infection and recovery states.

3. Accurate Text Classification

▪ Advanced preprocessing and ensemble modeling ensure strong performance.

▪ TF–IDF captures essential linguistic features effectively.

4. Scalability & Flexibility

▪ The modular architecture allows new models or graph types to be integrated easily.

▪ Simulation parameters can be extended for further complexity.

5. Cost-Effective and Accessible

▪ Uses standard Python libraries and runs on typical hardware.

▪ Requires no specialized equipment or external APIs.

Overcoming Obstacles

Throughout the development and integration of the project, various challenges were encountered. Below are major challenges encountered during implementation and the strategies adopted:

1. Classification Accuracy Issues

Obstacle: Some articles with complex writing styles (sarcasm, satire, ambiguous content) were misclassified.

Solution:

i. Improved preprocessing pipeline for better token consistency.

ii. Added ensemble modeling to stabilize predictions.

iii. Increased training samples by combining title and body text.

2. High Dimensionality of TF–IDF Vectors

Obstacle: Vectorization produced extremely large feature spaces.

Solution:

i. Limited max_features in TF–IDF.

ii. Used soft voting to avoid overfitting in high-dimensional space.

3. Centrality Misidentification in Small Graphs

Obstacle: Bridge/hub detection was inconsistent in small or dense networks.

Solution:

i. Applied normalization to centrality values.

ii. Used a BA graph for consistent network topology.

4. Performance Bottlenecks in Visualization

Obstacle: Large graphs caused delays in plotting updates.

Solution:

i. Reduced graph size for real-time demonstration.

ii. Optimized drawing functions and minimized redraw frequency.

5. SIR Model Instabilities

Obstacle: Certain probability values caused simulations to stall.

Solution:

i. Added safeguards to stop simulation when all nodes reach terminal states.

ii. Introduced minimum thresholds for infection and recovery.

6. Incorrect Node Selection by Users

Obstacle: Users attempted to simulate spread without selecting spread type.

Solution:

i. Added validation checks before launching the simulation.

ii. Integrated exception messages for invalid inputs.

7. Scalability Issues for Larger Networks

Obstacle: Very large networks slowed down both SIR loops and rendering.

Solution:

i. Process only active infected nodes in each iteration.

ii. Allow dynamic configuration of graph size.

Chapter 4: Modelling and Implementation Details

1.Design Diagram

1. Use Case Diagram

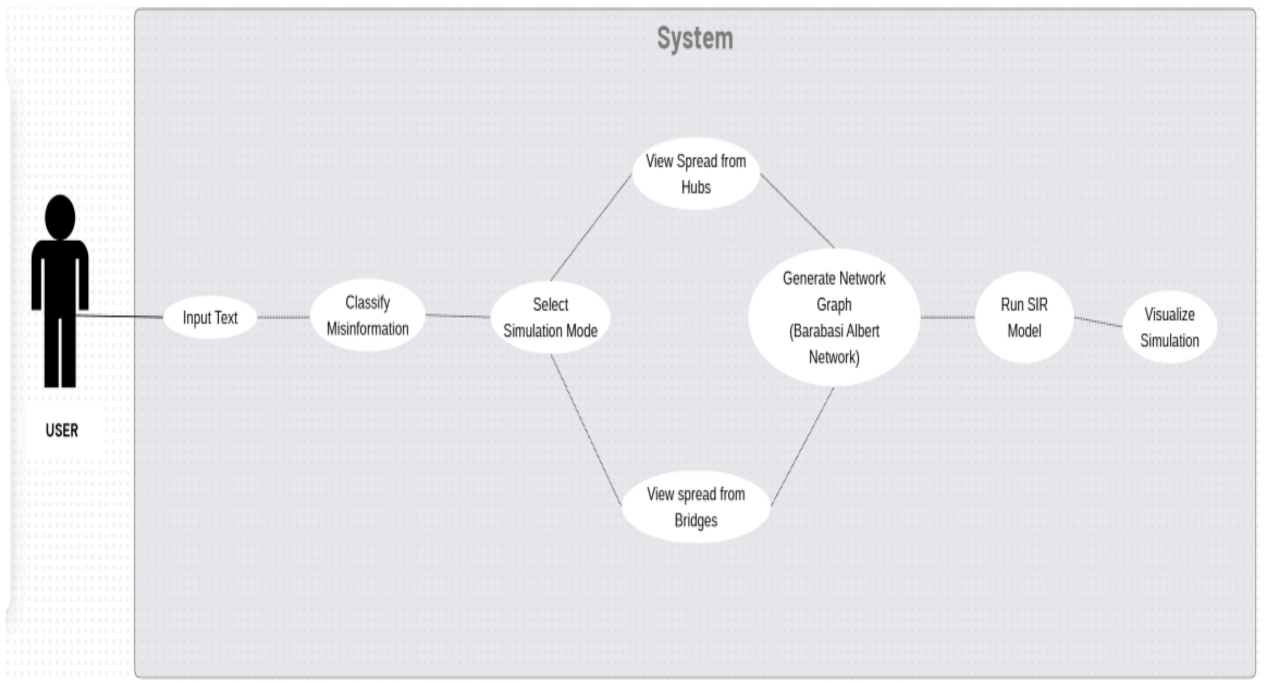


Fig 7:Workflow of user text classification and misinformation spread simulation.

The figure shows how a user inputs text for classification and, if misinformation is detected, how the system proceeds to simulate its spread through various network modes.

2. Flow Chart

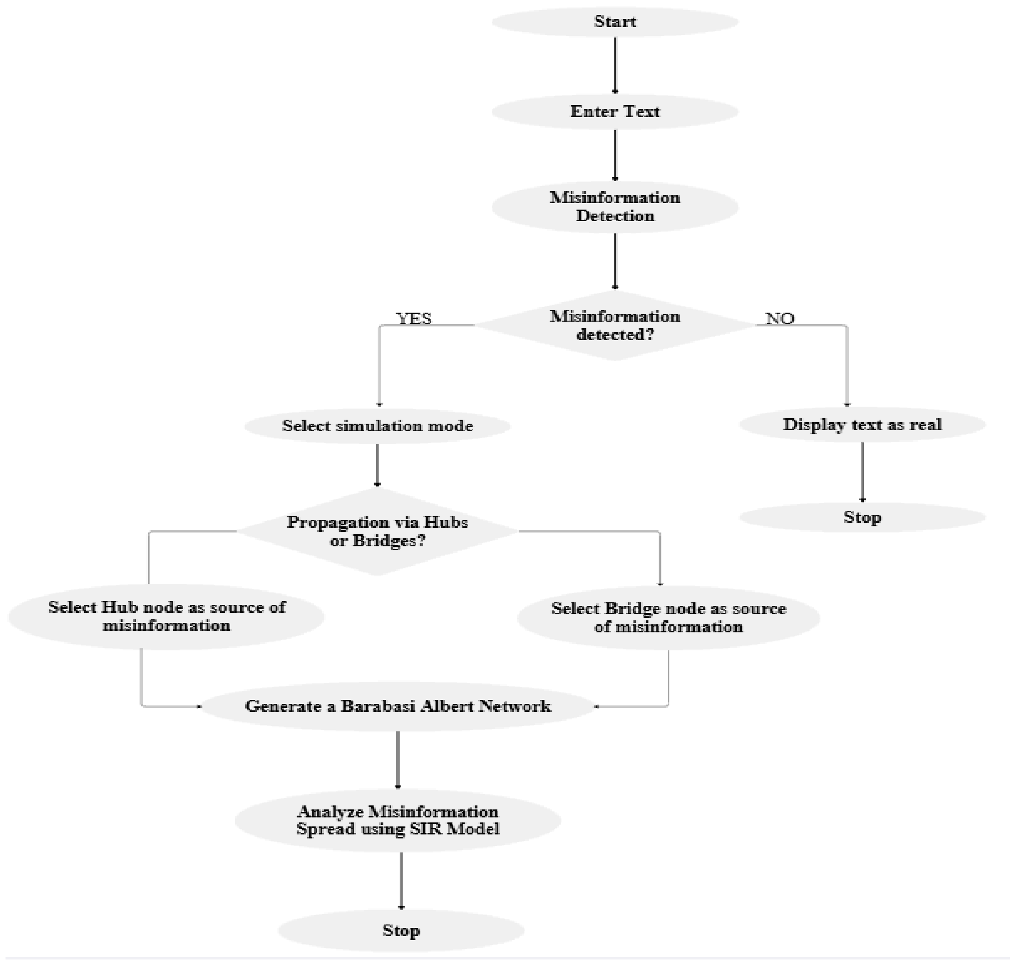

Start
Enter Text
Misinformation Detection
Misinformation detected?
YES
NO
Select simulation mode
Display text as real
Stop
Propagation via Hubs or Bridges?
Select Hub node as source of misinformation
Select Bridge node as source of misinformation
Generate a Barabasi Albert Network
Analyze Misinformation Spread using SIR Model
Stop

Fig 8: Flowchart of misinformation detection and spread simulation.

The flowchart shows how user text is classified for misinformation and, if detected, how the system simulates and visualizes its spread through a selected network model.

3. Class Diagram

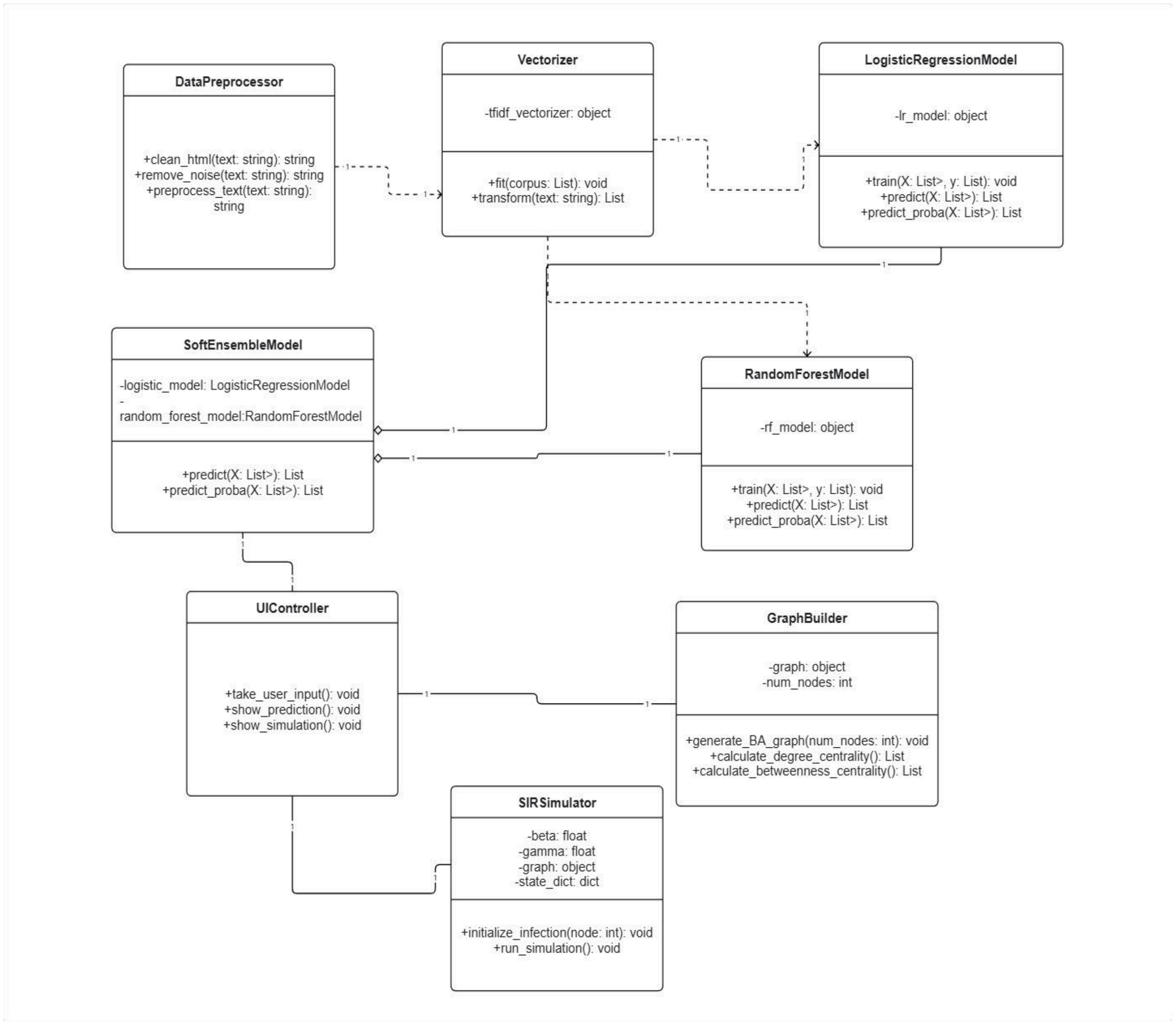

DataPreprocessor
+clean_html(text: string): string
+remove_noise(text: string): string
+preprocess_text(text: string): string
Vectorizer
-tfidf_vectorizer: object
+fit(corpus: List): void
+transform(text: string): List
LogisticRegressionModel
-lr_model: object
+train(X: List>, y: List): void
+predict(X: List>): List
+predict_proba(X: List>): List
SoftEnsembleModel
-logistic_model: LogisticRegressionModel
-random_forest_model:RandomForestModel
+predict(X: List>): List
+predict_proba(X: List>): List
RandomForestModel
-rf_model: object
+train(X: List>, y: List): void
+predict(X: List>): List
+predict_proba(X: List>): List
UIController
+take_user_input(): void
+show_prediction(): void
+show_simulation(): void
GraphBuilder
-graph: object
-num_nodes: int
+generate_BA_graph(num_nodes: int): void
+calculate_degree_centrality(): List
+calculate_betweenness_centrality(): List
SIRSimulator
-beta: float
-gamma: float
-graph: object
-state_dict: dict
+initialize_infection(node: int): void
+run_simulation(): void

Fig 9: Class diagram of the system's core modules and interactions.

This class diagram presents the core modules of the system and their interactions across preprocessing, feature extraction, machine-learning classification, graph construction, and SIR-based misinformation simulation.

2.Implementation Details and Issues

Implementation Details:

The implementation consists of several key components:

1. Text Preprocessing & Cleaning:

The input news text is processed through multiple steps such as lowercasing, removing URLs, punctuation cleaning, stopword removal, and stemming/tokenization.This ensures that the text is converted into a meaningful and structured format suitable for machine learning models.

2. Fake/Real Classification using ML Models:

The cleaned text is vectorized using TF–IDF, and multiple models such as Logistic Regression, Random Forest, and a Soft Voting Ensemble are trained to classify news as fake (0) or real (1). The ensemble helps improve reliability by combining strengths of linear and nonlinear models.

3. Misinformation Spread Simulation (SIR + SNA):

After the classification, the user interacts with the UI to choose whether they want to simulate spread through hub nodes or bridge nodes, both identified using network analysis metrics such as degree centrality and betweenness centrality. The SIR model is used to simulate how misinformation infects nodes, spreads across the network, and eventually stabilizes when nodes recover.

4. Network Visualization:

NetworkX is used to generate and visualize a Barabási–Albert graph. Nodes change color dynamically (Susceptible → Infected → Recovered) to demonstrate the flow of misinformation across the network.

Key Libraries Used:

- pandas – for dataset manipulation
- scikit-learn – for preprocessing, vectorization, and ML models
- NetworkX – for graph generation and centrality calculations
- matplotlib / seaborn – for plotting and visualizing simulations
- numpy – for mathematical operations

- nltk / re / bs4 – for text preprocessing

Issues Faced:

1. Noisy or Unstructured Input Text:

The model struggled with input containing slang, sarcasm, emojis, or incomplete sentences.

2. Class Imbalance During Training:

Fake and real news were not perfectly balanced, leading to biased predictions before applying balancing techniques.

3. Overfitting of Certain Models:

Models such as Random Forest sometimes overfitted the TF–IDF vectors due to high dimensionality.

4. Centrality-Based Node Selection Variability:

In some graphs, hubs or bridges were not clearly distinguishable, making simulation outcomes inconsistent.

5. SIR Model Instability:

For certain probabilities (infection = 0 or recovery = 0), the model produced unrealistic infinite loops or no spread at all, requiring careful tuning of parameters.

3.Risk Analysis and Mitigation

Risk 1: Misclassification of News Articles

Risk: The ML model may incorrectly classify ambiguous or highly contextual news text.

Mitigation: Improve the preprocessing pipeline, use more training samples, and fine-tune ensemble models using cross-validation.

Risk 2: Poor Generalization to Real-World News

Risk: The model is trained only on a Kaggle dataset and may not generalize to global, regional, or new misinformation types.
Mitigation: Expand dataset sources, include multilingual or multi-domain content, and periodically retrain the model.

Risk 3: Unreliable Spread Simulation on Small Networks

Risk: The SIR model may produce unrealistic propagation behavior on very small or synthetic graphs.
Mitigation: Introduce adjustable parameters (infection rate, recovery rate) and allow users to select different graph types (BA, ER, WS).

Risk 4: Incorrect Hub/Bridge Identification

Risk: Degree and betweenness centrality may oversimplify a node's influence.
Mitigation: Incorporate more metrics (closeness, eigenvector centrality) and give users the ability to override the automatic selection.

Risk 5: Performance Issues During Large Network Visualization

Risk: Visualizing large graphs can result in lag, poor rendering, or application freezing.
Mitigation: Limit graph size for real-time simulation, optimize layout algorithms, and offer static rendering options for larger networks.

Risk 6: Limited User Interaction and UI Constraints

Risk: Users may find it difficult to adjust parameters or interpret the simulation outcomes.
Mitigation: Provide clear instructions, add UI sliders for infection/recovery rates, and offer tooltips explaining hubs, bridges, and SIR states.

Risk 7: Heavy Performance Impact

Risk: Running multiple models, TF–IDF vectorization, and SIR simulation simultaneously can cause high CPU usage, especially on low-end devices.

Mitigation: Optimize computation pipelines, reduce vector dimensions, allow GPU acceleration, and enable adjustable simulation speed.

Chapter 5: Testing

1. Testing Plan

The testing plan outlines a systematic strategy to validate the accuracy, reliability, and overall performance of the misinformation detection system and its simulation module. The objective is to ensure that each module functions correctly on its own and integrates smoothly into the complete workflow. The testing procedure is divided into multiple stages:

● **Unit Testing:**

This phase focuses on testing individual components in isolation.
Key units include:
1. Text preprocessing functions
2. Machine Learning model (prediction pipeline)
3. Label encoding and data validation
4. Network graph generation
5. SIR propagation logic

Each unit is tested separately to confirm that it performs computations correctly and handles edge cases such as empty inputs or malformed text.

● **Integration Testing:**

Once individual components are validated, integration testing ensures that they interact cohesively.
This includes:
1. Passing user text through preprocessing → ML classifier → prediction output
2. Feeding the detected misinformation into the simulation workflow
3. Ensuring seamless data flow between the simulation mode selector, graph generator, and SIR model. This step verifies that modules communicate correctly and no mismatches occur between outputs and expected inputs.

● **System Testing:**

In this stage, the complete system is evaluated end-to-end under real usage scenarios.
Tests include:
1. Inputting various types of text (real, fake, borderline, noisy data)
2. Verifying that the system correctly identifies misinformation
3. Running network simulations using hub-based and bridge-based spread modes
4. Checking that the SIR model animates infection spread accurately

This ensures that the full workflow behaves as intended when executed in real-world conditions**.**

● **User Acceptance Testing (UAT):**

UAT involves engaging real users to validate usability and system responsiveness.
Users test:
1. How easily they can input text
2. Whether the detection results are clear and intuitive
3. Whether the simulation controls are understandable
4. How effectively the final visualization communicates misinformation spread

Feedback collected from users is used to enhance system clarity, UX design, and overall usability.

2. <u>Component Decomposition and Type of Testing Required</u>

The project consists of multiple interconnected components, each responsible for a specific stage in the misinformation detection and spread simulation workflow. To ensure accuracy, reliability, and robustness, each component requires targeted testing. The components and the corresponding testing strategies are described below:

I. Text Input and Data Handling

This component accepts user-provided news text and validates it before processing.

- Unit Testing: Check if text input is correctly captured. Validate handling of empty, extremely long, or noisy inputs.

- Integration Testing: Ensure smooth data transfer into preprocessing and classification modules.

II. Text Preprocessing Pipeline

This stage includes stopword removal, stemming/lemmatization, HTML cleansing, punctuation removal, and token filtering.

- Unit Testing: Verify each preprocessing step individually (e.g., stopword removal works correctly). Ensure the function handles malformed text, special characters, or missing values.
- Integration Testing: Test that the final cleaned text correctly feeds into the ML classifier.

III. Machine Learning Classification (Fake/Real)

This module uses a trained ML model (e.g., Logistic Regression, Random Forest, or Soft Voting Ensemble) to classify news as misinformation or genuine.

- Unit Testing: Ensure the model loads correctly and produces predictions. Validate label encoding and prediction format (0/1 or real/fake).
- Integration Testing: Test end-to-end flow from cleaned text → numerical vectorization → prediction output.
- System Testing: Check classification accuracy across diverse inputs (balanced, imbalanced, ambiguous cases).

IV. User Simulation Mode Selection (Hubs / Bridges)

After misinformation is detected, the user selects a spread simulation mode.

- Unit Testing: Ensure both options (hub-based and bridge-based) are selectable.
- Integration Testing: Validate that the selected mode influences source node selection in the network graph.

V. Network Graph Generation (Barabási–Albert Model)

This component generates a scale-free network used for spread simulation.

- Unit Testing: Verify graph creation (correct number of nodes, proper degree distribution).
- Integration Testing: Check that the generated network supports centrality calculations and SIR operations without errors.

VI. Centrality Calculation (Hub/Bridge Identification)

This module computes degree centrality or betweenness centrality depending on the spread mode.

- Unit Testing: Validate correctness of centrality values for sample graphs.
- Integration Testing: Ensure selected centrality aligns with the user's chosen spread mode.

VII. SIR Propagation Model

Implements the Susceptible–Infected–Recovered model to simulate how misinformation spreads.

- Unit Testing: Validate state transitions (S → I → R) and probability parameters (β, γ). Ensure time-step iteration works without logical or index errors.
- Integration Testing: Confirm that the initial infected node is correctly passed from the centrality module.
- System Testing: Run multi-step propagation to check stability, accuracy, and stopping conditions.

VIII. Visualization and Animation Engine

Displays the network graph with dynamic color changes for each state (Susceptible, Infected, Recovered).

- Unit Testing: Ensure nodes change color as per SIR state. Validate rendering functions (matplotlib, PyVis, etc.).
- Integration Testing: Confirm synchronization between SIR outputs and visual updates.
- System Testing: Check real-time animation performance for different graph sizes.

IX. Error Handling and Robustness Checks

Handles issues such as invalid input, model load errors, missing data, or computational failures.

- Unit Testing: Validate all exception messages.
- System Testing: Simulate failure scenarios (e.g., empty text, corrupted model file, invalid graph parameters).

X. Application Exit and Graceful Shutdown

Ensures safe termination of the simulation and release of memory resources.

i. Unit Testing: Check exit triggers (e.g., close button or UI command) properly terminate the simulation.

ii. Integration Testing:Ensure all components stop in sequence without freezes, hanging threads, or resource leaks.

3. <u>List of All Test Cases in Prescribed Format</u>

Below is a more comprehensive list of test cases that will be employed to ensure the system's reliability:

● Test Case 01: Text Input Capture

Objective: Verify if the system correctly captures and processes the input news text.

Expected Result: The input text should be accepted and passed to the preprocessing pipeline.

Test Steps:

- ▪ Open the application interface.
- ▪ Enter a sample news headline or paragraph.
- ▪ Submit the text for classification.

Edge Cases: Empty or excessively long text, text containing special characters or HTML tags.

Outcome: The system should display an appropriate error message if the input text is invalid.

- ● **Test Case 02: Text Preprocessing**

Objective: Ensure that the preprocessing operations (cleaning, lowercasing, stopword removal, stemming) are performed accurately.

Expected Result: The text should convert into a cleaned and tokenized format suitable for model input.

Test Steps:

- ▪ Input a raw text sample containing punctuation, URLs, numbers, and stopwords.
- ▪ Apply the preprocessing function.
- ▪ Verify whether the cleaned text output is correct.

Outcome: The cleaned text should contain meaningful processed tokens with noise removed.

- ● **Test Case 03: Fake/Real Classification**

Objective: Validate that the machine learning model predicts misinformation accurately.

Expected Result: The system should classify text as either Fake (0) or Real (1).

Test Steps:

- ▪ Input a known fake news sample.
- ▪ Input a known real news sample.
- ▪ Compare predicted labels with expected results.

Edge Cases: Very short or highly ambiguous news text.

Outcome: The classification should be consistent and should not produce irrelevant outputs.

● **Test Case 04: Spread Mode Selection**

Objective: Confirm that the selected spread mode (hub-based or bridge-based) is correctly applied**.**

Expected Result: The correct centrality-based node should be chosen as the initial infection point.

Test Steps:

▪ Select "Hub-based Spread" mode.

▪ Verify that the highest-degree node is selected.

▪ Select "Bridge-based Spread" mode.

▪ Verify that the top betweenness node is selected.

Outcome: The simulation should accurately follow the user-selected spreading mechanism.

● **Test Case 05: Network Graph Generation**

Objective: Ensure that the system generates a valid network graph for simulation.

Expected Result: The network graph with the specified number of nodes should be generated correctly.

Test Steps:

▪ Enter the desired number of nodes.

▪ Generate the graph.

▪ Verify node count, connectivity, and structure.

Outcome: A valid and complete graph should be generated without errors**.**

● **Test Case 06: SIR-Based Misinformation Spread Simulation**

Objective: Validate that the SIR model correctly simulates the spread of misinformation.

Expected Result: Nodes should change state from S → I → R according to simulation parameters.

Test Steps:

▪ Start the simulation with a selected source node.

▪ Observe state transitions over multiple steps.

▪ Check if spread and recovery rates behave as expected.

Edge Cases: Spread rate = 0, recovery rate = 1.

Outcome: The simulation flow should match SIR dynamics without incorrect transitions.

● **Test Case 07: Real-Time Visualization**

Objective: Ensure that the visual representation of node states updates correctly in real time.

Expected Result: Nodes should visibly change color based on their state (Susceptible, Infected, Recovered).

Test Steps:

▪ Initiate the simulation.

▪ Observe color updates for each timestep.

Outcome: The visualization should accurately reflect real-time simulation output.

● **Test Case 08: Error Handling (Invalid Input / Missing Model)**

Objective: Test the system's response to invalid input or system errors.

Expected Result: The system should show meaningful error messages and handle interruptions gracefully.

Test Steps:

▪ Leave the input field blank.

▪ Remove or rename the model file temporarily.

▪ Enter invalid simulation parameters.

Outcome: Errors should be caught without crashing the system.

● **Test Case 09: Performance and Responsiveness**

Objective: Verify that the simulation performs smoothly without lag.

Expected Result: The application should respond quickly and render updates smoothly.

Test Steps:

▪ Generate a graph with a large number of nodes (e.g., 300–500).

▪ Run the simulation.

▪ Monitor for delays or freezing.

Outcome: The system should remain responsive during computation and visualization.

● **Test Case 10: Exit Mechanism**

Objective: Verify the application's shutdown and reset functionality.

Expected Result: The system should exit or reset without unexpected behavior.

Test Steps:

▪ Start a misinformation simulation.

▪ Use the exit/reset option.

▪ Confirm that all processes terminate correctly.

Outcome: The program should close smoothly without leaving any active resources.

4.<u>Error and Exception Handling</u>

The system is designed to be robust and handle various types of errors gracefully:

I. Invalid or Empty Text Input

If the user enters an empty news text, extremely short input, or input containing only symbols, the system will display an error message and prevent the classification process from starting. This ensures that only meaningful text is passed through the preprocessing pipeline.

II. Preprocessing Failures

If the text contains unexpected patterns such as excessive HTML tags, special characters, or unsupported Unicode symbols, the system will automatically clean the input and remove
invalid tokens. This prevents errors during tokenization, vectorization, or embedding generation.

III. Model Prediction Errors

If the machine learning model encounters difficulty in generating predictions due to malformed input
or unusually long text, the system will safely handle the exception and notify the user.
This avoids situations where the model would produce random or undefined outputs.

IV. Invalid Node Selection for SIR Simulation

If the user attempts to run the misinformation spread simulation without selecting a hub or bridge node,
or selects a node not present in the graph, the system will display an error message and
halt the simulation. This ensures the SIR model always starts with a valid source node.

V. Graph Rendering or Visualization Errors

If the network visualization fails due to layout issues, missing connections, or UI refresh conflicts, the system will automatically fall back to a basic layout and maintain stability.
This prevents the application from freezing or crashing during graph display.

VI. SIR Model Computation Errors

If the simulation encounters cases such as no infected nodes, isolated nodes,
or probabilities that cause no transitions (infection = 0 or recovery = 0),
the system will terminate the simulation gracefully, avoiding infinite loops
or divide-by-zero errors during state updates.

VII. General Exceptions

The application includes try–except blocks for unexpected runtime errors such as
file access issues, memory limitations, or interrupted operations.
These safeguards prevent the system from crashing and provide the user with
a clear, meaningful error message.

5. Limitations of Solution

I. Dependence on Text Quality and Structure

The accuracy of misinformation classification relies heavily on the quality of the input text. Texts containing excessive slang, sarcasm, emojis, broken grammar, or incomplete information may not be correctly interpreted by the model.

Impact: The system may misclassify highly unstructured or noisy text samples.

II. Model Bias and Dataset Limitations

The model is trained on a specific dataset (Fake–Real News Dataset), which may not fully represent all types of misinformation, such as regional news, memes, satire, or political propaganda.

Impact: The classifier may fail to generalize well to new or unseen misinformation patterns.

III. Limited Context Understanding

Traditional ML models such as Logistic Regression, Random Forest, or Soft Voting Ensembles primarily analyze word-level patterns and cannot fully capture deeper semantic meaning or contextual nuances, unlike large transformer models.

Impact: Subtle misinformation, opinionated writing, or context-dependent claims may be incorrectly classified.

IV. No Real-Time Internet Verification

The system processes only the input text provided by the user and does not perform fact-checking or cross-referencing with credible sources, knowledge graphs, or live news APIs.

Impact: The classifier may label an article as real even if the content is outdated or factually incorrect today.

V. Simplified SIR Model Assumptions

The misinformation spread simulation is based on the classical SIR model, which assumes uniform infection probability, recovery probability, and static network structure.
In real social networks, these parameters vary widely.
Impact: The spread pattern shown in the simulation may not perfectly reflect real-world social media dynamics.

VI. Basic Graph Generation and Visualization Constraints

The simulation uses synthetic graphs (e.g., Barabási–Albert model) or static network inputs.Real networks are far more complex, with millions of nodes and dynamic connections. Graph visualization libraries may struggle with large-scale rendering.
Impact: The system cannot visualize very large or real-time social media networks.

VII. Fixed Hub and Bridge Detection Logic

The system identifies hubs based on degree centrality and bridges based on betweenness centrality. While standard, these measures may oversimplify influence in real social platforms where engagement metrics (likes, impressions, repost chains) also matter.
Impact: The selected hub/bridge nodes may not always represent the most influential users.

VIII. Limited User Interaction and Configuration

The current UI only allows choosing spread mode (hub-based or bridge-based) and visualizing propagation. Advanced controls such as changing infection rates, recovery rates, network structure, or simulation time are limited.
Impact: Users cannot completely customize spread scenarios for deeper analysis.

Chapter 6: Findings Conclusion and Future Work

6.1 FINDINGS

The implementation and evaluation of the misinformation detection system, combined with a Social Network Analysis (SNA) based propagation model yielded several significant technical findings. These insights are organized across two major components:
(1) machine-learning based text classification, and
(2) network-driven misinformation diffusion modelling using a discrete SIR framework.

1. Performance Analysis of Machine Learning Models

A range of baseline classifiers including Logistic Regression and Random Forest were trained using TF-IDF vectorization. Both hard and soft voting ensembles were evaluated, with the Soft Ensemble demonstrating consistently superior performance due to its probabilistic averaging of classifier outputs.

Superior Generalization of the Soft Ensemble

The Soft Ensemble achieved the highest overall performance:

- **Training Accuracy:** 96.4%
- **Testing Accuracy:** 95.6%

The confusion matrix (Figure 10) further showed strong classification reliability:

- **4438** true fake predictions
- **4106** true real predictions
- **257** false positives
- **136** false negatives

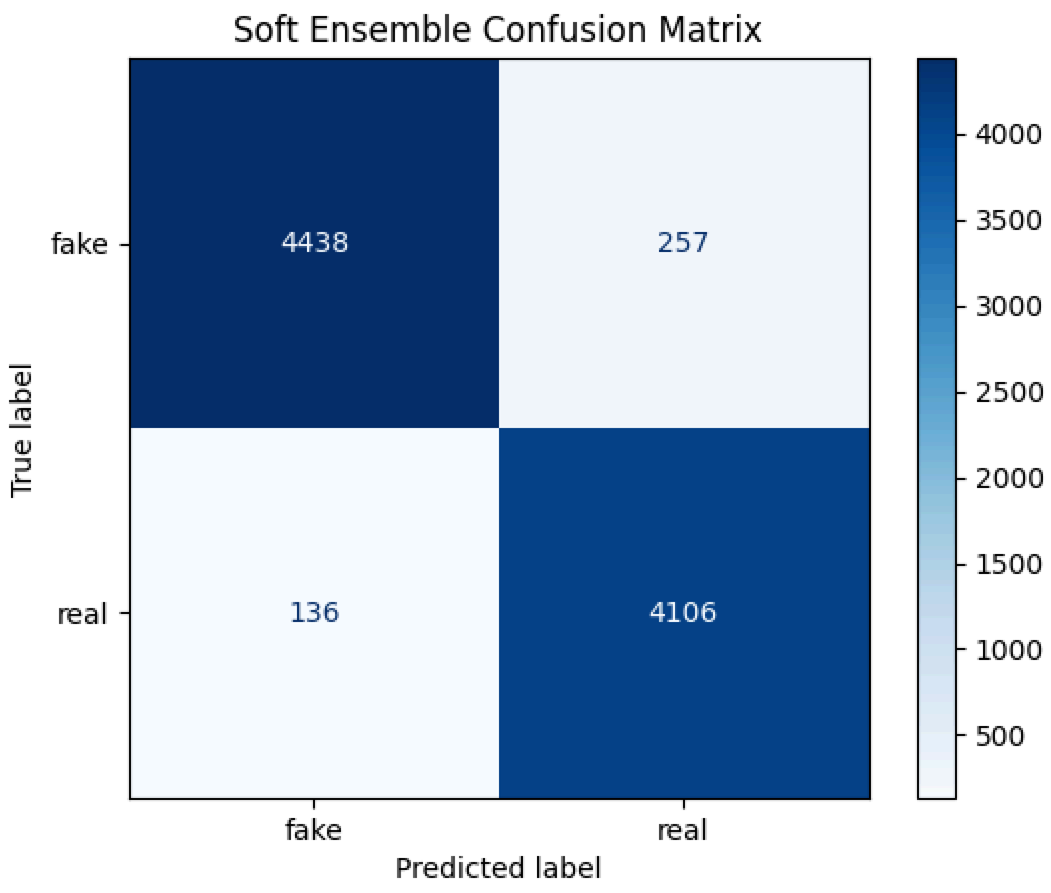
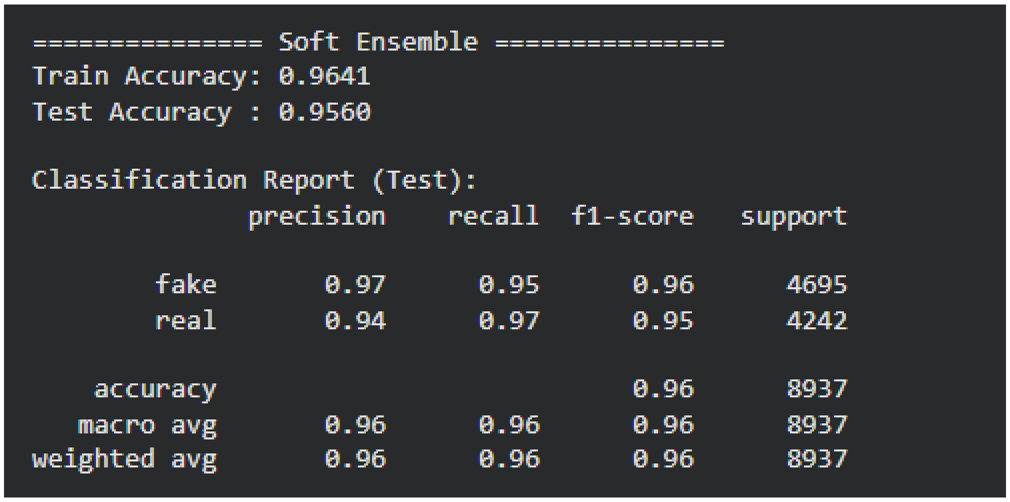


Fig 10: Confusion Matrix for Soft Voting Ensemble

The detailed classification metrics as shown in the figure 11 below, were:

- **Precision:** Fake = 0.97, Real = 0.94
- **Recall:** Fake = 0.95, Real = 0.97

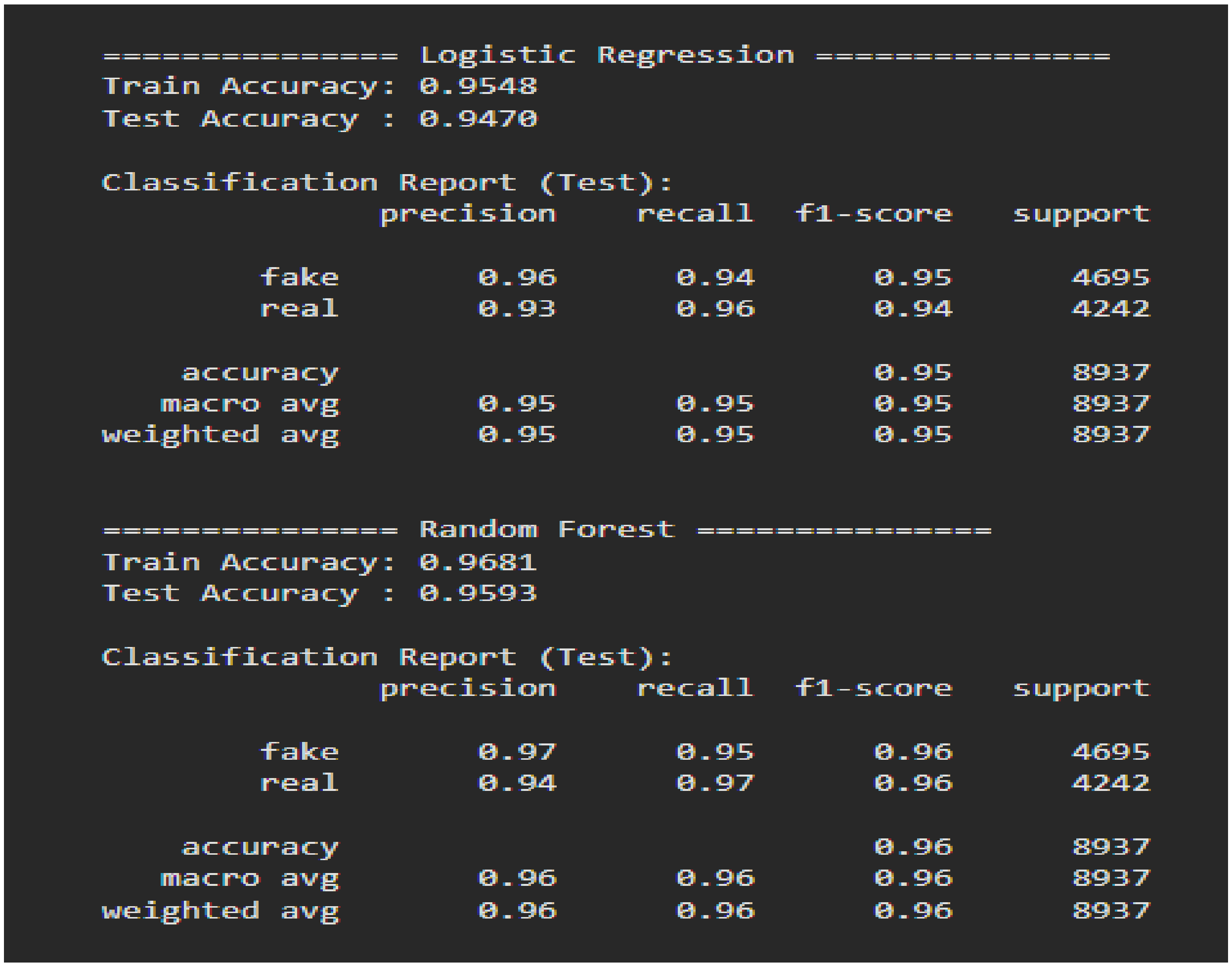
- **F1-Score:** Overall = 0.96



```
=============== Soft Ensemble ===============
Train Accuracy: 0.9641
Test Accuracy : 0.9560

Classification Report (Test):
              precision    recall  f1-score   support

        fake       0.97      0.95      0.96      4695
        real       0.94      0.97      0.95      4242

    accuracy                           0.96      8937
   macro avg       0.96      0.96      0.96      8937
weighted avg       0.96      0.96      0.96      8937
```

Fig 11: Detailed Classification Metrics of Soft Voting Ensemble

Compared to individual Logistic Regression and Random Forest models, which displayed lower stability across classes, the Soft Ensemble provided the most balanced precision–recall behaviour.



```
=============== Logistic Regression ===============
Train Accuracy: 0.9548
Test Accuracy : 0.9470

Classification Report (Test):
              precision    recall  f1-score   support

        fake       0.96      0.94      0.95      4695
        real       0.93      0.96      0.94      4242

    accuracy                           0.95      8937
   macro avg       0.95      0.95      0.95      8937
weighted avg       0.95      0.95      0.95      8937


=============== Random Forest ===============
Train Accuracy: 0.9681
Test Accuracy : 0.9593

Classification Report (Test):
              precision    recall  f1-score   support

        fake       0.97      0.95      0.96      4695
        real       0.94      0.97      0.96      4242

    accuracy                           0.96      8937
   macro avg       0.96      0.96      0.96      8937
weighted avg       0.96      0.96      0.96      8937
```

Fig 12: Accuracies of Individual Baseline Models

Robustness on Unseen Data

A diverse set of real and artificially constructed misinformation samples was tested.
The classifier accurately detected:

- fabricated political announcements,
- non-existent world events,
- exaggerated social claims.

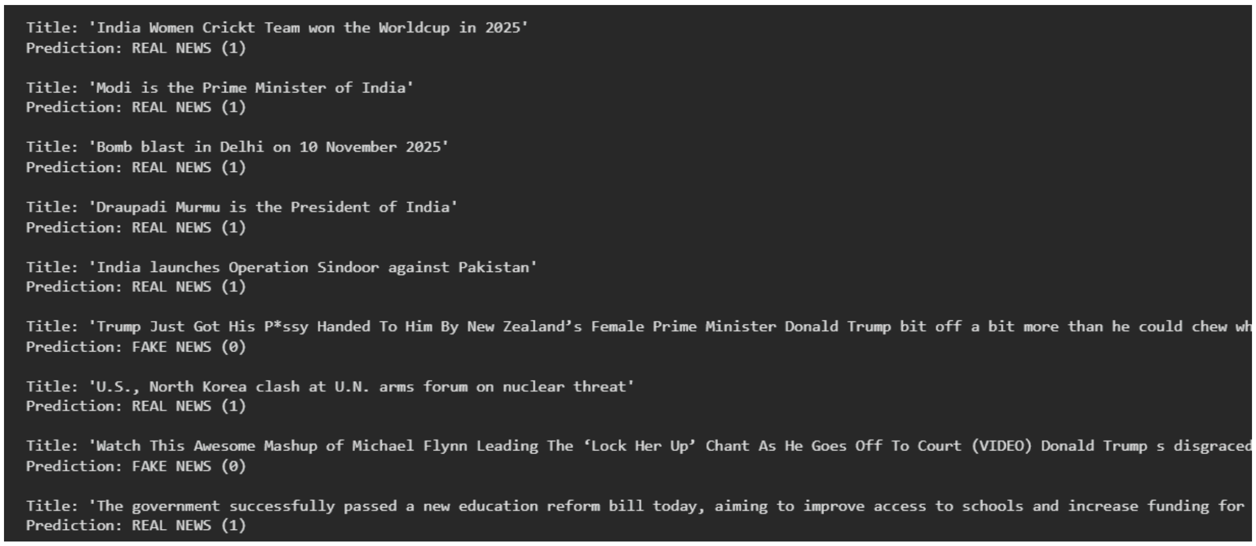
This demonstrates resilience to domain shift, writing style variation, and linguistic ambiguity.

Role of BERT (Limitations and Insights)

A fine-tuned BERT model was trained separately and produced contextually strong predictions on test samples.
However, BERT was not deployed in the Streamlit UI due to:

- high inference latency without GPU acceleration,
- increased memory footprint,
- Streamlit execution delays.

Despite this deployment constraint, BERT experiments validated the semantic complexity inherent in misinformation detection, and motivated future inclusion of optimized transformer variants.



```
Title: 'India Women Crickt Team won the Worldcup in 2025'
Prediction: REAL NEWS (1)

Title: 'Modi is the Prime Minister of India'
Prediction: REAL NEWS (1)

Title: 'Bomb blast in Delhi on 10 November 2025'
Prediction: REAL NEWS (1)

Title: 'Draupadi Murmu is the President of India'
Prediction: REAL NEWS (1)

Title: 'India launches Operation Sindoor against Pakistan'
Prediction: REAL NEWS (1)

Title: 'Trump Just Got His P*ssy Handed To Him By New Zealand's Female Prime Minister Donald Trump bit off a bit more than he could chew wh
Prediction: FAKE NEWS (0)

Title: 'U.S., North Korea clash at U.N. arms forum on nuclear threat'
Prediction: REAL NEWS (1)

Title: 'Watch This Awesome Mashup of Michael Flynn Leading The 'Lock Her Up' Chant As He Goes Off To Court (VIDEO) Donald Trump s disgraced
Prediction: FAKE NEWS (0)

Title: 'The government successfully passed a new education reform bill today, aiming to improve access to schools and increase funding for
Prediction: REAL NEWS (1)
```

Fig 13: Performance of BERT on sample test-cases

2. Behaviour of Misinformation Spread in Social Networks

The SNA component used a Barabási–Albert (BA) scale-free network, reflecting realistic social network characteristics, and simulated misinformation using an SIR-based propagation model.

Influence of High-Degree Nodes (Hubs)

Simulation results indicated that misinformation originating from hubs produced:

- immediate, explosive propagation,
- rapid infection of large connected components,
- dense red clusters around core nodes in early steps,
- steep infection-curve growth.

These observations validate network-science theory that hubs function as super-spreaders due to high connectivity.

Cross-Community Spread via Bridges

When misinformation originated from structurally weak or low-degree nodes (bridges), the spread exhibited:

- delayed early propagation,
- gradual infection of small pockets,
- eventual cross-cluster jumps leading to larger cascades.

This reflects real-world behaviour, where bridge nodes enable inter-community diffusion across otherwise disconnected social groups.

Effectiveness of Barabási–Albert Graph Structure

The BA graph provided:

- naturally emerging hubs,
- scale-free degree distribution,
- realistic clustering behaviour,
- long-range connections.

These structural features produced propagation patterns similar to documented social-media misinformation cascades.

System Performance Observations

- Real-time simulations performed reliably for up to ~500 nodes.

- Networks beyond 1000 nodes exhibited slower frame rendering and increased latency.
- Hub/bridge selection and dynamic SIR visualization were consistently handled by the UI.

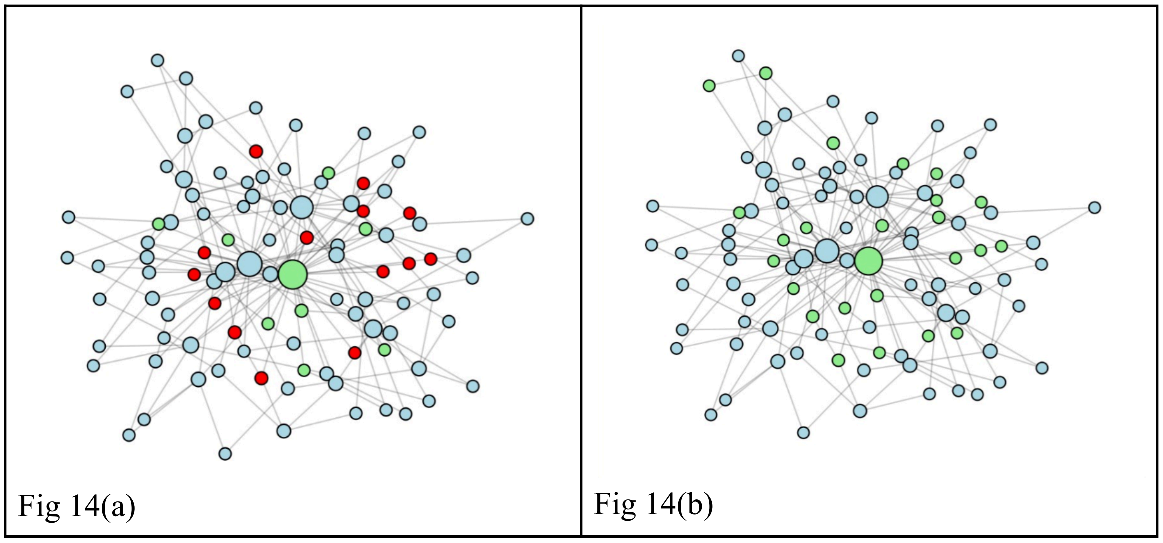


Fig 14: Network state updates from initial spread (a) to final state (b).

The figures above shows the dynamic network updation of states, Fig 14(a) being the initial state where misinformation began propagating and Fig 14(b) being the final state after the simulation is complete

6.2 CONCLUSION

This work presents a cohesive, end-to-end framework for misinformation analysis, integrating machine-learning–based classification with SNA-driven propagation modelling. The system successfully demonstrates both accurate detection and realistic simulation of misinformation spread.

I. Reliable Detection via Hybrid Ensemble Modelling

The Soft Ensemble emerged as the most effective classifier, achieving high accuracy, robust generalization, and well-balanced precision–recall characteristics. Its performance indicates suitability for practical misinformation screening and operational use.

II. Structural Role of Nodes Determines Spread Potential

The SNA simulations validated key network-science principles:

- **Hubs** act as super-spreaders, enabling rapid, localized amplification.
- **Bridges** serve as infiltration points, allowing misinformation to traverse community boundaries.

These results echo real misinformation events where influencers (hubs) and cross-community accounts (bridges) play pivotal roles.

III. SIR Model Offers a Realistic Approximation of Misinformation Flow

Mapping SIR states to social behaviours:
Susceptible (unaware) → Infected (sharing) → Recovered (non-sharing), produced realistic cascade and containment curves. The model captured both growth dynamics and natural decay of misinformation.

IV. Educational and Analytical Value of the Integrated System

The combined ML & SNA platform provides:

- visual understanding of misinformation diffusion,
- comparative impact of hubs vs. bridges,
- interactive simulation of infection curves,
- clear classification results with supporting metrics.

This makes the system effective for teaching, experimentation, and awareness activities.

V. Identified Limitation

The trained BERT model could not be integrated into the UI due to computational constraints. This limits deeper semantic interpretation during real-time classification and is addressed in future directions.

6.3 FUTURE WORK

Several extensions can significantly enhance accuracy, scalability, and real-world applicability.

I. Integration of Optimized Transformer Models

Adopting lightweight models such as DistilBERT, ALBERT, or ONNX-accelerated versions will allow deployment of transformer-level performance without compromising real-time responsiveness.

II. Incorporation of Real Social Media Network Data

Using real interaction graphs (Twitter API, Reddit, Facebook public data) will generate spread patterns that better match contemporary misinformation ecosystems.

III. Multi-Source and Coordinated Spread Simulations

Future versions should support:

- concurrent misinformation sources,
- botnet-like behaviour,
- coordinated influence operations.

This will improve ecological validity.

IV. Performance and Scalability Enhancements

To support larger networks:

- GPU-accelerated rendering,
- multiprocessing of SIR updates,
- caching of graph layouts,
- optimized graph libraries (e.g., Graph-Tool).

V. Exploration of Alternative Diffusion Models

Beyond SIR, additional models can be implemented, such as:

- SEIR
- Independent Cascade
- Linear Threshold
- SIHR

These models capture different behavioural and temporal characteristics.

VI. Real-Time Data Stream Integration

A major enhancement involves enabling live misinformation monitoring using:

- Twitter Streaming API
- Reddit live comments
- Fact-checking databases

Benefits include:

- continuous real-time detection,
- dynamic TF-IDF/embedding updates,
- evolving graph construction,
- automatic re-simulation upon new misinformation events.

This transforms the system into a proactive early-warning and analysis tool.

References

[1] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999. Available: https://www.science.org/doi/10.1126/science.286.5439.509

[2] M. E. J. Newman, "The Structure and Function of Complex Networks," *SIAM Rev.*, vol. 45, no. 2, pp. 167–256, 2003. Available: https://epubs.siam.org/doi/10.1137/S003614450342480

[3] R. Pastor-Satorras and A. Vespignani, "Epidemic Dynamics and Endemic States in Complex Networks," *Phys. Rev. E*, vol. 63, no. 6, 066117, 2001. Available: https://journals.aps.org/pre/abstract/10.1103/PhysRevE.63.066117

[4] C. Castellano, S. Fortunato, and V. Loreto, "Statistical Physics of Social Dynamics," *Rev. Mod. Phys.*, vol. 81, pp. 591–646, 2009. Available: https://journals.aps.org/rmp/abstract/10.1103/RevModPhys.81.591

[5] S. Boccaletti *et al.*, "Complex Networks: Structure and Dynamics," *Phys. Rep.*, vol. 424, no. 4–5, pp. 175–308, 2006. Available: https://www.sciencedirect.com/science/article/pii/S037015730500462X

[6] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani, "Epidemic Processes in Complex Networks," *Rev. Mod. Phys.*, vol. 87, no. 3, pp. 925–979, 2015.

[7] NetworkX Documentation, "NetworkX 3.0: Graph Analysis in Python," [Online]. Available: https://networkx.org/.

[8] GeeksforGeeks, "SIR Model for Disease Spread in Python," [Online]. Available: https://www.geeksforgeeks.org/sir-model/.

[9] P. K. Reddy, "WSDM 2018 Tutorial on Influence Maximization in Social Networks," *SlideShare*, 2018. Accessed: Nov. 2025. [Online]. Available: https://www.slideshare.net/slideshow/wsdm-2018-tutorial-on-influence-maximization-in-social-networks/101906797