Project Report

on

Fake News Detection with Dynamic Model Updates Based on Classifier Comparison

Submitted by

Project Members

Maitreyee Jadhav (1032211369)

Vinay More (1032211811)

Mansi Patil (1032221625)

Preshika Giri (1032221686)

Under the Internal Guidance of

Dr Yogita Hande

School of Computer Engineering and Technology
MIT World Peace University, Kothrud,
Pune 411 038, Maharashtra - India
2024-2025

Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

DEPARTMENT OF COMPUTER ENGINEERING AND TECHNOLOGY

C E R T I F I C A T E

This is to certify that, **Maitreyee Jadhav, Vinay More, Mansi Patil, Preshika Giri** of BTech.(Computer Science & Engineering- Cyber Security and Forensics) have completed their project titled **"*Fake News Detection with Dynamic Model Updates Based on Classifier Comparison*"** and have submitted this Capstone Project Report towards fulfillment of the requirement for the Degree-Bachelor of Computer Science & Engineering (BTech-CSE) for the academic year 2024-2025

[Dr Yogita Hande]
Project Guide
School of CET
MIT World Peace University, Pune

[Dr. Balaji M Patil]
Program Coordinator
School of CET
MIT World Peace University, Pune

Internal Examiner:_____

External Examiner:_____

Date: 6 May, 2025

Acknowledgement

We, the members of our project group, would like to express our sincere gratitude to all those who supported and guided us throughout the successful completion of our capstone project.

First and foremost, we are extremely thankful to our project guide, Dr. Yogita Hande, for her invaluable support, constant encouragement and expert guidance during the project.

We would also like to thank Dr. Balaji Patil, Head of the Computer Science and Engineering Department, for providing us with the opportunity to undertake this project and for the facilities offered by the department.

We are also grateful to our fellow classmates and friends who supported us, directly or indirectly, during various stages of the project.

Lastly, we express our heartfelt appreciation to our families for their patience, motivation, and encouragement throughout the journey.

Group Members:

Maitreyee Jadhav (1032211369)

Vinay More (1032211811)

Mansi Patil (1032221625)

Preshika Giri (1032221686)

Abstract

In the digital age, the rapid dissemination of information through online platforms has led to a significant rise in the spread of fake news, posing serious threats to societal trust, political stability and public safety. This project focuses on the development of a machine learning-based system for automatic fake news detection. The system analyzes textual news content to classify it as real or fake using various classification algorithms. Initially, data preprocessing steps such as tokenization, stopword removal, and stemming were applied to clean the dataset. Features were extracted using techniques like TF-IDF and Count Vectorizer. Multiple models including Logistic Regression, Support Vector Machine (SVM), and Naive Bayes, were trained and evaluated for performance using metrics such as accuracy, precision, recall, and F1-score. The model with the highest accuracy is selected for deployment. To ensure the system remains effective over time, a dynamic model updating strategy is implemented, wherein the model is periodically retrained with newly labeled data. This approach not only enhances prediction accuracy but also adapts to evolving patterns in misinformation. The project demonstrates an adaptive solution for combating fake news using data-driven techniques.

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Project Statement

The rise of digital platforms has led to an increase in the spread of fake news, with existing detection systems relying on static models that quickly become outdated. These models struggle to keep up with evolving misinformation styles, reducing accuracy. There is a need for a dynamic system that continuously retrains fake news detection models using fresh data to maintain their effectiveness and relevance over time.

1.2 Area

This project falls under the domain of Machine Learning and Natural Language Processing (NLP), specifically in the sub-area of text classification and dynamic model training. It combines ML techniques with web development to detect misinformation in news content.

1.3 Project Introduction and Aim

Fake news presents a significant threat to societal stability by spreading false or misleading information. Static models, though effective initially, quickly become outdated as new types and formats of fake news emerge. This creates a pressing need for a system that can adapt dynamically to the changing landscape of digital misinformation.

The aim of this project is to develop a dynamic fake news detection system that periodically updates its model using newly collected real and fake news articles. This approach ensures that the model remains current and effective in identifying evolving patterns of misinformation. The project seeks to address the major limitations of static models by introducing a retraining mechanism that keeps the model aligned with recent data trends.

Implementation overview :

- Data Collection: Fetching new real and fake news articles at regular intervals from news websites.
- Data Preprocessing: Cleaning and preparing text data using NLP techniques.
- Model Training and Evaluation: Using ML algorithms (e.g., Logistic Regression, SVM, Naive Bayes) to build classification models.

- Dynamic Updating: Periodically retraining the model to maintain performance as new data becomes available.
- Developing and Integrating APIs for retraining the model and predicting the authenticity, with frontend.

Application of the project is the dynamic fake news detection system can be applied in several areas-

Social Media Platforms: To monitor and flag misleading news in real time.

Government and NGOs: To combat misinformation campaigns and promote digital literacy.

Academic and Research Tools: For studying trends and evolution in misinformation patterns.

By integrating dynamic retraining into fake news detection, the project enhances the robustness and adaptability of misinformation filtering systems, making them more effective in real-world environments.

Chapter 2

Literature Survey

Fake news has become a growing concern in the digital age, especially with the widespread use of social media platforms where misinformation can spread rapidly, influencing public opinion and causing real-world consequences. Researchers have explored various methods to tackle this issue, with machine learning and natural language processing emerging as effective tools for fake news detection.

Logistic Regression has been widely used due to its effectiveness in binary classification tasks such as distinguishing between real and fake news. SVM is particularly well-suited for high-dimensional text data, offering robust performance even with limited samples. Naive Bayes, known for its simplicity and speed, remains a strong baseline model in text classification.

Based on these findings, we selected Logistic Regression, SVM, and Naive Bayes for our project to evaluate their performance and suitability for fake news detection.
The following table highlights key referenced research papers that have addressed fake news detection using various machine learning and natural language processing approaches. It provides insights into the models used, datasets, and methodologies, helping us understand existing solutions and justify our choice of models:

Table 1. Literature Review

Ref	Title	Publication Year	Methodology	Positive points of publication	Gaps in publication work
1.	Detecting and Mitigating the Dissemination of Fake News: Challenges and Future Research Opportunities – Wajiha Shahid, Bahman Jamshidi, Saqib Hakak, Haruna Isah, Wazir Zada Khan, Muhammad Khurram Khan, Kim-Kwang Raymond Choo (Journal)	2024	GloVe embeddings with a CNN model (FNDNet) for fake news detection, a CNN-LSTM model for stance detection, and applies feature extraction with PCA and Chi-Square for dimensionality reduction.	The most common and critical challenges encountered by other researchers to make an efficient classifier. It addressed solutions to those challenges.	Lacks real-time implementation like the authors could have implemented any one proposed model for clear understanding.
2.	Enhancing Fake News Detection by Multi-feature classification – Ahmed Hashim Jawad Almarashy, Mohammad-Reza Feizi-Derakhshi, Pedram Salehpour (Journal)	2023	TF-IDF (global features), CNN (spatial features), and BiLSTM (temporal features) using early fusion, and classifies the result with a Fast Learning Network (FLN) for fake news detection.	A model that integrates global, spatial and temporal features of text for enhancing the accuracy. The hybrid approach accurately classifies fake news, even when dealing with datasets having large number of training samples	The model is tested only on English Language dataset.

3.	Real-Time Fake News Detection Using Big DataAnalytics and Deep Neural NetworkMuhammad Babar,Awais Ahmad, Member, IEEE,Muhammad Usman Tariq, and Sarah Kaleem	2023	A hybrid N-gram and LSTM model for fake news detection, deployed on a big data platform for parallel and distributed processing to achieve real-time, accurate classification.	Combined DNNs and blockchain for a highly accurate and secure fake news detection system, ensuring source credibility and traceability Built a real-time detection system using big data analytics, enabling fast, Large-scale news analysis with minimal delays.	The model does not focus on language nuances such as sarcasm, satire, and ambiguous phrasing, leading to misclassification.
4.	Fake News Detection Using Enhanced BERT Shadi A. Aljawarneh	2022	Fine-tunes the BERT model on a fake and real news dataset to enhance its detection capabilities, achieving 99.96% accuracy and outperforming other models.	The optimized BERT model improves language understanding, reducing errors from misleading or satirical headlines. Enhanced BERT outperforms traditional models, boosting accuracy and feature extraction for better fake news detection.	No addressing of the high computational cost of BERT,making real-time large-scale detection difficult.

5.	Fake News in Virtual Community, Virtual Society, and Metaverse: A Survey Jinxia Wang , Stanislav Makowski, Alan Cieślik, Haibin Lv , and Zhihan Lv	2023	A survey on fake news in virtual communities and the metaverse, analyzing its manifestation in single-modal and multimodal forms, and reviewing detection methods. It also discusses future directions for intelligent detection and information security in these environments.	Fake news challenges in emerging digital spaces, highlighting new threats in virtual societies and the metaverse. A comprehensive comparison of existing fake news detection techniques, identifying key gaps and future research directions.	A broad survey of fake news in virtual environments but lacks practical implementation strategies for real-time detection and mitigation. Solution: Developing AI-driven real-time monitoring systems tailored for virtual communities and the metaverse can help detect and counter fake news more effectively.

Chapter 3
Problem Statement

The rise of digital platforms has led to an increase in the spread of fake news, with existing detection systems relying on static models that quickly become outdated. These models struggle to keep up with evolving misinformation styles, reducing accuracy. There is a need for a dynamic system that continuously retrains fake news detection models using fresh data to maintain their effectiveness and relevance over time.

3.1 Project Scope

This project focuses on building a machine learning-based fake news detection system that not only classifies news as real or fake but also includes a mechanism for dynamic model updating. The scope includes data collection, preprocessing, training initial models, evaluating performance, and implementing an retraining pipeline. It primarily targets English-language news articles from online sources. The final system is intended to serve as a proof of concept for real-world deployment in platforms such as news aggregators, fact-checking websites, and social media monitoring tools.

3.2 Project Assumptions

- The input news data (headlines or full articles) is in English and consists of textual content only.
- The datasets used for training and retraining are labeled accurately as real or fake.
- The news scrapped from legitimate news sources is real.
- The collected news data for dynamic updates will follow a similar structure to the original training data.
- Users of the system will input well-formed and grammatically correct news text.

3.3 Project Limitations

The system may not perform well on languages other than English due to language-specific preprocessing and training limitations. The accuracy of the system depends heavily on the quality and balance of the labeled training data. Sarcastic or satirical content may be misclassified due to the lack of semantic understanding by traditional machine learning models.

The project does not address image- or video-based fake news, only textual data. Real-time data collection for dynamic retraining may be limited by rate limits or access restrictions of third-party sources.

3.4 Project Objectives

1. Designing and developing a dynamic fake news detection system that updates periodically using new news data with comparative analysis if ML models to find the best classifier.
2. Improve predictive performance against evolving misinformation patterns.
3. Bridge the gap between static detection methods and real-time, adaptive solutions.

Chapter 4
Project Requirements

4.1 Resources

4.1.1 Human Resources

Project Guide - To provide technical guidance and oversee project progress.

Project Team Members - Responsible for data collection, preprocessing, model training, evaluation, UI creation, API development and integration, and report documentation.

4.1.2 Reusable Software Components

Text Preprocessing Scripts- Reusable Python scripts/functions for cleaning, tokenization, stopword removal, stemming/lemmatization, etc.

Vectorization Module- TF-IDF/CountVectorizer reusable across multiple experiments.

Model Training Functions- Training pipeline for models like Logistic Regression, SVM, and Naive Bayes can be reused across different datasets.

Evaluation Module- Common performance metrics such as accuracy, precision, recall, F1-score.

Retraining Module- Used to retrain models with new data.

4.1.3 Hardware Requirements:

Operating System: Windows/Linux/macOS

RAM: 8GB

Storage: 2GB of free disk space

4.1.4 Software Requirements:

Programming Language:

Python: Widely used for machine learning and natural language processing tasks.

- Integrated Development Environment (IDE): Any IDE for efficient code development and debugging. Examples: PyCharm, Jupyter Notebooks, Visual Studio Code.

Machine Learning Libraries:

- Scikit-learn: For machine learning algorithms and tools.
- TensorFlow or PyTorch: Deep learning frameworks for neural network-based mode.

Web Scraping Tools (for data collection):

- Beautiful Soup: Python library for web scraping.
- Scrapy: Open-source web crawling framework.

Version Control:-

- Git: Popular version control system.

4.2 Requirements Rationale

Table 2. Requirements Rationale

Requirement	**Rationale**
Dynamic model retraining	To adapt to emerging fake news patterns and improve detection accuracy
Preprocessing of text data	Ensures model receives clean, uniform input for accurate learning
Use of TF-IDF or similar vectorizer	Converts text into numerical format usable by ML models
Performance evaluation metrics	To assess model quality and compare different approaches
GUI/web interface	Allows end users to interact with the system easily
Periodic data collection web scraping	To keep the training data up-to-date for dynamic updates

4.3 Risk Management

Table 3. Risk Management

Risk Factor	**Impact Level**	**Mitigation Strategy**
Inaccurate or biased dataset	High	Use reliable data
Internet/data source access issues	Medium	Use multiple sources and cache existing data
Model performance degradation	High	Implement dynamic model updating

4.4 Functional Specifications

4.4.1 Interfaces

The project architecture comprises multiple interfaces that enable communication between components, facilitate data exchange, and support user interactions.

4.4.1.1 External Interfaces:

The system exposes two primary external interfaces through backend APIs. The Flask backend provides the /predict API for fake news prediction and the /retrain API for model retraining. The Node.js backend exposes an API endpoint to serve scraped BBC news articles to the frontend. Both external interfaces accept HTTP requests from the frontend, process the data, and return structured responses in JSON format.

4.4.1.2 Internal Interfaces:

Internally, the system manages interfaces between various modules such as the web scraper and the dataset management module. The web scraper passes cleaned news article data to MongoDB and the JSON dataset, which is then consumed by the retraining module. Additionally, the HuggingFace-based fake news generation module interfaces with the dataset management component to append synthetic data into the training dataset.

4.4.1.3 Communication Interfaces:

Communication between the frontend and backends is achieved via RESTful APIs over HTTPS. API requests and responses follow a standardized JSON structure to ensure smooth data exchange. The frontend communicates with both Flask and Node.js backends securely through these APIs, leveraging modern HTTP methods such as POST (for prediction and retraining) and GET (for fetching news articles).

4.4.1.4 Graphical User Interfaces (GUI)

The frontend graphical user interface, developed using React.js and Tailwind CSS, presents a clean and intuitive design for users. It consists of an input form for entering news articles, a "Predict" button to submit data, and dynamically updated sections displaying prediction results and scraped BBC articles. The layout is responsive, ensuring compatibility with both desktop and mobile devices.

4.4.2 Interactions

End users interact with the system primarily through the web application interface. Users input a news article or headline into the text form on the frontend and click the "Predict" button. This triggers an API request to the Flask backend, which returns a prediction result (Real or Fake). The result is displayed instantly below the input form.

In parallel, users can view a list of recently scraped BBC articles, which are displayed in card format by fetching data from the Node.js backend API. These cards dynamically update to reflect the latest available data. The system ensures smooth interactions between user actions, API requests, and result displays, providing a seamless user experience.

4.4.2.1 Sustainability

The project architecture is designed with long-term sustainability in mind. The use of cloud platforms (Vercel and Render) ensures scalable hosting without the need for manual server maintenance. The modular design, with clearly separated frontend, scraping, dataset management, and prediction modules, allows individual components to be updated, improved, or replaced independently without disrupting the entire system. The retraining mechanism supports continuous model improvement, ensuring

the system adapts to evolving data over time.

4.4.2.2 Quality Management

Quality assurance was integrated throughout the development process using best practices such as modular coding, API testing, and dataset validation. The system underwent rigorous functional and integration testing to ensure accuracy, reliability, and usability. Code quality was maintained through version control (GitHub), code reviews, and use of linters and formatters in both frontend and backend codebases. Comprehensive documentation supports maintainability and future enhancements.

4.4.2.3 Security

To safeguard the system, multiple security measures were implemented. HTTPS communication encrypts data transmission between frontend and backends. CORS policies restrict API access to authorized frontend origins only. Cloud platforms (Vercel and Render) offer built-in security features including HTTPS, resource isolation, and automated server management. Together, these measures ensure the integrity, confidentiality, and secure operation of the application.

Chapter 5
Proposed Architecture

5.1 Design Consideration

Use of Lightweight, Interpretable Models- The system employs lightweight machine learning models such as Logistic Regression and Naive Bayes for text classification. These models are not only computationally efficient, allowing for faster training and inference, but are also highly interpretable. Their simplicity aids in understanding decision boundaries and contributes to ease of debugging and transparency an essential factor in applications where understanding the rationale behind a prediction is important. Furthermore, these models are well-suited for deployment in environments with limited computational resources.

Modular Design for Future Model Upgrades- The architecture follows a modular design approach, separating components such as data preprocessing, model training, evaluation, and user interface (UI). This allows for easy replacement or upgrade of individual modules without affecting the entire system. For instance, while the current implementation uses traditional machine learning models, the system can be enhanced in the future to incorporate deep learning models (e.g., LSTM, BERT) by replacing only the model component, keeping the rest of the pipeline intact.

Maintainability and Clean Code Architecture- The codebase is structured following best practices for maintainability. Clear separation of concerns is maintained across the following components:

- Data Collection: Scripts for fetching or ingesting news articles.
- Preprocessing: Tokenization, stopword removal, and text normalization.
- Model: Training, validation, and inference logic.
- User Interface (UI): Interaction layer using web technologies (e.g., Flask). This separation facilitates debugging, testing, and collaborative development, enabling easy handoffs and long-term maintenance.

5.2 Assumption and Dependencies

Text-Based, English Language News Data- It is assumed that the input news articles will be predominantly textual and written in the English language. Consequently, the preprocessing pipeline and feature extraction techniques (e.g., TF-IDF, bag-of-words) are tailored for English text, leveraging tools like NLTK which are optimized for English corpora.

Balanced and Representative Labeled Datasets- The training process assumes access to labeled datasets that are both balanced (i.e., roughly equal proportions of fake and real news) and representative of the kinds of news articles the system will encounter in deployment. Any significant bias or imbalance in the training data may affect model generalization and performance.

Availability of Required Python Libraries

- The system depends on several widely-used Python libraries, including:
- scikit-learn (for model building and evaluation)
- Flask (for API and web app deployment)
- pandas (for data handling and manipulation)
- NLTK (for natural language preprocessing)

It is assumed that these libraries will be installed and accessible in the runtime environment, either through package managers like pip or pre-configured virtual environments or containers.

5.3 General Constraints

No Multimedia Processing (Images or Videos)- The current project scope is limited strictly to textual data. Misinformation through multimedia formats (such as manipulated images, deepfake videos, or memes) is beyond the current system's capabilities. Processing such content would require significantly different technologies (e.g., computer vision, deep learning models like CNNs or transformers for video analysis), which are out of scope for this text-focused project.

Only Text-Based News Content is Considered- The system exclusively deals with textual news content-either entered manually by a user or extracted from datasets/APIs. It assumes that the text is in a human-readable format (such as a news headline or article paragraph). Other formats like audio news, scanned documents, or unstructured social media posts with mixed content (text and emojis/images) are not supported.

5.4 System Architecture

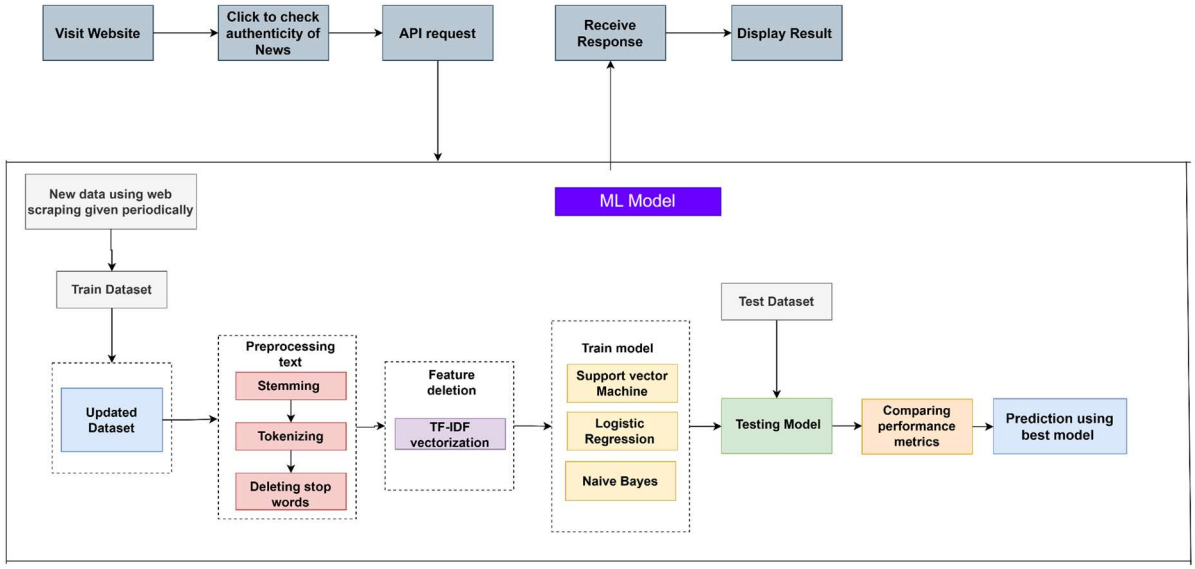

Fig. No. 1

The system architecture depicted in Fig. No. 1 outlines the workflow of a machine learning-based fake news detection system. Users interact with the system through a website, where they can submit news content to check its authenticity. The input is sent via an API request, processed by a machine learning model, and the result is displayed back to the user. In the backend, news data is periodically collected using web scraping and preprocessed through tokenization, stemming, and stop word removal. Features are then extracted using TF-IDF vectorization, and irrelevant features are removed. The system trains multiple models, Support Vector Machine, Logistic Regression, and Naive Bayes, using the training dataset. These models are evaluated using a test dataset by comparing performance metrics, and the best-performing model is selected to generate predictions for future inputs.

5.5 Modules of the Project

The fake news detection system is divided into several functional modules that work together to classify news content as real or fake. Each module is responsible for a specific part of the process, from data handling to prediction and visualization. The major modules are as follows:

1. Data Collection Module: This module gathers news data from reliable datasets or online sources. It includes both fake and real news articles and stores them in a structured format suitable for further processing.
2. Data Preprocessing Module: Raw data often contains noise such as punctuation,

stopwords, HTML tags, and special characters. This module performs cleaning operations, tokenization, stemming/lemmatization, and transformation (e.g., lowercasing) to prepare the data for analysis.

3. Feature Extraction Module: Textual data needs to be converted into numerical format for machine learning. This module uses techniques like TF-IDF (Term Frequency-Inverse Document Frequency), Bag of Words, or Word Embeddings to extract meaningful features from the news text.
4. Model Training and Evaluation Module: This module involves training machine learning models such as Logistic Regression, Support Vector Machine (SVM), and Naive Bayes using the extracted features. It also evaluates model performance using metrics like accuracy, precision, recall, and F1-score.
5. Prediction Module: Once trained, the model is used to classify new, unseen news articles. The prediction module takes user input or test data and outputs whether the news is real or fake.
6. User Interface: The system includes a front-end or graphical interface, this module allows users to input news text and view the prediction result. It enhances accessibility and usability for non-technical users.

5.6 Low level Design

The fig no. 2 illustrates the detailed flow of interactions between the system components involved in processing user input for fake news detection. The process begins with the User Input, where the user submits news content via a Website interface. This input is then forwarded as an API Request to the backend, where it is handled by the ML Model. Once the prediction is made, the Model Response is sent back through the API to the Website, which then displays the result to the user. This low-level design highlights the end-to-end communication path from user interaction to model prediction and response delivery.

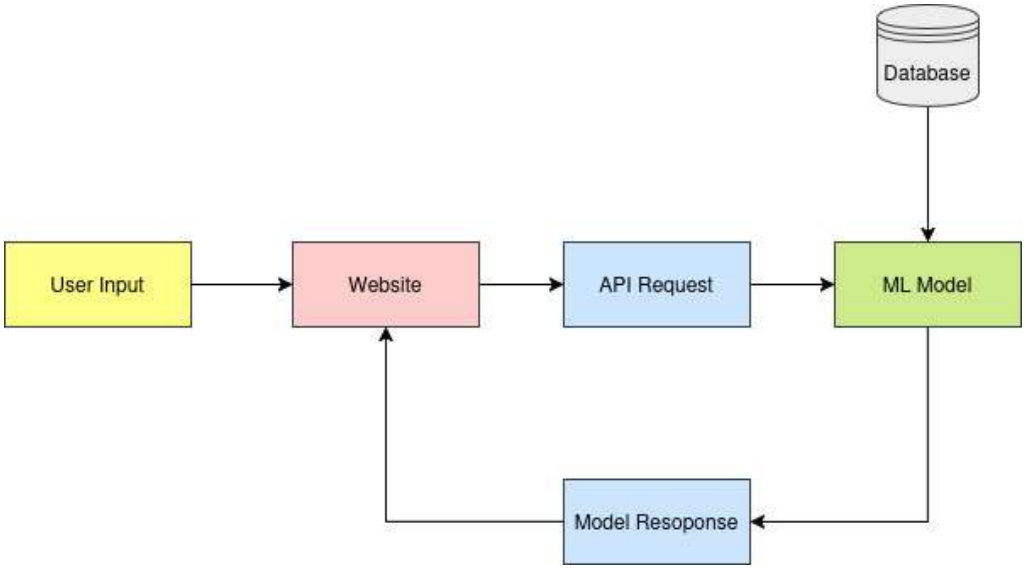

Fig No. 2

5.7 UML Diagrams

5.7.1 Use Case Diagram

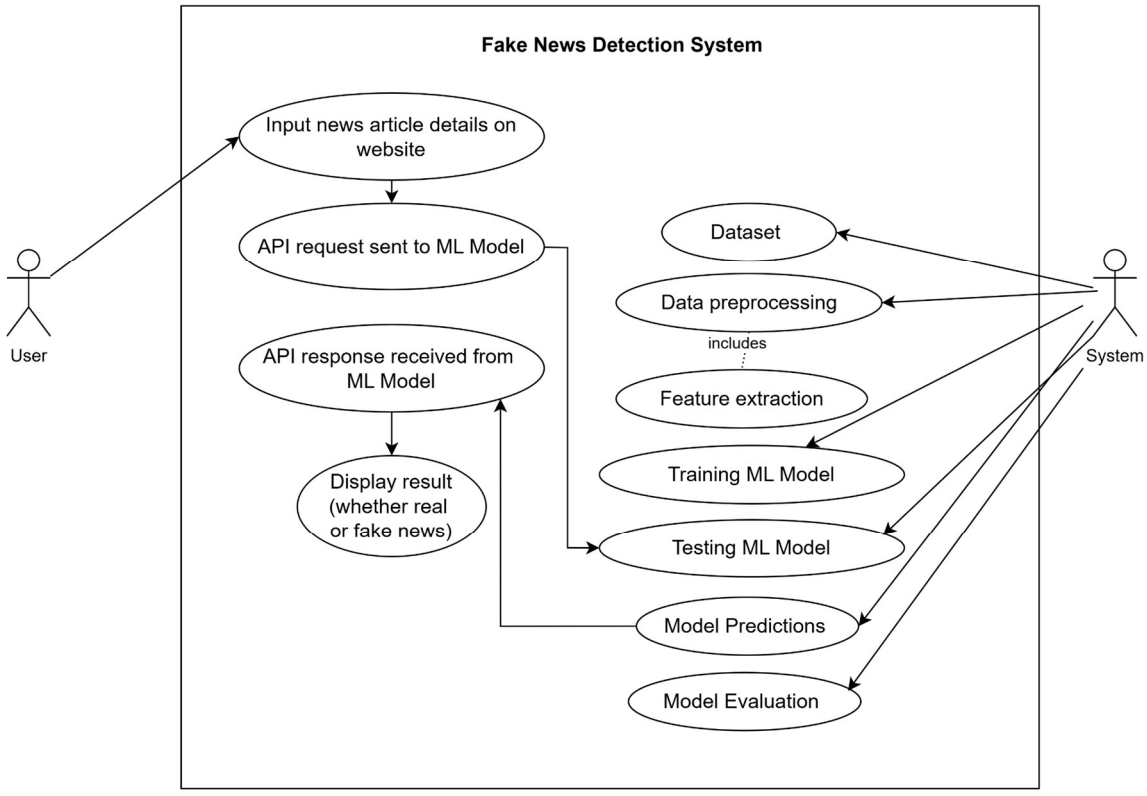

Fig. No. 3

5.7.2 Activity Diagram

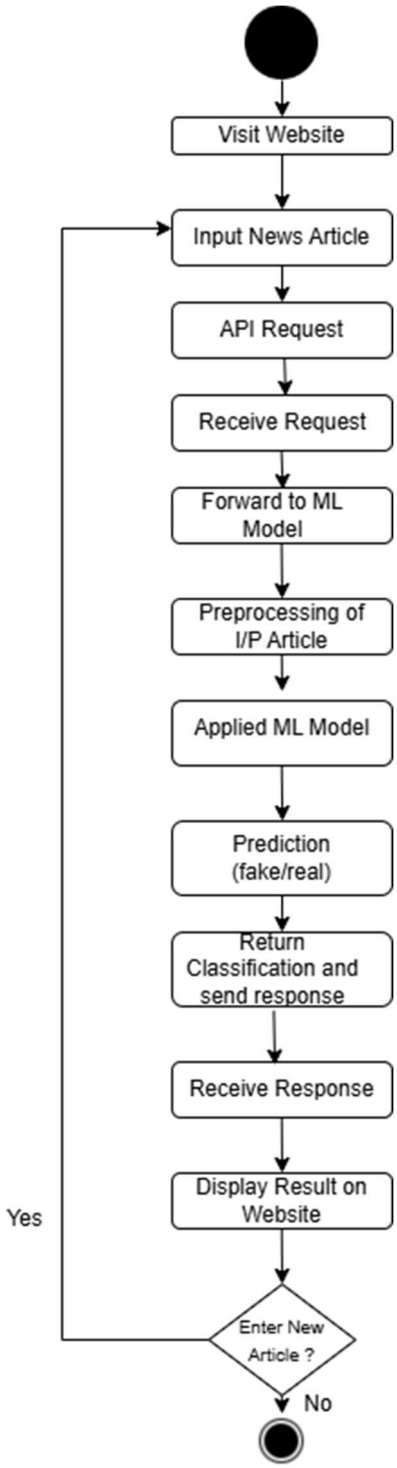

Fig. No. 4

Chapter 6
Project Plan

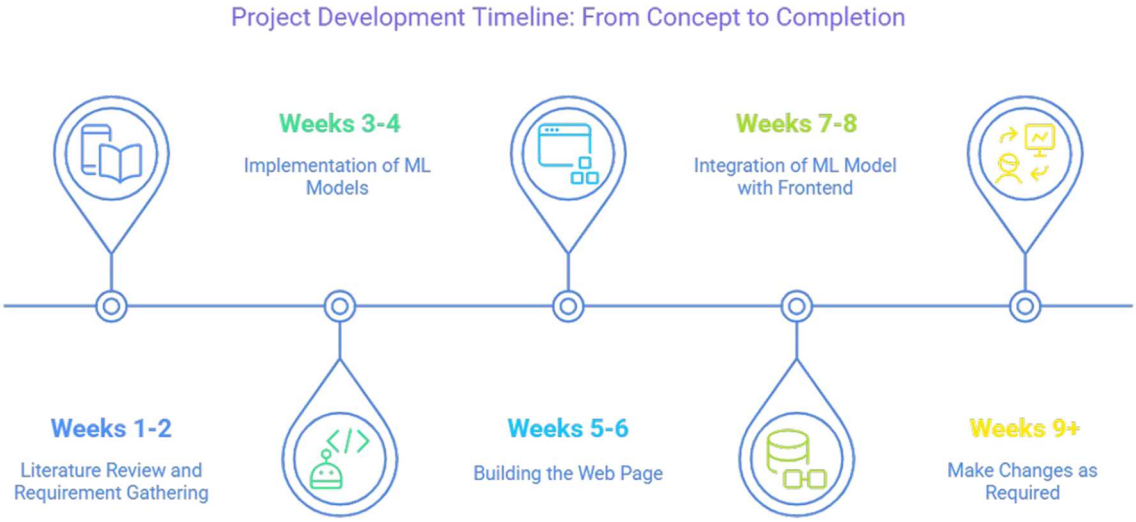

Fig. No. 5

The project timeline diagram titled "Project Development Timeline: From Concept to Completion" outlines a structured weekly roadmap for completing a development project, particularly one involving machine learning (ML) and web development.

In Weeks 1-2, the project begins with Literature Review and Requirement Gathering. This phase focuses on researching existing solutions, understanding the problem domain, and gathering technical and functional requirements. It lays the groundwork for the rest of the project by identifying what needs to be built and the constraints to consider.

Moving into Weeks 3-4, the team starts the Implementation of Machine Learning Models. This stage involves selecting appropriate ML algorithms, preprocessing data, training models, and evaluating their performance. This is a core technical phase where the intelligent backbone of the project is developed.

During Weeks 5-6, attention shifts to Building the Web Page. Here, developers focus on designing and developing the front-end and the back-end of the web application that will host or interact with the ML models. This stage emphasizes user experience and functionality.

In Weeks 7-8, the focus is on Integration of ML Model with Frontend. This phase bridges the gap between the machine learning model and the user interface, ensuring that the model's

predictions or functions are seamlessly available through the web platform.

Finally, from Week 9 onwards, the project enters a continuous improvement phase "Make Changes as Required". This involves fixing bugs, optimizing performance, and possibly adding new features. It's an iterative phase that ensures the product remains functional, usable, and relevant.

Chapter 7
Implementation

The project employs a systematic methodology that includes collection of data, preprocessing, training models, evaluation and taking the best model into consideration for dynamic model updating. We are addressing the noted research gaps of dynamic model updating in the methodology. The proposed system architecture is shown in Fig. 1.

7.1 Data Collection

The news data for model training was collected in the following ways:

Real news was collected from legitimate news website. Details like article title, article description, date of scrapping the news were collected through web scraping and stored in a json file. The real news is labelled as ‘0’ in the dataset.

Fake news was collected through two ways, first, some websites which had a section for some identified fake news articles for awareness, second, using a GenAI model to paraphrase some real news articles and convert into fake version of it. Details like article title, description, date (refers to the date on which article was scrapped or created using GenAI model), label, are stored in a json format.

Table 4. Dataset Features

Sr no.	Feature name	Description
1	title	Title if the news article
2	text	Description if the news article
3	label	Label of the news article; fake: 1 and real: 0
4	date	Date on which the article was web scrapped

7.2 Preprocessing:

The combined CSV file is used for preprocessing. A preprocessing function is developed where ‘text’ is given as input. The preprocess_text function is designed to clean and normalize raw text data for natural language processing tasks. It begins by removing all non-alphabetic characters using regular expressions, then converts the text to lowercase and splits it into individual words. Common stopwords (e.g., "the", "is", "and") are filtered out to reduce noise. Each remaining word is then stemmed using the Porter Stemmer to reduce it to its root form (e.g., "running" becomes "run"). Finally, the processed words are joined back into a single string. This preprocessing helps in reducing vocabulary size and improving the performance of machine learning models by focusing only on the meaningful content of the text.

7.3 Feature extraction

Feature extraction from textual data can be accomplished by using word embeddings or TF-IDF vectorization (Term Frequency-Inverse Document Frequency). By converting text into numerical attributes, we can determine the semantic content and word importance of each document. In our implementation, we utilized the TfidfVectorizer from the scikit-learn library, configuring it with specific parameters to enhance feature extraction:

- N-gram Range: Set to (1, 2) to capture both unigrams and bigrams, allowing the model to consider individual words and pairs of consecutive words, thereby incorporating some contextual information.
- Maximum Document Frequency (max_df): Set to 0.9 to exclude terms that appear in more than 90% of the documents, as such terms are likely to be non-informative and may not contribute to distinguishing between documents.
- Minimum Document Frequency (min_df): Set to 2 to ignore terms that appear in fewer than two documents, reducing the impact of rare terms that may not be relevant for the analysis.

The vectorizer was fitted to the corpus and the textual data was transformed into a TF-IDF weighted term-document matrix, which served as the input for subsequent analytical processes.

7.4 Model Training

The dataset is split into training and testing sets using the train_test_split function. By setting test_size, the data can be divided into training and testing and random_state is used to ensure the split is reproducible. Then we have used three ML Models, Support Vector Machine (SVM), Logistic Regression (LR) and Naive Bayes(NB) for training. The models are then trained on the training data using model.fit().

Support Vector Machine (SVM) is an effective classification method that searches for the best hyperplane in a multidimensional space to separate different classes. It is robust against outliers and efficiently handles both linear and non-linear decision boundaries through the use of kernel functions. By maximizing the margin between classes, SVM improves generalization on unseen data. Its adaptability to different data types and the ability to control overfitting through regularization parameters make SVM a popular choice for various classification tasks.

Logistic Regression is a widely used statistical method for binary and multiclass classification problems. It models the probability of a data instance belonging to a particular class using a logistic (sigmoid) function. The model estimates the relationship between input features and the log-odds of the outcome, making it both interpretable and computationally efficient. Logistic Regression performs best when the classes are linearly separable and includes regularization techniques to prevent overfitting, allowing it to generalize well to new data. Naive Bayes is a simple yet powerful probabilistic classification algorithm based on Bayes' Theorem. It assumes that the features are conditionally independent given the class label, which simplifies the computation significantly. Despite its 'naive' assumption, Naive Bayes often performs competitively, particularly in text classification and spam detection tasks. It is highly efficient, requires a small amount of training data, and works well even when the independence assumption is somewhat violated in practice. After training these models, we calculate the evaluation metrics such as accuracy, precision, recall and f1- score. The model with best accuracy is selected and saved in a .pkl file using joblib library in Python as the final model for predicting fake news.

7.5 Prediction

Flask API endpoint (/predict) is created that allows users to make predictions using the

pre-trained ML models. When a GET request is sent with a query parameter text, the API first loads the saved model and the associated TF-IDF vectorizer using joblib. The input text is then preprocessed using the same preprocessing logic used during training and converted into a numerical vector using the loaded vectorizer. This vector is passed to the ML model to generate a prediction. If no input text is provided, the API returns an error message; otherwise, it responds with the predicted label in JSON format.

7.6 Dynamic Model Updating

Dynamic Model Updating is the ability of a deployed machine learning system to periodically or continuously incorporate new data into its learning process. Instead of relying solely on a static model trained once on historical data, dynamic updating enables the model to retrain or fine-tune itself as new labeled data, thereby improving its accuracy and adaptability over time. This is achieved by retraining the model on a schedule (e.g., daily or weekly). We are scraping the real news data daily and retraining the model on weekly basis. The updated model is then saved and replaces the old version, ensuring that the prediction system always uses the most relevant and recent insights. This approach is essential in today's world where data patterns evolve quickly to ensure reliable predictions.

7.7 Implementation of Project UI

Designed a clean, intuitive user interface (UI) using React.js and Tailwind CSS. Developed the NewsHub platform with: Fake News Detector component (Text input form for prediction), Top News Display Cards (Showcasing latest BBC news articles), and a responsive and mobile-friendly layout for user accessibility. Implemented interactive UI elements: "Predict" button to classify input news as Real or Fake, clear visual feedback with result display (Prediction: Real/Fake), and a navigation bar for seamless movement between sections (Breaking, Business, Innovation, etc.).

7.8 API Integration

Integrated Flask Backend APIs for dynamic interaction:

- /predict API: Sends user input from frontend → receives and displays prediction result

- /retrain API: Triggers model retraining when new dataset (scraped/generated articles) is appended

Integrated Node.js Backend API to fetch and display scraped BBC news articles dynamically on frontend cards. Ensured secure API communication between frontend (Vercel) and backends (Render) using HTTPS. Implemented error handling and status feedback to manage API request failures gracefully

Verified seamless real-time interaction:

- User input → API request → Prediction response → Display in UI
- Scraped news data → Node.js API → Frontend news cards

Chapter 8
Performance Evaluation and Testing

8.1 Performance Evaluation (Time Complexity Analysis)

The project involves multiple stages, including text preprocessing, vectorization using TF-IDF, and classification using machine learning models. The time complexity of each major component is analyzed below:

1. Text Preprocessing

- Each input text undergoes regular expression cleaning, tokenization, stopword removal, and stemming:
- Time Complexity: O(n) per document, where n is the number of characters in the text.
- Reason: All operations (regex, tokenization, stopword check, and stemming) iterate linearly over the text.

2. TF-IDF Vectorization

- The TfidfVectorizer is used with n-grams (unigrams and bigrams), along with frequency-based filtering:
- Time Complexity (fit and transform): O(N * L)
    - N = number of documents
    - L = average number of unique tokens/ngrams per document (after preprocessing)
- Explanation: The vectorizer scans all documents and computes term frequencies and inverse document frequencies.

3. Model Training

- The SVM model with a linear kernel is trained on the transformed feature matrix:
- Time Complexity (SVM with linear kernel): O(n * d) to $O(n^2 * d)$ depending on implementation and data separability
    - n = number of training samples
    - d = number of features (ngrams)
- Explanation: Linear SVMs are faster than kernel-based ones, but the cost can grow quadratically with the number of samples in worst-case scenarios.

Alternate models:

- Logistic Regression: O(n * d) (gradient descent-based training)
- Multinomial Naive Bayes: O(n * d) (simple probabilistic training)

4. Model Inference

   For prediction (inference) on test data:

   - Vectorization (transform): O(M * L) where M is the number of test samples
   - Prediction (SVM): O(M * d) - dot product with learned hyperplane

   Overall: Linear with respect to test data size and feature count

8.2 Testing:

To ensure the reliability, accuracy, and robustness of the fake news detection system, a comprehensive testing strategy was employed throughout the development lifecycle. The testing process was designed to validate individual components, their interactions, and the overall system performance under various conditions. Both manual and automated testing approaches were used to assess the functional correctness, data integrity, user experience, and security aspects of the project.

Functional Testing was performed on the frontend application to verify that all user interface components worked as intended. The Fake News Detector form was tested to ensure that users could enter news text seamlessly and submit it for prediction. The display of prediction results ("Real" or "Fake") was validated to confirm that the feedback appeared clearly and accurately after every prediction. Additionally, the frontend news cards displaying BBC articles were tested to ensure proper rendering of titles, descriptions, and images. The responsiveness and mobile compatibility of the frontend were also verified to guarantee a smooth user experience across different devices and screen sizes.

Integration Testing was carried out to validate the communication between the frontend and backend services. The connection between the React.js frontend and the Flask backend APIs (/predict, /retrain) was tested to ensure that user inputs were correctly transmitted, processed, and the appropriate results returned. Similarly, the integration between the frontend and the Node.js backend was validated to confirm that scraped BBC articles were correctly fetched and displayed on the frontend. Special attention was given to verifying that all API requests and responses were handled gracefully, even under slow network conditions or temporary backend downtimes.

Dataset Testing focused on the accuracy and integrity of data collected and managed by the web scraping and dataset management components. The BeautifulSoup-based scraper was tested to ensure it correctly extracted relevant fields (title, description, label) from the BBC Fake News section without introducing incomplete or malformed records. The insertion of scraped articles into MongoDB and the appending of new entries into the JSON dataset (real-news.json) were validated for consistency and correctness. The dataset management pipeline was further tested to ensure that both real and synthetic fake news articles were accurately integrated into the dataset for periodic model retraining.

API Testing was conducted to validate the backend APIs independently. Using tools such as Postman, the Flask /predict and /retrain endpoints were rigorously tested to verify correct functionality. For /predict, tests included sending various types of textual inputs (short headlines, long articles, edge cases) to ensure accurate classification and meaningful responses. For /retrain, tests ensured that model retraining was correctly triggered when new data was appended. The Node.js backend APIs were similarly tested to confirm proper retrieval of news articles stored in MongoDB and delivery of accurate responses to frontend requests.

Deployment and Cloud Testing was performed after deploying the frontend (on Vercel) and the backends (on Render). The objective was to ensure that all components functioned correctly in the live cloud environment. The frontend was tested for its accessibility over HTTPS, page load speed, and API communication with the cloud-hosted backends. Backend services on Render were tested to verify that they auto-scaled correctly, remained accessible under load, and maintained stable operation across deployment cycles. The interaction between Vercel and both Render-based backends was validated extensively to confirm consistent end-to-end functionality.

Security Testing was integrated throughout the system validation process. CORS (Cross-Origin Resource Sharing) configurations were verified to ensure that only legitimate requests from the frontend could access the backend APIs. User inputs in the frontend form were sanitized to prevent potential cross-site scripting (XSS) or injection attacks. HTTPS communication between frontend and backends was confirmed to ensure encrypted and secure data transmission. The web scraper was also tested to comply with ethical scraping practices, such as respecting public access restrictions and not overloading target servers.

Overall, the extensive testing process ensured that each module of the fake news detection

system- frontend, web scraping, dataset management, model retraining, and API interaction- functioned reliably both in isolation and as part of the integrated system. The testing strategy contributed significantly to the robustness, security, and usability of the deployed application.

Chapter 9
Deployment Strategies

The deployment strategy for this project was carefully designed to ensure scalability, accessibility, and seamless integration of all system components. The project architecture consisted of three main components the frontend application, the Flask backend (which handled the core functionalities of fake news prediction and model retraining), and the Node.js backend (which served the scraped BBC news articles to the frontend). Each of these components was deployed on cloud-based platforms to ensure ease of access, robust performance, and minimal operational overhead.

The frontend application, built using React.js and styled with Tailwind CSS, was deployed on Vercel. Vercel offers an optimized environment for React-based applications, providing features such as automatic continuous deployment from the project's GitHub repository, built-in content delivery network (CDN) distribution, and HTTPS-secured hosting. By leveraging Vercel, the frontend benefits from fast page loads, high availability, and global scalability. Any changes pushed to the main branch of the repository trigger an automatic deployment, ensuring that the latest version of the frontend is always accessible to users. The frontend serves as the main interface where users input news text for prediction and view the latest scraped news articles.

The Flask backend, which hosts the core APIs /predict and /retrain, was deployed independently on Render (a cloud application hosting platform that supports Python-based applications). The Flask backend is responsible for processing user-inputted news articles, classifying them as real or fake, and retraining the machine learning model whenever new data is appended to the dataset. Deploying this backend separately on Render ensures that prediction and retraining processes are isolated from other services, allowing for optimized resource allocation and scaling based on workload demands. Render's managed service simplifies backend deployment by handling server provisioning, HTTPS setup, and scaling automatically.

The Node.js backend, which fetches and serves the latest BBC news articles scraped and stored in MongoDB, was also deployed on Render as a separate service. This backend provides the necessary APIs for the frontend to retrieve and display real-time news articles on the user

interface. Hosting the Node.js API separately allows for better modularity and maintains clear boundaries between the system's different functionalities. It ensures that the scraping-related API can operate independently without impacting the prediction functionalities of the Flask backend.

By deploying each component on specialized platforms, the system architecture benefits from a microservices-style deployment, improving maintainability, scalability, and fault isolation. The frontend on Vercel communicates securely with both backends over HTTPS, ensuring safe and reliable data exchange. Both Render-deployed backends operate independently, allowing them to scale or restart as needed without affecting other parts of the system.

This deployment strategy offers multiple advantages: It ensures that user interaction via the frontend is consistently fast and reliable (thanks to Vercel's CDN and optimizations), and that backend services can independently handle prediction requests and data fetching tasks with optimal resource usage. Additionally, separating the services reduces the risk of system-wide failures, simplifies debugging, and allows each module to evolve or scale independently in the future.

Overall, the chosen cloud-based deployment architecture guarantees a robust, maintainable, and secure operational environment that meets the functional and non-functional requirements of the fake news detection system.

9.1 Security Aspects

Ensuring the security of the deployed system was a key consideration throughout the design and deployment of the fake news detection platform. The project architecture involved multiple cloud-hosted components that communicated over the internet, making it essential to safeguard data transmission, API access, and overall system integrity.

All communications between the frontend (deployed on Vercel) and the backend services (deployed on Render) were configured to operate over **HTTPS**, ensuring that data exchanges, including user input and API responses, were encrypted and secure from interception or tampering. Both Render and Vercel provide automatic HTTPS support, allowing seamless enforcement of encrypted communication channels without additional manual configuration.

Cross-Origin Resource Sharing (CORS) policies were properly configured on both Flask

and Node.js backends to restrict API access exclusively to requests originating from the deployed frontend. This ensured that only authorized frontend applications could communicate with backend services, preventing unauthorized access attempts from external or malicious sources.

The **Flask backend APIs**, which handle sensitive operations such as model prediction and retraining, were designed to validate all incoming requests and ensure that retraining actions are triggered only under appropriate conditions. The retraining API was tested to prevent abuse, such as unnecessary or repeated retraining requests that could exhaust server resources.

Additionally, the **web scraping component** was developed following ethical and responsible scraping guidelines. Care was taken to respect the public accessibility of target news sources (BBC), avoid overloading their servers with excessive requests, and ensure that scraped data did not violate usage policies.

By deploying the frontend and backends on reputable cloud platforms like Vercel and Render, the project leveraged their inherent **platform-level security features**. These included automatic HTTPS certificates, secure server environments, managed databases (for MongoDB access), and automatic resource isolation between deployed services.

Overall, the security aspects integrated into the deployment strategy ensured the protection of user data, the integrity of backend services, and the safe operation of the system in a cloud-hosted environment. These measures contribute to the long-term sustainability, reliability, and trustworthiness of the fake news detection platform.

Chapter 10
Result and Analysis

In this experiment, the machine learning model was tested using four different train-test split ratios: 80-20, 70-30, 60-40, and 50-50. The evaluation was conducted weekly over three weeks and each run was assessed using four key performance metrics: Accuracy, Precision, Recall, and F1-score. The goal was to analyze how the amount of training data influences model performance across different time periods.

Number of real news article in week 3 is 580 and fake news article is week 3 is 566.

10.1 Performance Metrics

Table 5. Performance Metrics for 80-20 Train-Test Split

Train-Test Split **80-20**	**Accuracy**	**Precision**	**Recall**	**F1-score**
Week 1	0.7826	0.7647	0.9286	0.8387
Week 2	0.6700	0.7500	0.5660	0.6452
Week 3	0.7913	0.7895	0.7895	0.7895

The 80-20 train-test split consistently delivered the best performance overall. In Week 1, the model achieved high scores in all metrics, particularly a Recall of 0.9286 and an F1-score of 0.8387, indicating its effectiveness in identifying true positive cases and maintaining a good balance between precision and recall. Week 3 also demonstrated solid and balanced performance across all metrics, suggesting that the model performs reliably when trained on 80% of the data. However, Week 2 under this split showed a noticeable drop in recall, which may point to irregularities or noise in the data collected during that period.

Table 6. Performance Metrics for 70-30 Train-Test Split

Train-Test Split 70-30	Accuracy	Precision	Recall	F1-score
Week 1	0.7143	0.6786	0.9500	0.7917
Week 2	0.6711	0.7183	0.6375	0.6755
Week 3	0.7238	0.7176	0.7219	0.7198

With the 70-30 split, there was a slight decrease in overall performance compared to the 80-20 split, but the model still showed robust results. In Week 1, the model maintained a high F1-score of 0.7917 and a very strong recall of 0.9500, although precision declined slightly. Week 2 and Week 3 produced relatively balanced values for all metrics, indicating that the model was still capable of learning effectively with 70% of the data, though it benefitted from more training data in the 80-20 case.

Table 7. Performance Metrics for 60-40 Train-Test Split

Train-Test Split 60-40	Accuracy	Precision	Recall	F1-score
Week 1	0.6739	0.6486	0.9231	0.7619
Week 2	0.6131	0.6304	0.5743	0.6010
Week 3	0.7211	0.7124	0.7188	0.7156

The performance began to degrade more noticeably with the 60-40 split. In Week 1, the recall remained high at 0.9231, but both precision and accuracy dropped, affecting the overall F1-score. Week 2 showed the weakest results under this split, with all four metrics dipping significantly. This suggests that with only 60% of the data for training, the model's ability to generalize decreased. Week 3, while slightly better, still did not reach the performance levels observed with the higher training ratios.

Table 8. Performance Metrics for 50-50 Train-Test Split

Train-Test Split 50-50	Accuracy	Precision	Recall	F1-score
Week 1	0.7193	0.6818	0.9375	0.7895
Week 2	0.6371	0.6230	0.6333	0.6281
Week 3	0.7277	0.7168	0.7220	0.7194

In the 50-50 split, performance showed a mixed trend. Week 1 surprisingly still showed strong recall and an F1-score close to that of higher split ratios, but as with other splits, Week 2 again showed weaker performance in all metrics. Week 3 showed relatively stable values but was not significantly better than the 60-40 split. This indicates that while the model can function with 50% training data, the trade-off in performance becomes more apparent.

In conclusion, the results clearly show that the model performs best when trained on a larger portion of the dataset. The 80-20 and 70-30 splits yielded the most favorable and consistent results, especially in terms of recall and F1-score. As the training data decreased, performance dropped, particularly in Week 2 across all splits. This highlights the importance of using sufficient training data for achieving reliable and generalizable machine learning outcomes. Overall, the study emphasizes that data quantity and potentially data quality significantly influence model success over time.

10.2 Graphs

The graph compares the accuracy of SVM models trained over three weeks using different train-test splits (50-50 to 80-20). Week 3's model, trained on the largest dataset, shows the highest and most consistent accuracy, peaking at 0.7913 with an 80-20 split. Week 1's model, despite using the smallest dataset, performs well with high recall and reaches 0.7826 accuracy, suggesting good generalization. In contrast, Week 2's model has the lowest and flattest accuracy curve, likely due to an imbalanced or less effective dataset. Overall, more data and balanced classes improved model performance over time.

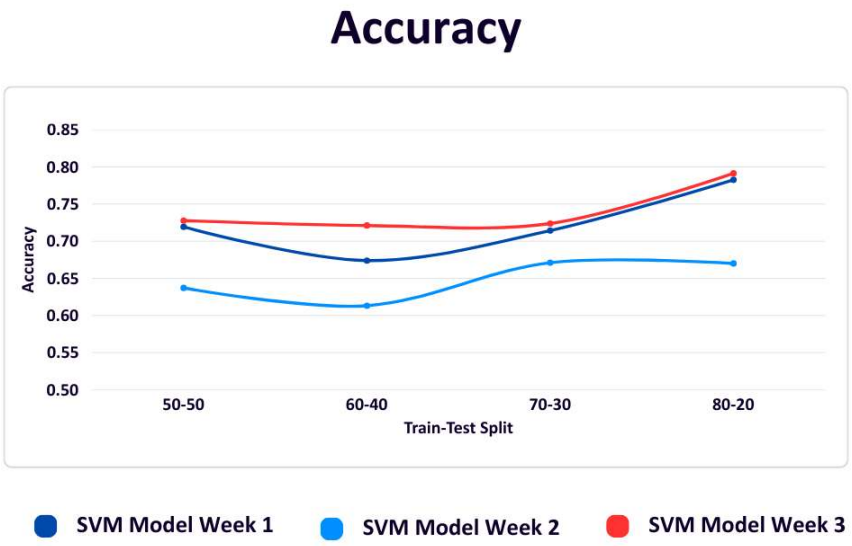

Fig. No. 6

The Precision graph shows that the SVM model's precision improves as the training data increases (toward the 80-20 split) for all three weeks. Week 3 (red) performs best overall due to having the largest and most balanced dataset. Week 1 (dark blue), despite limited data, maintains decent precision. Week 2 (light blue) starts lower but improves steadily with more training data. This highlights that more and balanced data leads to more accurate predictions of fake news.

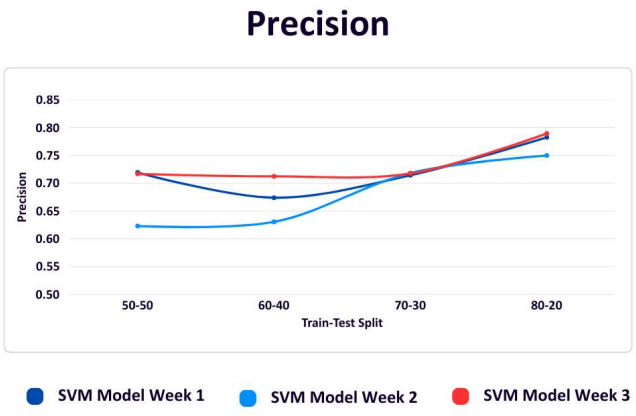

Fig. No. 7

The Recall graph shows how well the SVM model correctly identifies fake news across different train-test splits over three weeks. Week 1 (dark blue) consistently has very high recall-above 0.92, due to a smaller dataset and possibly overfitting. Week 3 (red), with more balanced and larger data, shows a steady increase in recall, peaking at 0.7895 in the 80-20 split, indicating better real-world performance. Week 2 (light blue) fluctuates more and has lower recall, reflecting challenges from limited and less balanced data. Overall, recall improves with more training data, especially in Week 3.

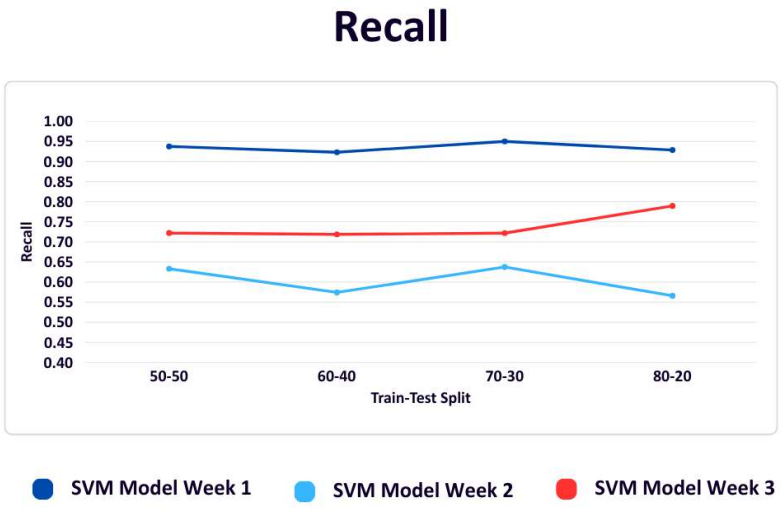

Fig. No. 8

The F1-score graph summarizes the balance between precision and recall for the SVM model across different weeks and train-test splits. In Week 1 (dark blue), performance improves from 0.67 to 0.78 as the training data increases. Week 2 (light blue) has the lowest F1-scores overall, ranging from 0.60 to 0.675, due to lower recall and precision. Week 3 (red) maintains a stable F1-score early on and shows the best result (0.7857) in the 80-20 split, reflecting strong generalization as training data increases. Overall, the F1-score improves with larger training sets, and Week 3 shows the most balanced and robust performance.

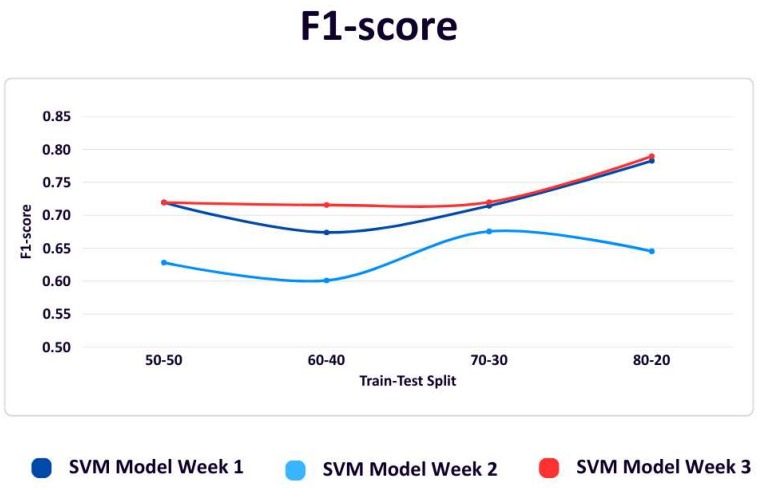
F1-score
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
F1-score
50-50
60-40
70-30
80-20
Train-Test Split
SVM Model Week 1
SVM Model Week 2
SVM Model Week 3

Fig. No. 9

Applications

1. **Social Media Monitoring Tools**

   Useful for platforms like Twitter, Facebook, and Instagram to detect and curb the spread of misinformation.

2. **Browser Extensions**

   Can be deployed as a browser plugin to help users verify the authenticity of news articles they read online.

3. **Fact-Checking Organizations**

   Supports fact-checking agencies by automating the detection of potentially fake or manipulated news articles.

4. **Government and Law Enforcement**

   Aids in identifying and tracking the origin and spread of harmful misinformation, which could impact public safety or elections.

5. **Academic and Research Use**

   Provides a research basis for further studies in NLP, model generalization, and real-time model retraining.

6. **Corporate Reputation Management**

   Helps organizations monitor and respond to false information that could affect their brand image.

Conclusion

The project successfully demonstrates the implementation of a robust machine learning system for fake news detection using comparative analysis between Support Vector Machine (SVM), Logistic Regression (LR) and Naive Bayes (NB). By focusing on accuracy, generalization, and adaptability, the module effectively classifies news articles as real or fake and integrates into a dynamic model updating framework.

Through regular data updates and retraining, the system remains current with evolving news patterns, addressing a major gap identified in existing fake news detection research. Comparative analysis showed that the SVM model outperformed baseline models like Logistic Regression and Naïve Bayes, making it the most suitable choice for dynamic deployment.

The project not only highlights the potential of machine learning in combating misinformation but also sets a foundation for future improvements such as deep learning integration, multilingual support and real-time detection systems.

Future prospects of the project

The current implementation of the fake news detection system lays the foundation for a scalable and adaptable solution. In the future, the project can be expanded in several key areas:

1. Integration of Deep Learning Models: Incorporating advanced models like LSTM, BERT, or transformers could enhance the system's ability to understand context and semantics in complex news content.
2. Multilingual Support: Extending the system to detect fake news in regional and international languages would increase its applicability across diverse populations.
3. Real-Time Detection: Developing real-time APIs or browser plugins could allow users to verify news articles on-the-fly, improving the system’s usability and impact.
4. User Feedback Loop: Integrating a user feedback mechanism could further refine the model through reinforcement learning, enhancing dynamic model updating.
5. Deployment as a Public Tool: The system can be hosted as a public web platform or integrated with social media platforms to flag potentially misleading news content.
6. Automated Data Collection: Enhancing the data pipeline with web crawlers and scrapers that autonomously gather recent articles would strengthen periodic retraining and keep the model current.
7. Image and Video Analysis: Expand beyond text to detect fake content in multimedia using computer vision.

Part B: Individual Contributions
Module 1- Frontend and Web Scraping

Problem Statement:

In today's digital age, the rapid spread of fake news through online platforms has significant societal, political, and economic impacts. A major limitation in current fake news detection systems is that they mostly work on static datasets and offer limited ways to interactively check or validate information online. This module aims to build a dynamic, user-friendly web application where users can not only check the authenticity of news but also continuously enrich the fake news detection system by scraping real articles from trusted sources (like BBC). Moreover, a mechanism to generate synthetic fake news to balance datasets and periodically retrain the detection model has been incorporated. The module provides an integrated end-to-end system involving frontend, backend, scraping, and deployment.

Name of the Student: Vinay More
Module Title: Frontend Development, Web Scraping, and Dataset Management for Dynamic Fake News Detection

Project's Module Objectives

The objective of this module is to build a full-stack system that offers users an interactive platform for fake news detection while simultaneously supporting continuous model improvement. To achieve this, a responsive and visually appealing frontend interface was developed using React.js and Tailwind CSS.

An automated web scraping mechanism was implemented using BeautifulSoup to collect real news articles from the BBC website and store them in a MongoDB database for future model retraining. A dataset management system was created to dynamically append new articles to a JSON file (`real-news.json`) ensuring a balanced dataset.

Additionally, synthetic fake news articles were generated using HuggingFace Transformers to supplement the dataset with artificial samples. The system supports dynamic retraining by

integrating backend APIs, facilitating seamless communication between the frontend and backend services for both prediction and retraining tasks.

Project's Module Scope

This module covers the design and implementation of a full-stack solution with several components. It includes a frontend web interface that enables user interaction, news input, and news display. The system also incorporates web scraping functionality to gather real news articles and store them both in a database (MongoDB) and a JSON file for dataset expansion. The module integrates a synthetic fake news generation mechanism to balance the dataset. Dataset management and a retraining pipeline were implemented to ensure the model is updated regularly with new data. To provide a seamless user experience, the system interacts with two backend services: a Flask backend for prediction and retraining, and a Node.js backend for serving the latest scraped articles. Finally, the entire system, including frontend and backend components was deployed on cloud platforms to ensure accessibility and scalability.

Project's Module(s)

Hardware & Software Requirements

The hardware requirements for this module included a standard system with a dual-core processor (or higher), a minimum of 8GB RAM, and at least 4GB of available storage space. The software stack consisted of React.js and Tailwind CSS for frontend development, Python 3.x with BeautifulSoup4 and Requests for web scraping, and MongoDB for storing scraped news articles. HuggingFace Transformers were utilized for synthetic fake news generation. Node.js with Express.js served the scraped news to the frontend, and Flask provided APIs for prediction and retraining functionalities. Development environments used were Visual Studio Code and Jupyter Notebook.

Module Interfaces

The frontend interface accepts user textual input for fake/real prediction and displays news articles scraped from BBC. The preprocessing interface cleans the scraped data before storing it in MongoDB or the JSON dataset. The backend interface, consisting of Flask APIs (/predict, /retrain), handles prediction requests and facilitates periodic retraining using new data. The output interface returns prediction results on the frontend and showcases the latest news articles fetched via the Node.js API.

Module Dependencies

The frontend interface accepts user textual input for fake/real prediction and displays news articles scraped from BBC. The preprocessing interface cleans the scraped data before storing it in MongoDB or the JSON dataset. The backend interface, consisting of Flask APIs (/predict, /retrain), handles prediction requests and facilitates periodic retraining using new data. The output interface returns prediction results on the frontend and showcases the latest news articles fetched via the Node.js API.

Module Design

The frontend was designed using React.js and Tailwind CSS to provide a clean interface, including a text input form and cards to display BBC news. The web scraper was built using Python's BeautifulSoup to extract articles from BBC's Fake News section. These articles, containing title, description, and label, were stored in both MongoDB and appended to the JSON file (real-news.json). The dataset management component automatically updated the dataset with new articles. HuggingFace Transformers were employed to generate synthetic fake news articles. The backend architecture involved Flask APIs for prediction and retraining functionalities, while a Node.js backend served the scraped news articles. The entire deployment architecture included the frontend hosted on Vercel, and both Flask and Node.js backends deployed on Render.

Module Implementation

The frontend was implemented by building reusable React.js components styled using Tailwind CSS, resulting in a responsive NewsHub interface. The web scraper was implemented using Python BeautifulSoup scripts, which collected BBC articles, cleaned them, and stored

the results in MongoDB and JSON files. HuggingFace Transformers were utilized to generate synthetic fake news samples. A Node.js backend was created to fetch and serve the MongoDB-stored articles to the frontend. Flask APIs `/predict` and `/retrain` were implemented to manage model interaction and retraining tasks. All services, frontend and both backends, were deployed on Vercel and Render cloud platforms. The dataset management scripts were configured to dynamically append new articles to the JSON file and automatically trigger retraining pipelines.

Module Testing Strategies

Extensive testing was conducted to validate the system. Manual testing ensured the scraping pipeline correctly inserted data into MongoDB and the JSON file. Functional testing was performed to verify the frontend input forms, news display components, and API interactions. Flask APIs were tested using Postman as well as through frontend integration. The integrity of dataset updates and the retraining pipeline was validated to ensure correctness. Finally, deployment testing was conducted to confirm that communication between frontend and backend services worked correctly on the live servers.

Module Deployment

The frontend application was deployed on Vercel, providing the news website interface, input form, and result display. The Flask backend, responsible for prediction and retraining functionalities, was deployed on Render. The Node.js backend, which serves BBC scraped news articles to the frontend, was also deployed on Render. The cloud-based deployment ensured scalability, accessibility, and secure interaction between all system components.

Module 2- Support Vector Machine

Problem Statement:
Develop a robust machine learning module capable of classifying news articles as real or fake using Support Vector Machine (SVM), with a focus on accuracy, generalization, and adaptability for integration into a dynamic model updating framework.

Name of the Student: Mansi Manoj Patil

Module Title:
SVM-Based Fake News Classification and Integration into Dynamic Model Updating Framework

Project's Module Objectives

- To implement and evaluate an SVM classifier for fake news detection.
- To optimize the model for high accuracy and generalization using appropriate preprocessing and feature extraction techniques.
- To enable periodic retraining as part of the dynamic model updating system.
- To provide a scalable and adaptable classification component for the full system architecture.

Project's Module Scope

This module is responsible for the classification of preprocessed news articles using the Support Vector Machine algorithm. It covers the model training, hyperparameter tuning, evaluation against other models, and final selection for dynamic retraining. The module also interfaces with data processing and updating components for weekly retraining with new datasets.

Project's Module(s)

Hardware & Software Requirements

- **Hardware:** Standard system with minimum 8 GB RAM and i5 or equivalent processor
- **Software:**

- Python 3.x
- scikit-learn
- NumPy, Pandas
- Jupyter Notebook
- TF-IDF Vectorizer
- Matplotlib / Seaborn (for evaluation visuals)

Module Interfaces

The **Module Interfaces** are designed to ensure seamless integration with other components of the fake news detection system. This module primarily receives input from the data preprocessing unit, where raw text data is cleaned and transformed into numerical form using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. These vectors represent the significant features extracted from the news articles, which are then fed into the SVM classifier for analysis. Once the classification is performed, the module generates output in the form of prediction labels assigning a '0' for real news and a '1' for fake news along with key evaluation metrics such as accuracy, precision, recall, and F1-score. Additionally, the module is designed to interface with the dynamic model updating script, enabling the classifier to be periodically retrained using newly collected and preprocessed datasets. This setup ensures the model remains current and maintains high performance over time.

Module Dependencies

The module depends on a consistent supply of preprocessed and vectorized data, typically generated through techniques like TF-IDF, to ensure that input text is in a structured and machine-readable format suitable for classification. It is tightly integrated with evaluation metric scripts that facilitate performance comparison across different models using standard metrics such as accuracy, precision, recall, and F1-score. These scripts are essential for validating the model's effectiveness and selecting the best-performing algorithm. Additionally, the module is designed to work within a dynamic updating framework and is dependent on regular weekly updates of the dataset. These updates are crucial for retraining the model, allowing it to adapt to emerging trends and patterns in fake news content, thus ensuring sustained accuracy and robustness over time.

Module Design

The module is designed around a Support Vector Machine (SVM) classifier configured with a linear kernel, which offers a good balance between simplicity, computational efficiency, and classification performance especially suitable for high-dimensional text data. For feature extraction, the module uses a Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer with an n-gram range of (1,2), allowing the model to capture both individual words and short word sequences, thereby improving context understanding. Additional parameters like max_df=0.9 and min_df=2 are set to filter out extremely common and rare terms, ensuring that only meaningful and informative features are retained. To ensure a fair and representative evaluation, the dataset is split into training and testing sets using stratified sampling, which maintains the original class distribution. Model performance is then rigorously evaluated using key metrics such as accuracy, precision, recall, and F1-score, which collectively provide a comprehensive understanding of the classifier's effectiveness in distinguishing between real and fake news.

Module Implementation

The implementation of the module was carried out using the SVC (Support Vector Classification) class from the scikit-learn library, which provides a reliable and flexible framework for training Support Vector Machine models. The classifier was trained on a labeled dataset where each news article was tagged as either real (0) or fake (1), ensuring clear distinction during supervised learning. As part of the implementation process, the SVM model was rigorously compared against other popular classification algorithms, Logistic Regression and Naïve Bayes, to assess its relative performance. Through comprehensive evaluation using accuracy, precision, recall, and F1-score, the SVM model consistently outperformed the alternatives, establishing itself as the most suitable classifier for this task. Following its selection, the model was seamlessly integrated into a dynamic retraining pipeline that supports periodic updates. This integration enables the model to automatically retrain on newly scraped data each week, thus maintaining its relevance and effectiveness in detecting evolving patterns of fake news over time.

Module Testing Strategies

To evaluate the stability and robustness of the model, multiple test splits were performed, including 80-20 and 70-30 train-test ratios. Cross-validation techniques were employed to ensure the model's generalizability across different data subsets. The performance metrics of

the Support Vector Machine model were compared with baseline models such as Logistic Regression and Naïve Bayes to assess relative effectiveness. Additionally, the dynamic retraining process was validated by testing the model on incrementally appended datasets collected weekly, ensuring consistent performance over time.

Module Deployment

The SVM-based fake news detection model was deployed as a callable Python module within the broader project architecture. It included exposed functions that allowed seamless execution of tasks such as model training, evaluation, and retraining. To ensure the model stays current, it was integrated with scheduled automation scripts using tools like cron or task scheduler, enabling weekly updates with newly collected data. Furthermore, the trained model was serialized using joblib, allowing efficient storage and reuse for real-time predictions without the need for retraining each time.

Module 3- Logistic Regression

Problem Statement: In the digital era, social media platforms and online news outlets play a significant role in disseminating information. However, the rapid spread of fake news- false or misleading content presented as legitimate news- has emerged as a serious threat to public trust, democratic processes, and societal harmony. Traditional methods of manually verifying news articles are time-consuming, inefficient, and impractical at scale.

To address this issue, the project aims to develop a machine learning-based fake news detection system using Logistic Regression. The system will classify news articles as fake or real based on the content of the text. The project seeks to build a lightweight yet effective model that can help automatically detect misinformation and reduce its spread online.

Name of the Student: Preshika Giri

Module Title: Fake News Detection using Logistic Regression

Project's Module Objectives:

The objective of this module is to build a fake news detection system using Logistic Regression, a supervised learning algorithm ideal for binary classification. The process begins with text preprocessing, applying lowercasing, punctuation removal, stopword elimination, and lemmatization, to clean the data. This is followed by TF-IDF vectorization to convert text into meaningful numerical features. The dataset is then split into training and testing sets, and the Logistic Regression model is trained with regularization to enhance generalization. Model performance is evaluated using metrics like accuracy, precision, recall, and F1-score. Finally, comparisons with other algorithms such as SVM and Naive Bayes help validate Logistic Regression as a strong baseline model for classifying news as real or fake.

Project's Module Scope:

This module focuses on implementing Logistic Regression as a baseline model for fake news detection using a labeled dataset. It involves text preprocessing (lowercasing, stopword removal, lemmatization), feature extraction using TF-IDF, and splitting the data into training and testing sets. The model is trained and evaluated using metrics like accuracy, precision, recall, and F1-score. Its performance is compared with models like SVM and Naive Bayes to

assess suitability.

Project's Module(s):

Hardware & Software Requirements:

Hardware:

CPU: Dual-core processor or higher

RAM: Minimum 4GB

Storage: 2GB free space

Software:

Operating System : Windows/ Linux/ macOS

Programming Language: Python

Libraries: scikit-learn, pandas, numpy, nltk, joblib

IDE/Editor: Jupyter Notebook or Python IDE or VS Code

Dataset Source: Kaggle

Module Interfaces:

The module incorporates a well-structured set of interfaces to streamline the fake news detection pipeline using Logistic Regression. The Input Interface is designed to accept textual data from various formats such as .csv, .txt, or direct user input, providing flexibility in how news content is ingested into the system. Once received, the Preprocessing Interface takes over, performing essential text-cleaning operations including tokenization, stop-word removal, lowercasing, and feature vectorization using TF-IDF. This ensures that the raw text is transformed into a structured numerical format suitable for model consumption. The processed data is then passed to the Model Interface, which is responsible for training and managing the Logistic Regression model. It takes the feature vectors as input and outputs binary classification results indicating whether the news is "Fake" or "Real." For interpretability and evaluation purposes, the model may also return associated probabilities or confidence scores. Finally, the Output Interface presents the results in a user-readable format, enabling users to understand the classification outcome clearly. The overall system is operated and managed through the User Interface, which in this case is Visual Studio Code (VS Code), serving as the primary development and testing environment for the module.

Module Dependencies:

The module depends primarily on a labeled dataset of real and fake news articles, typically in CSV format, containing fields such as the title, full text, and corresponding labels indicating whether the news is real or fake. This dataset forms the basis for training and evaluating the Logistic Regression model. In terms of software dependencies, several key Python libraries are required. The pandas library is used for loading and managing structured data efficiently, while numpy supports numerical operations and array handling. The scikit-learn library is essential for implementing the machine learning pipeline, including TF-IDF vectorization, model training using Logistic Regression, performance evaluation, and splitting the dataset into training and testing sets. Additionally, the nltk (Natural Language Toolkit) library is used for various text preprocessing tasks such as tokenization, stop-word removal, and lemmatization, which help in cleaning and preparing the text data for effective feature extraction. These dependencies together enable the complete workflow of the fake news detection module.

Module Design:

The module begins with loading and exploring a labeled dataset containing real and fake news. Text preprocessing is applied, including lowercasing, punctuation removal, stopword elimination, and lemmatization. The cleaned text is then transformed into numerical features using TF-IDF vectorization. The dataset is split into training and testing sets to prepare for model training.

A Logistic Regression model is trained with regularization to prevent overfitting. Model performance is evaluated using accuracy, precision, recall, and F1-score. To validate its effectiveness, the Logistic Regression model is compared against other baseline classifiers such as SVM and Naive Bayes. Finally, results are summarized and visualized to support analysis and interpretation.

Module Implementation:

1. **Dataset Loading** – A labeled dataset containing news articles classified as "real" or "fake" was imported, typically in CSV format. This dataset serves as the foundation for training and evaluating the fake news detection model.

2. **Text Preprocessing** – The raw news text was cleaned and normalized to prepare it for

machine learning. This involved converting text to lowercase, removing punctuation and special characters, tokenizing sentences into words, eliminating common stopwords (e.g., “the”, “and”), and applying lemmatization to reduce words to their root forms, improving consistency and reducing dimensionality.

3. **Feature Extraction** – After preprocessing, the textual data was transformed into numerical format using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. This technique assigns weights to words based on their frequency and uniqueness across the corpus, enabling the model to focus on meaningful terms that differentiate real from fake news.
4. **Train-Test Split** – The dataset was divided into training and testing subsets (80:20 ratio). The training set was used to fit the model, while the testing set was reserved to evaluate how well the model generalizes to unseen data.
5. **Model Training** – The Logistic Regression algorithm was applied to the training data. It learned patterns by mapping the TF-IDF feature vectors to their corresponding class labels (real or fake). Hyperparameters, such as regularization strength, were tuned to enhance generalization and reduce overfitting.
6. **Model Evaluation** – The trained model was assessed using the test data. Evaluation metrics included accuracy (overall correctness), precision (relevance of positive predictions), recall (ability to identify all positives), and F1-score (harmonic mean of precision and recall). These metrics provided a comprehensive view of the model's performance.
7. **Prediction** – Finally, the model was used to generate predictions on the test dataset. The predicted results were compared with the actual labels to validate performance and check for misclassifications. This step also involved analyzing the model’s probability scores to understand its confidence in predictions.

Module Testing Strategies:

- Train/Test Split Validation:

  The dataset was split into training (80%) and testing (20%) sets to evaluate model generalization.

  This ensured the model wasn't overfitting and could handle unseen data.
- Performance Metrics:

  Evaluated the model using:

  i. **Accuracy** – overall correctness

ii. **Precision** – correctness of positive predictions

iii. **Recall** – model's ability to find all fake news

iv. **F1-Score** – harmonic mean of precision and recall

v. **Confusion Matrix** – visual verification of TP, TN, FP, FN

Module Deployment:

- Model Packaging:

The trained Logistic Regression model and TF-IDF vectorizer were saved using joblib or pickle

Module 4 - Naive Bayes Implementation

Problem Statement: The rise of fake news on digital platforms poses a serious threat to society by spreading misinformation. This module aims to build a system using the Naive Bayes algorithm to classify news articles as real or fake based on their textual content. The model will be trained on a labeled dataset, evaluated using standard metrics and saved for future use in detecting fake news efficiently.

Name of the Student: Maitreyee Jadhav

Project's Module Objectives:

The main objective of this module is to design and implement a machine learning-based classification model using the Naive Bayes algorithm to detect fake news articles. The goal is to preprocess the textual data, train the model, evaluate its accuracy and save the trained model for future use in real-time or batch processing systems.

The main idea behind the Naive Bayes classifier is to use Bayes' Theorem to classify data based on the probabilities of different classes given the features of the data. It is used mostly in high-dimensional text classification

The Naive Bayes Classifier is a simple probabilistic classifier and it has very few parameters which are used to build the ML models that can predict at a faster speed than other classification algorithms.

It is a probabilistic classifier because it assumes that one feature in the model is independent of the existence of another feature. In other words, each feature contributes to the predictions with no relation between each other.

Project's Module Scope:

1. Dataset Preparation for Classification:

   This step involves collecting raw text data and converting it into a structured format suitable for classification tasks. The dataset is cleaned by removing irrelevant content, handling missing values, and labeling the data appropriately.
2. Feature Extraction Using Text Vectorization (TF-IDF):

   The cleaned textual data is transformed into numerical representations using Term Frequency-Inverse Document Frequency (TF-IDF). This technique weighs the importance of each word relative to the entire dataset, allowing the model to focus on

informative terms.

3. Model Building and Training – Naive Bayes:

   A Multinomial Naive Bayes classifier is used for training. This algorithm is especially effective for text-based classification tasks. The model is trained on the TF-IDF feature matrix and the corresponding labels.

4. Model Evaluation:

   The trained model is evaluated using classification metrics such as accuracy, precision, recall, and F1-score. These metrics provide insight into how well the model is performing on unseen data and help identify areas for improvement.

5. Model Persistence:

   To make the model reusable without the need for retraining, it is saved using serialization libraries like joblib or pickle. This enables fast deployment and integration into applications.

Project's Module(s):

1. Data Cleaning and Preprocessing:

   This module is responsible for preparing the text data by performing operations such as tokenization, converting to lowercase, removing punctuation, eliminating stopwords, and other relevant preprocessing tasks to enhance data quality.

2. Text Vectorization Using TF-IDF:

   This module takes the cleaned text and converts it into a numerical feature matrix using the TF-IDF vectorizer, which helps the model understand the importance of each term in the document.

3. Model Training Using the Multinomial Naive Bayes Algorithm:

   In this module, the Multinomial Naive Bayes classifier is trained using the TF-IDF features and their corresponding labels. It learns the probability distributions required to make predictions.

4. Model Evaluation Using Accuracy, Precision, Recall, and F1-score:

   This module evaluates the model's predictions by comparing them with actual labels. It calculates key metrics to understand the effectiveness of the classifier.

5. Saving the Model Using Joblib:

   Once the model is trained and evaluated, it is saved to disk using joblib. This allows the model to be loaded and used in future predictions without retraining.

6. Writing Modular Python Scripts for Reusability:

Each function or process (such as preprocessing, vectorization, training, and evaluation) is written as a separate script or function. This modular structure makes the codebase easier to manage and extend.

7. Creating APIs for Retrain and Predict:
   APIs are developed to allow external systems to interact with the model. A /retrain endpoint allows the model to be retrained with new data, and a /predict endpoint provides predictions for new input text, facilitating integration into real-time systems.

Hardware & Software Requirements:

Hardware:

CPU: Dual-core processor or higher

RAM: Minimum 4GB

Storage: 2GB free space

Software:

OS: Windows/Linux/macOS

Lanuguage: Python

Libraries: scikit-learn, pandas, numpy, nltk, joblib

Jupyter Notebook or any Python IDE or VS Code

Module Interfaces:

Input Interface: CSV file containing news text and corresponding labels (real/fake).

Output Interface: Printed evaluation metrics, and saved .pkl or .joblib file for the trained model.

Module Dependencies:

- Requires scikit-learn for Naive Bayes and metric calculation.
- Depends on nltk for natural language preprocessing.
- Relies on joblib or pickle for model serialization.

Module Design:

Preprocessing Pipeline: Load → Clean Text → Tokenize → Remove Stopwords → Vectorize

Model Pipeline: TF-IDF Vectorizer → Naive Bayes Classifier

Storage Pipeline: Trained model → Save as a .joblib file

Module Implementation:

THe Implementation involves developing a complete pipeline that begins with data preprocessing and ends with model persistence. Initially, raw text data from a CSV file was cleaned using natural language processing techniques, including tokenization, stopword removal, and lowercasing, utilizing the NLTK library. This cleaned data was then transformed into numerical features using the TF-IDF vectorizer to highlight the significance of terms within the dataset. The Multinomial Naive Bayes classifier from scikit-learn was then trained on these features to distinguish between fake and real news. After training, the model's performance was evaluated using key classification metrics such as accuracy, precision, recall, and F1-score. Finally, to enable future use without retraining, the model and vectorizer were serialized and saved using joblib. Modular Python scripts were written to ensure reusability, and RESTful APIs were created to allow prediction and retraining through endpoints, facilitating integration into larger systems or web applications.

Module Testing Strategies:

- Integration Testing: Ensure the full pipeline (preprocessing to evaluation) works end-to-end.
- Unit Testing using Postman for retrain and predict API.

Module Deployment:

- Save the trained model and vectorizer using joblib.
- Use the saved model in a separate inference script or a web app API.
- Ensure proper versioning of model files and Python dependencies for reproducibility.

Project to Outcome Mapping

Objectives:

1. Implement Data Collection, Preprocessing of Dataset, Model Training, Evaluation, API Integration and Deployment
2. Implementation of Dynamic Model Updating
3. Implementation of Model Integration and Deployment
4. Integrate Frontend and Backend

Table 9. Project to Outcome Mapping

Sr. No.	**PRN No.**	**Student Name**	**Individual Project Student Specific Objective**	**Learning Outcomes mapped (To be filled by Guide)**
1.	1032211369	Maitreyee Jadhav	Implement: Preprocessing Model Training Evaluation & Updates (For NB) API Development and Documentation	
2.	1032211811	Vinay More	Implement: Frontend/UI Data Collection - Web Scraping API Integration Deployment and Documentation	
3.	1032221625	Mansi Patil	Implement: Preprocessing, Model Training, Evaluation & Updates (For SVM) and	

			Documentation	
4.	1032221686	Preshika Giri	Implement: Preprocessing Model Training Evaluation & Updates (For LR) and documentation	

References

[1] W. Shahid et al., "Detecting and Mitigating the Dissemination of Fake News: Challenges and Future Research Opportunities," in IEEE Transactions on Computational Social Systems, vol. 11, no. 4, pp. 4649-4662, Aug. 2024, doi: 10.1109/TCSS.2022.3177359.
https://ieeexplore.ieee.org/document/9789171

[2] A. H. J. Almarashy, M. -R. Feizi-Derakhshi and P. Salehpour, "Enhancing Fake News Detection by Multi-Feature Classification," in IEEE Access, vol. 11, pp. 139601-139613, 2023, doi: 10.1109/ACCESS.2023.3339621.

https://ieeexplore.ieee.org/document/10343156

[3] M. Babar, A. Ahmad, M. U. Tariq and S. Kaleem, "Real-Time Fake News Detection Using Big Data Analytics and Deep Neural Network," in IEEE Transactions on Computational Social Systems, vol. 11, no. 4, pp. 5189-5198, Aug. 2024, doi: 10.1109/TCSS.2023.3309704.

https://ieeexplore.ieee.org/document/9965362

[4] J. Wang, S. Makowski, A. Cieślik, H. Lv and Z. Lv, "Fake News in Virtual Community, Virtual Society, and Metaverse: A Survey," in IEEE Transactions on Computational Social Systems, vol. 11, no. 4, pp. 4828-4842, Aug. 2024, doi: 10.1109/TCSS.2022.3220420.

https://ieeexplore.ieee.org/document/10038616

[5] W. Shahid et al., "Detecting and Mitigating the Dissemination of Fake News: Challenges and Future Research Opportunities," in IEEE Transactions on Computational Social Systems, vol. 11, no. 4, pp. 4649-4662, Aug. 2024, doi: 10.1109/TCSS.2022.3177359.

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9789171

[6] A. H. J. Almarashy, M. -R. Feizi-Derakhshi and P. Salehpour, "Enhancing Fake News Detection by Multi-Feature Classification," in IEEE Access, vol. 11, pp. 139601-139613, 2023, doi: 10.1109/ACCESS.2023.3339621.

https://ieeexplore.ieee.org/document/10343156

[7] M. Babar, A. Ahmad, M. U. Tariq and S. Kaleem, "Real-Time Fake News Detection Using Big Data Analytics and Deep Neural Network," in IEEE Transactions on Computational Social Systems, vol. 11, no. 4, pp. 5189-5198, Aug. 2024, doi: 10.1109/TCSS.2023.3309704.

https://ieeexplore.ieee.org/document/10241287

[8] S. A. Aljawarneh and S. A. Swedat, "Fake News Detection Using Enhanced BERT," in IEEE Transactions on Computational Social Systems, vol. 11, no. 4, pp. 4843-4850, Aug. 2024, doi: 10.1109/TCSS.2022.3223786.

https://ieeexplore.ieee.org/document/9965362

[9] J. Wang, S. Makowski, A. Cieślik, H. Lv and Z. Lv, "Fake News in Virtual Community, Virtual Society, and Metaverse: A Survey," in IEEE Transactions on Computational Social Systems, vol. 11, no. 4, pp. 4828-4842, Aug. 2024, doi: 10.1109/TCSS.2022.3220420.

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10038616

[10]G. G. Devarajan, S. M. Nagarajan, S. I. Amanullah, S. A. S. A. Mary and A. K. Bashir, "AI-Assisted Deep NLP-Based Approach for Prediction of Fake News From Social Media Users," in IEEE Transactions on Computational Social Systems, vol. 11, no. 4, pp. 4975-

4985, Aug. 2024, doi: 10.1109/TCSS.2023.3259480.

https://ieeexplore.ieee.org/document/10086954

[11]J. Wang, S. Makowski, A. Cieślik, H. Lv and Z. Lv, "Fake News in Virtual Community, Virtual Society, and Metaverse: A Survey," in IEEE Transactions on Computational Social Systems, vol. 11, no. 4, pp. 4828-4842, Aug. 2024, doi: 10.1109/TCSS.2022.3220420.

https://ieeexplore.ieee.org/document/10038616

[12]S. A. Aljawarneh and S. A. Swedat, "Fake News Detection Using Enhanced BERT," in IEEE Transactions on Computational Social Systems, vol. 11, no. 4, pp. 4843-4850, Aug. 2024, doi: 10.1109/TCSS.2022.3223786.

https://ieeexplore.ieee.org/document/9965362

[13]A. Altheneyan and A. Alhadlaq, "Big Data ML-Based Fake News Detection Using Distributed Learning," in IEEE Access, vol. 11, pp. 29447-29463, 2023, doi: 10.1109/ACCESS.2023.3260763.

https://ieeexplore.ieee.org/document/10078408

[14]M. Babar, A. Ahmad, M. U. Tariq and S. Kaleem, "Real-Time Fake News Detection Using Big Data Analytics and Deep Neural Network," in IEEE Transactions on Computational Social Systems, vol. 11, no. 4, pp. 5189-5198, Aug. 2024, doi: 10.1109/TCSS.2023.3309704.

https://ieeexplore.ieee.org/document/10241287

[15]M. A. Wani et al., "Toxic Fake News Detection and Classification for Combating COVID-19 Misinformation," in IEEE Transactions on Computational Social Systems, vol. 11, no. 4, pp. 5101-5118, Aug. 2024, doi: 10.1109/TCSS.2023.3276764.

https://ieeexplore.ieee.org/abstract/document/10151899

Fake News Detection with Dynamic Model Updates Based on Classifier Comparison

Maitreyee Jadhav	Mansi Patil	Preshika Giri	Vinay More
Dept. of Comp Engg	Dept. of Comp Engg	Dept. of Comp Engg	Dept. of Comp Engg
MITWPU	MITWPU	MITWPU	MITWPU

INTRODUCTION

The proliferation of digital platforms has led to an unprecedented rise in the dissemination of fake news, resulting in widespread misinformation and public mistrust. Detecting fake news has become critical in maintaining the integrity of information consumed by the public. Existing fake news detection systems primarily focus on static model training, where models are trained on a fixed dataset and remain unchanged over time. However, the dynamic and evolving nature of misinformation demands systems that can adapt continuously.

Currently, fake news detection models suffer from performance degradation as new styles, formats, and types of misinformation emerge. Static models become outdated, leading to reduced accuracy and reliability. Furthermore, there is a noticeable gap in the industry and research: the absence of dynamic model updating mechanisms. To date, minimal efforts have been directed towards dynamically updating fake news detection models to maintain relevance with real-time data.

Addressing this gap, the present study proposes a system for dynamic model updating, wherein the model is periodically retrained with newly collected real and fake news articles. By doing so, the model remains aligned with current trends, improving predictive accuracy and robustness over time. The proposed methodology systematically integrates data collection, preprocessing, model training, evaluation, and dynamic updating to address this pressing research gap.

NEED FOR STUDY

In today's digital age, the widespread circulation of fake news poses a serious threat to public awareness, political stability, and the credibility of online information. With the increasing sophistication of misinformation campaigns, fake news has advanced beyond the detection capabilities of traditional models. Most existing systems rely on static machine learning models that, once deployed, remain unchanged unless manually retrained. Over time, this lack of adaptability results in declining model accuracy and effectiveness due to the evolving nature of language patterns, trending news topics, and misinformation techniques.

One of the primary motivations for this study stems from the rapid evolution of fake news content. As false narratives become more nuanced and data-driven, the need for dynamic and adaptable detection systems becomes crucial. Unfortunately, many current academic and industry solutions fail to address this gap—they neglect the importance of continuous model updating, making them inadequate in real-time detection scenarios.

Furthermore, this study is motivated by the necessity of real-world applicability. A model that is capable of dynamic updates remains more relevant and contextually aware, thus improving its robustness and accuracy over time. Automated retraining also minimizes the manual labor involved in maintaining such systems, saving time and computational resources while ensuring consistent performance. Consequently, there is a clear and pressing need to develop a fake news detection framework that not only performs well initially but also adapts over time through automated updates, making it more resilient and effective in combating misinformation.

RESEARCH OBJECTIVES

1. The primary aim of this research is to bridge the gap in current fake news detection systems by introducing a dynamically updating model. Traditional models often fail to adapt to evolving data, which compromises their performance. This study, therefore, seeks to design a system capable of automatically updating itself with new information, ensuring long-term reliability and accuracy.
2. One of the key objectives is to implement a dynamic fake news detection framework that supports periodic retraining using newly collected data. This includes building a pipeline that automates the process of web scraping for news data, followed by thorough text preprocessing and feature extraction using Term Frequency-Inverse Document Frequency (TF-IDF) techniques. Once the data is prepared, it will be used to train multiple machine learning models, including Support Vector Machine (SVM), Logistic Regression, and Naïve Bayes.
3. The system will be rigorously evaluated using important performance metrics such as accuracy, precision, recall, and F1-score to determine the most effective model. The best-performing classifier will then be integrated into the final framework and retrained on a weekly basis using updated datasets. This ensures that the model consistently adapts to changes in misinformation strategies and maintains a high level of detection accuracy over time.

Literature Review:

Ref	Title	Publication Year	Methodology	Positive points of publication	Gaps in publication work
1.	Detecting and Mitigating the Dissemination of Fake News: Challenges and Future Research Opportunities – Wajiha Shahid, Bahman Jamshidi, Saqib Hakak, Haruna Isah, Wazir Zada Khan, Muhammad Khurram Khan, Kim-Kwang Raymond Choo (Journal)	2024	Here's an even shorter version, focused only on methodology: The paper uses GloVe embeddings with a CNN model (FNDNet) for fake news detection, a CNN-LSTM model for stance detection, and applies feature extraction with PCA and Chi-Square for dimensionality reduction.	The authors addressed most common and critical challenges encountered by other researchers to make an efficient classifier They also addressed solutions to those challenges.	The paper lacks real-time implementation like the authors could have implemented any one proposed model for clear understanding.

2.	Enhancing Fake News Detection by Multi-feature classification – Ahmed Hashim Jawad Almarashy, Mohammad-Reza Feizi-Derakhshi, Pedram Salehpour (Journal)	2023	The paper combines TF-IDF (global features), CNN (spatial features), and BiLSTM (temporal features) using early fusion, and classifies the result with a Fast Learning Network (FLN) for fake news detection.	The authors proposed a model that integrates global, spatial and temporal features of text for enhancing the accuracy. The hybrid approach accurately classifies fake news, even when dealing with datasets having large number of training samples	The model is tested only on English Language dataset.
3.	Real-Time Fake News Detection Using Big DataAnalytics and Deep Neural NetworkMuhammad Babar, Member, IEEE, Awais Ahmad, Member, IEEE,Muhammad Usman Tariq, Member, IEEE, and Sarah Kaleem	2023	The paper uses a hybrid N-gram and LSTM model for fake news detection, deployed on a big data platform for parallel and distributed processing to achieve real-time, accurate classification.	The authors combined DNNs and blockchain for a highly accurate and secure fake news detection system, ensuring source credibility and traceability. They built a real-time detection system using big data analytics, enabling fast, large-scale news analysis with minimal delays.	The model does not focus on language nuances such as sarcasm, satire, and ambiguous phrasing, leading to misclassification.
4.	Fake News Detection Using Enhanced BERT Shadi A. Aljawarneh	2022	Here's the short summary of the methodology: The paper fine-tunes the BERT model on a fake and real news dataset to enhance its detection capabilities, achieving 99.96% accuracy and outperforming other models.	The optimized BERT model improves language understanding, reducing errors from misleading or satirical headlines. 3 Enhanced BERT outperforms traditional models, boosting accuracy and feature extraction for better fake news detection.	It does not address the high computational cost of BERT,making real-time large-scale detection difficult.

5.	Fake News in Virtual Community, Virtual Society, and Metaverse: A Survey Jinxia Wang , Stanislav Makowski, Alan Cieślik, Haibin Lv , Senior Member, IEEE, and Zhihan Lv , Senior Member, IEEE	2023	The paper provides a survey on fake news in virtual communities and the metaverse, analyzing its manifestation in single-modal and multimodal forms, and reviewing detection methods. It also discusses future directions for intelligent detection and information security in these environments.	The authors explore fake news challenges in emerging digital spaces, highlighting new threats in virtual societies and the metaverse. The paper provides a comprehensive comparison of existing fake news detection techniques, identifying key gaps and future research directions.	The study provides a broad survey of fake news in virtual environments but lacks practical implementation strategies for real-time detection and mitigation. Solution: Developing AI-driven real-time monitoring systems tailored for virtual communities and the metaverse can help detect and counter fake news more effectively.

Gap Areas:

The current research lacks the feature of dynamic model updating, so we are proposing a system for dynamic model updating, that is updating the model on regular intervals to keep it up to date with current news articles to produce better prediction outcomes.

Methodology:

The project will employ a systematic methodology that includes collection of data, preprocessing, training models, evaluation, and taking the best model into consideration for dynamic model updating. We are addressing the noted research gaps of dynamic model updating in the methodology. The proposed system architecture is shown in Fig. 1.

A. Data Collection

We are collecting real news data from legitimate news websites through web scraping and labelling them as 0. The fake news is scraped through some legitimate websites that have a category for identified fake news, the rest of the fake news are created by using any GenAI model, prompting it to create a paraphrased fake version of some real news articles. The fake news articles data is labelled as 1. The details that we are scraping from news websites are: article title, description and the date on which the article was scraped.

Sr no.	Feature name	Description
1	title	Title if the news article
2	text	Description if the news article
3	label	Label of the news article; fake: 1 and real: 0
4	date	Date on which the article was web scrapped

Table 1. Dataset Features

B. Text Preprocessing

To prepare textual data for analysis, a preprocessing function is developed, which involves removing stop-words, tokenizing text, converting to lowercase, eliminating non alphabetic characters, and applying stemming or lemmatization. The dataset that we have created goes through a pre-processing function.

C. Feature extraction

Feature extraction from textual data can be accomplished by using word embeddings or TF-IDF vectorization (Term Frequency-Inverse Document Frequency). By converting text into numerical attributes, we can determine the semantic content and word importance of each document. In our implementation, we utilized the TfidfVectorizer from the scikit-learn library, configuring it with specific parameters to enhance feature extraction:

- N-gram Range: Set to (1, 2) to capture both unigrams and bigrams, allowing the model to consider individual words and pairs of consecutive words, thereby incorporating some contextual information.
- Maximum Document Frequency (max_df): Set to 0.9 to exclude terms that appear in more than 90% of the documents, as such terms are likely to be non-informative and may not contribute to distinguishing between documents.
- Minimum Document Frequency (min_df): Set to 2 to ignore terms that appear in fewer than two documents, reducing the impact of rare terms that may not be relevant for the analysis.
- The vectorizer was fitted to the corpus and the textual data was transformed into a TF-IDF weighted term-document matrix, which served as the input for subsequent analytical processes.

D. Model Training

The pre-processed features are divided into training and testing groups using a suitable splitting technique, such as a random split or cross-validation. We have utilized random split. When selecting a machine learning model, the goals of the research and the characteristics of the dataset are taken into account. We have trained the data on three ML models, Support Vector Machine (SVM), Logistic Regression (LR) and Naïve Bayes (NB).

Support Vector Machine (SVM) is an effective classification method that searches for the best hyperplane in a multidimensional space to separate different classes. It is robust against outliers and efficiently handles both linear and non-linear decision boundaries through the use of kernel functions. By maximizing the margin between classes, SVM improves generalization on unseen data. Its adaptability to different data types and the ability to control overfitting through regularization parameters make SVM a popular choice for various classification tasks.

Logistic Regression is a widely used statistical method for binary and multiclass classification problems. It models the probability of a data instance belonging to a particular class using a logistic (sigmoid) function. The model estimates the relationship between input features and the log-odds of the outcome, making it both interpretable and computationally efficient. Logistic Regression performs best when the classes are linearly separable and includes regularization techniques to prevent overfitting, allowing it to generalize well to new data.

Naive Bayes is a simple yet powerful probabilistic classification algorithm based on Bayes' Theorem. It assumes that the features are conditionally independent given the class label, which simplifies the computation significantly. Despite its 'naive' assumption, Naive Bayes often performs competitively, particularly in text classification and spam detection tasks. It is highly efficient, requires a small amount of training data, and works well even when the independence assumption is somewhat violated in practice.

After training these models, we calculate the evaluation metrics such as accuracy, precision, recall and f1- score. The model with best accuracy is selected as the final model for predicting fake news.

For dynamic model updating, we are scrapping the real news from websites on a daily basis and appending it to our dataset, and training the model weekly on this newly obtained dataset to improve the accuracy of the model based on the latest news.

Assumptions used:

We are assuming that the news taken from legitimate news websites is real or true.

Result and Analysis:

The study examines the LR, SVM and NB methodologies' performance using our dataset. The models' performance is evaluated using key performance indicators like F1-score, recall, precision, and accuracy. The method with best accuracy is used for dynamic model updating.

It is observed that SVM performs better compared to LR and NB so it is selected as the final model. This SVM model is then trained periodically to increase accuracy for predicting the latest fake news. Table 2, 3, 4 and 5 show the evaluation metrics for SVM such as accuracy, precision, recall and f1-score, for two weeks of data. The model was trained on new collected data once a week. The graphs show the same visualization of evaluation metrics.

The 80-20 train-test split consistently delivered the best performance overall. In Week 1, the model achieved high scores in all metrics, particularly a Recall of 0.9286 and an F1-score of 0.8387, indicating its effectiveness in identifying true positive cases and maintaining a good balance between precision and recall. Week 3 also demonstrated solid and balanced performance across all metrics, suggesting that the model performs reliably when trained on 80% of the data. However, Week 2 under this split showed a noticeable drop in recall, which may point to irregularities or noise in the data collected during that period.

Train Test Split **80-20**	Accuracy	Precision	Recall	F1-score
Week 1	0.7826	0.7647	0.9286	0.8387
Week 2	0.6700	0.7500	0.5660	0.6452

Table 2. Evaluation metrics for SVM with 80-20 train-test split

With the 70-30 split, there was a slight decrease in overall performance compared to the 80-20 split, but the model still showed robust results. In Week 1, the model maintained a high F1-score of 0.7917 and a very strong recall of 0.9500, although precision declined slightly. Week 2 and Week 3 produced relatively balanced values for all metrics, indicating that the model was still capable of learning effectively with 70% of the data, though it benefitted from more training data in the 80-20 case.

Train Test Split **70-30**	Accuracy	Precision	Recall	F1-score

Week 1	0.7143	0.6786	0.9500	0.7917
Week 2	0.6711	0.7183	0.6375	0.6755

Table 3. Evaluation metrics for SVM with 70-30 train-test split

The performance began to degrade more noticeably with the 60-40 split. In Week 1, the recall remained high at 0.9231, but both precision and accuracy dropped, affecting the overall F1-score. Week 2 showed the weakest results under this split, with all four metrics dipping significantly. This suggests that with only 60% of the data for training, the model's ability to generalize decreased. Week 3, while slightly better, still did not reach the performance levels observed with the higher training ratios.

Train Test Split **60-40**	Accuracy	Precision	Recall	F1-score
Week 1	0.6739	0.6486	0.9231	0.7619
Week 2	0.6131	0.6304	0.5743	0.6010

Table 4. Evaluation metrics for SVM with 60-40 train-test split

In the 50-50 split, performance showed a mixed trend. Week 1 surprisingly still showed strong recall and an F1-score close to that of higher split ratios, but as with other splits, Week 2 again showed weaker performance in all metrics. Week 3 showed relatively stable values but was not significantly better than the 60-40 split. This indicates that while the model can function with 50% training data, the trade-off in performance becomes more apparent.

Train Test Split **50-50**	Accuracy	Precision	Recall	F1-score
Week 1	0.7193	0.6818	0.9375	0.7895
Week 2	0.6371	0.6230	0.6333	0.6281

Table 5. Evaluation metrics for SVM with 50-50 train-test split

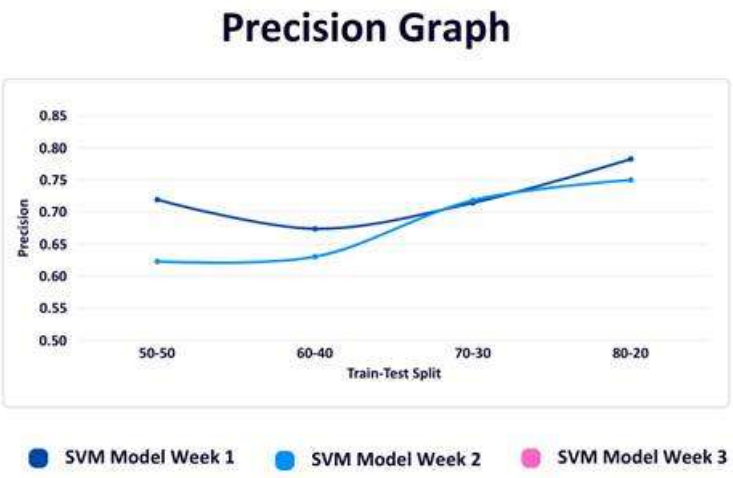

Figure 2

The graph shows SVM model accuracy over three weeks with varying train-test splits. Week 3, trained on the most data, achieved the highest accuracy (0.7913) with an 80-20 split. Week 1 performed well despite less data, reaching 0.7826 accuracy. Week 2 had the lowest and flattest performance, likely due to data imbalance. Overall, more data and better balance improved accuracy over time.

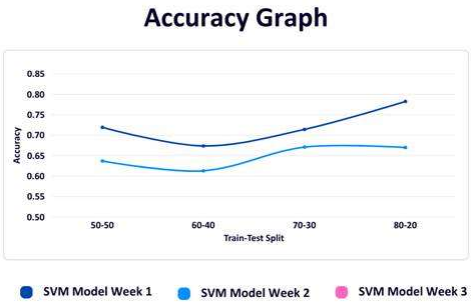

Figure 3

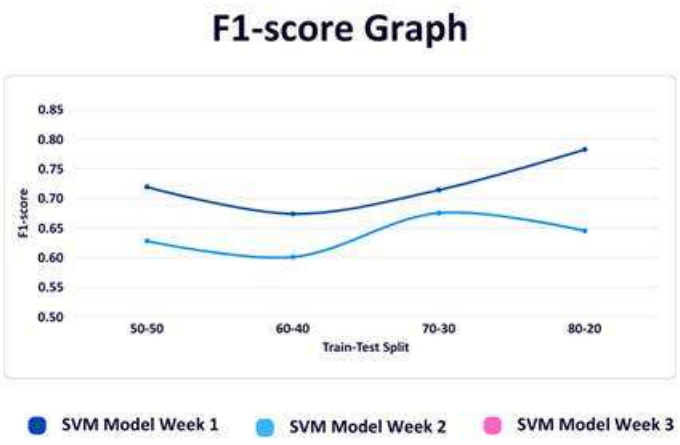

Figure 4

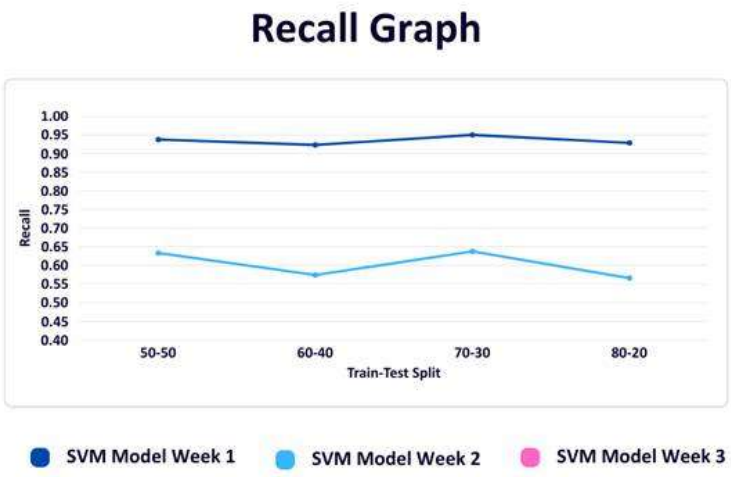

Figure 5

Conclusion

In this paper, we developed a fake news detection system using machine learning techniques such as Logistic Regression, Support Vector Machine (SVM), and Naive Bayes. The system evaluates news article text to determine its authenticity. To ensure ongoing relevance and accuracy, we implemented dynamic model updating by selecting the best-performing algorithm based on accuracy evaluations. The model is retrained using newly collected data obtained through web scraping, allowing it to adapt to emerging trends and linguistic patterns in fake news. This dynamic approach significantly enhances the model's reliability and robustness over time, making it a practical solution for combating the evolving challenge of misinformation.

Future Scope

The current system provides a strong foundation for further enhancements aimed at improving the adaptability and effectiveness of fake news detection. One potential direction is the automation of model selection, wherein the optimal algorithm can be dynamically chosen based on comprehensive performance metrics such as precision, recall, and F1-score, rather than accuracy alone. Furthermore, expanding the scope of web scraping to incorporate diverse data sources, multilingual content, and multimedia formats, such as images and videos, can substantially enrich the training dataset and enhance the model's generalization capabilities. The integration of advanced deep learning architectures, including Long Short-Term Memory (LSTM) networks and transformer-based models like BERT, may also contribute to improved contextual understanding and classification accuracy.