Fake News Detection using Sentimental Analysis And Machine Learning Algorithms

A PROJECT REPORT

Submitted by

M. POORNA PRADEEP REDDY [RA2111026010094]
PENUMURU NAVEEN [RA2111026010085]

Under the Guidance of

(Dr. G. Senthil Kumar)
(Assistant Professor, *Department of COMPUTATIONAL INTELLIGENCE*)

in partial fulfillment of the requirementsfor the degree of
BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
with specialization in ARTIFICIAL INTELLIGENCE and MACHINE LEARNING

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE ANDTECHNOLOGY
KATTANKULATHUR- 603 203

NOVEMBER 2024

Department of Computational Intelligence

SRM Institute of Science & Technology

Own Work Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course : B. Tech/CSE-AIML

Student Name : M. POORNA PRADEEP REDDY, P. Naveen

Registration Number : RA2111026010094, RA2111026010085

Title of Work : Fake News Detection using Sentimental Analysis And Machine Learning Algorithms

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook /University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 18CSP107L - Minor Project [18CSP108L- Internship] report titled "**Fake News Detection using Sentimental Analysis And Machine Learning Algorithms**" is the Bonafide work of "**M. POORNA PRADEEP REDDY [RA2111026010094], PENUMURU NAVEEN [RA2111026010085]**" who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. G. Senthil Kumar

SUPERVISOR
Assistant Professor,
Department of
COMPUTATIONAL
INTELLIGENCE

SIGNATURE

DR. R. ANNIE UTHRA

PROFESSOR &HEAD
DEPARTMENT OF
COMPUTATIONAL
INTELLIGENCE

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. T. V. Gopal ,** Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Com puting, SRM Institute of Science and Technology, for her support throughout the project work.

We encompass our sincere thanks to , **Dr. M. Pushpalatha**, Professor and Associate Chairperson, School of Computing and **Dr. C.Lakshmi,** Professor and Associate Chairperson, School of Computing, SRM Institute of Science and Technology, for their invaluable support.

We are incredibly grateful to our Head of the Department, **Dr. R. Annie Uthra**, Professor, Department of Computational Intelligence, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinators, Panel Head and Panel Members, Department of Computational Intelligence, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. S. Amudha**, Department of Computational Intelligence, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr. G. Senthil Kumar** ,Department of Computational Intelligence, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his / her mentorship. He / She provided us with the freedom and support to explore the research topics of our interest. His / Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff and students of Computational Intelligence, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement.

Authors

ABSTRACT

In contemporary society, digital news plays a pivotal role in providing the general public with crucial information and acting as a conduit for communication and education on ongoing events. The widespread transition from traditional print and broadcast media to internet-based sources has led to significant support for digital news. However, it faces challenges, particularly the proliferation of fake news and misinformation, compounded by algorithm-driven personalization resulting in a dearth of diversity and balance in information. Addressing these issues is crucial to uphold the credibility and reliability of digital news platforms. To address such challenges, our paper proposes a three-tiered strategy. Initially, we focus on collecting, analysing, and categorizing news data using machine learning techniques, namely K Nearest Neighbors (KNN), Support Vector Machine (SVM), Logistic Regression (LR), and Naive Bayes (NB).

These algorithms, with their distinct strengths, collectively enhance the accuracy of classification and thematic identification of news articles. Through this approach, we aim to fortify the robustness and effectiveness of our analysis. Our research aims to provide a comprehensive solution to the challenges posed by digital news. By employing advanced machine learning techniques, we seek to process and analyse a vast array of digital news articles, classifying them into distinct categories. Preliminary results reveal prevalent news themes such as Crime, Cure and Treatment, Economy, Communal, and Entertainment across India. Additionally, our Fake News Detection Model demonstrates promising accuracy, with a specific focus on the communal theme. The sentiment analysis model further contributes to a nuanced understanding of news articles. Through these efforts, we anticipate providing valuable insights to foster a more reliable and diverse digital news landscape.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

KNN	K-Nearest Neighbors
SVM	Support Vector Machine
LR	Logistic Regression
NB	Naïve Bayes
RF	Random Forest
TF-IDF	Term Frequency Inverse Document Frequency
PR	Precision Recall

CHAPTER 1

INTRODUCTION

1.1 Introduction

Fake news detection is essential in today's digital age, where the rapid dissemination of information is facilitated by social media and online platforms. Misinformation and fake news can have severe consequences, including shaping public perception inaccurately, inciting social unrest, and influencing political events. Consequently, researchers and technologists are focusing on developing sophisticated methods for identifying and filtering fake news to maintain the integrity of information shared online.

Machine learning algorithms play a pivotal role in this effort by analyzing large volumes of news content and identifying patterns associated with misleading or false information. For instance, Support Vector Machines (SVM) can effectively classify content by drawing boundaries between genuine and fake news articles based on features like word choice, sentence structure, and source credibility. Naive Bayes models, which operate on probabilistic assumptions, analyze the likelihood of certain keywords and phrases appearing in fake news, while Logistic Regression offers insights into the probability of a news item being categorized as fake, enabling data-driven decisions.

Sentiment analysis adds another layer of intelligence by examining the emotional tone or sentiment of a news article. Fake news often uses emotionally charged language—positive, negative, or polarizing—to capture readers' attention and provoke reactions. By detecting unusual sentiment patterns, such as excessive negativity in politically charged topics or exaggerated positivity in promotional content, sentiment analysis can serve as a red flag for potentially biased or misleading information. Moreover, combining sentiment scores with machine learning classification creates a hybrid model that not only assesses the factual accuracy of the content but also considers the psychological impact it may have on readers.

The integration of these technologies has practical applications beyond simply flagging false content. For instance, media organizations can use these models to monitor trending topics and identify stories that may require fact-checking, while social media platforms can implement automated checks to reduce the spread of suspicious articles. Policymakers and researchers can also benefit by gaining insights into public sentiment around key issues, which can inform public communications and policy decisions. As digital information continues to grow, enhancing fake news detection with machine learning and sentiment analysis provides a powerful toolset to support a more reliable, transparent, and balanced media landscape.

1.2 Motivation

The motivation for using sentiment analysis and machine learning algorithms in fake news detection arises from the urgent need to address the growing impact of misinformation on society. With the rise of digital and social media, fake news can now reach a global audience almost instantaneously, affecting public opinion, trust in institutions, and decision-making. Traditional methods of fact-checking and verification are often too slow and resource-intensive to keep pace with the sheer volume of content generated online. Machine learning and sentiment analysis offer scalable, efficient solutions by automating the process of detecting deceptive patterns in text and identifying language that may indicate manipulation or bias. Sentiment analysis, in particular, adds a crucial dimension by capturing the emotional undertones in news articles; fake news often exploits strong emotions, like fear or outrage, to increase its reach and influence. By combining machine learning models that assess linguistic and statistical patterns with sentiment analysis that evaluates emotional cues, we can develop a more comprehensive system that not only classifies news as fake or genuine but also helps in understanding its intent and potential impact. This approach aims to empower users, media organizations, and policymakers with tools to mitigate the harmful effects of misinformation, ultimately fostering a more trustworthy and resilient information ecosystem.

1.3 Sustainable Development Goal of the Project

The fake news detection project also indirectly supports other Sustainable Development Goals by promoting informed decision-making across various sectors, from health (SDG 3) to climate action (SDG 13) and education (SDG 4). Misinformation can hinder progress in many areas, especially in public health, where false information about vaccines or treatments can put lives at risk, or in environmental action, where climate-related misinformation can delay necessary policy interventions. By strengthening the reliability of information, this project aids individuals, communities, and policymakers in accessing accurate data essential for making sustainable and impactful decisions across different domains.

Machine learning and sentiment analysis add rigor to this project by enabling real-time, scalable solutions that are crucial for managing the volume of data generated online. Fake news often circulates quickly, exploiting emotional appeals to maximize engagement, and manual fact-checking efforts struggle to keep up. By automating fake news detection and examining emotional cues, the project not only speeds up detection but also reduces the spread of harmful misinformation before it can cause significant impact. Furthermore, it empowers users with credible information, fostering digital literacy and helping people develop skills to critically assess news content. This aligns with SDG 4, "Quality Education," by promoting knowledge and awareness about online content evaluation, thus building a more media-literate society.

Moreover, reliable fake news detection supports SDG 17, "Partnerships for the Goals," by encouraging collaboration among media platforms, technology providers, governments, and civil society to address misinformation. A credible and trusted information ecosystem allows diverse stakeholders to work together more effectively, whether on policy advocacy, community engagement, or cross-border initiatives. By contributing to a shared standard of information credibility, the project strengthens partnerships essential for advancing sustainable development goals across the board.

CHAPTER 2

LITERATURE SURVEY

1. Aímeur, Amri, and Brassard (2023)

Title: “Fake news disinformation and misinformation in social media: a review”

Journal: Social Network Analysis and Mining, vol. 13, no. 1, pp. 30

Year: 2023

Methodology:

This research paper discuss the problem of fake news spreading on social media, where it can quickly reach and mislead large numbers of people. They divide fake news into two types: 'misinformation', which is spread unintentionally, and 'disinformation', which is deliberately false and meant to deceive. The authors look at various ways to detect fake news, including machine learning algorithms, language analysis, and studying how information spreads on social networks. Techniques like Support Vector Machines (SVM) and Naive Bayes classifiers, as well as sentiment analysis, help identify fake news by detecting patterns in language or emotional content that can indicate misleading information. However, they note that these systems are not perfect, often needing large amounts of data and struggling with detecting fake news in real-time.

The review suggests that tackling fake news will require a combined approach: better technology for detection, stricter rules for social media platforms, and more education for users to recognize fake news. Although putting this solution into practice may be costly and complex, especially in adapting to different cultures and languages, the authors believe it is necessary for an effective response. By taking these combined steps, the authors argue, we can work toward reducing the impact of fake news on social media.

Technology Used:

The several technologies are discussed for detecting fake news on social media. The authors highlight machine learning algorithms, such as Support Vector Machines and Naive Bayes, which classify news articles as true or false based on their content. They also use natural language processing techniques to analyze the language in articles, including sentiment analysis to gauge emotional tone. Additionally, social network analysis helps track how news spreads among users, while data mining tools sift through large amounts of social media data to identify patterns. Visualization tools are used to present their findings clearly, helping to understand the dynamics of misinformation.

Pros:

- Comprehensive Analysis of Current Techniques.
- Emphasis on Hybrid Solutions.
- Identification of Gaps and Future Research.

Cons:

- Limited Focus on Regional and Cultural Factors.

- Reliance on Secondary Data.
- Challenges in Implementing Recommendations.

2. Granik and Mesyura (2017)

Title: Fake News Detection Using Naive Bayes Classifier.

Conference: 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)

Year: 2017

Methodology:

This research paper propose a method for detecting fake news using the Naive Bayes classifier, a straightforward and effective machine learning algorithm. Their methodology involves training the Naive Bayes classifier on labeled datasets containing examples of both fake and legitimate news. By analyzing word frequencies and patterns within these datasets, the classifier learns to identify distinguishing characteristics of fake news, such as specific word choices, tone, and phrasing that differ from credible news sources. This allows it to assess the likelihood that new, unseen articles are fake. The researchers chose the Naive Bayes classifier because of its simplicity and relatively low computational cost, making it suitable for real-time detection. Through experimental tests, they demonstrate that the classifier performs well in distinguishing fake from real news, proving its potential as a fast and efficient tool for combating misinformation.

Technology Used:

The main technology used is the Naive Bayes algorithm, a machine learning model that classifies news articles as fake or legitimate based on word frequency. They preprocess the data by cleaning and preparing the text, using techniques like removing unnecessary words and converting the text into numerical forms through methods like Bag of Words or TF-IDF. The researchers likely used programming languages like Python or R, along with libraries that help implement the Naive Bayes algorithm. They also measure the classifier's performance using metrics like accuracy and precision to evaluate how well it detects fake news.

Pros:

- Simplicity and Speed increases.
- Effective Performance.
- Increases Scalability.

Cons:

- Assumption of Independence.
- Sensitivity to Training Data Quality.
- Limited Contextual Understanding.

3. Vosoughi, Roy, and Aral (2018)

Title: The spread of true and false news online.

Journal: Science, vol. 359, no. 6380, pp. 1146-1151

Year: March 2018

Methodology:

This research paper investigates the dynamics of true and false news spread on online platforms using a comprehensive methodology that combines data analysis with statistical modeling. They analyze a vast dataset of tweets, specifically focusing on information shared on Twitter over a period of several years. The researchers categorize news stories as true or false based on fact-checking organizations and credible sources. They employ statistical techniques to assess the spread of these news stories, examining factors such as the number of retweets, likes, and replies to gauge engagement levels. The methodology includes modeling the rate of dissemination and the demographic factors influencing the sharing of true and false news. By comparing the spread patterns of true versus false news, the researchers aim to identify key differences in how each type of information circulates within social networks. Their analysis provides insights into the mechanisms that lead to the rapid propagation of false news and the social dynamics that contribute to its reach.

Technology Used:

The researchers used various technologies to analyze news on Twitter. They collected a large dataset of tweets using Twitter's API to track user interactions like retweets and likes over several years. They employed natural language processing to help categorize news stories as true or false and used statistical analysis software to examine the spread patterns. Machine learning algorithms may have also been used for better classification, and visualization tools helped present their findings clearly. This approach provided important insights into how news spreads on social media.

Pros:

- Comprehensive Data Analysis.
- Identification of Key Factors.
- Insights into Social Dynamics.

Cons:

- Reliance on Fact-Checking Sources.
- Platform Limitation.
- Dynamic Nature of Social Media.

4. **Jain et al. (2019)**

Title: A smart system for fake news detection using machine learning.

Conference: 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)

Year: 2019

Methodology:

The methodology for creating a smart system to detect fake news using machine learning involves several important steps. First, a large dataset of news articles is collected from different sources, including social media and news websites, ensuring it contains examples of both true and false news. Next, the data is cleaned by removing irrelevant parts, like HTML tags and common words that don't add much meaning. Then, the text is transformed into numerical formats using techniques like Bag of Words or TF-IDF to highlight important words and their frequencies. After this, various machine learning algorithms, such as Naive Bayes or Support Vector Machines, are trained on the dataset to learn how to identify fake news. The models are then tested to see how well they perform using metrics like accuracy and precision. Finally, the best model is set up in a smart system that can analyze new articles in real time and provide users with a rating of whether the news is likely true or false. The system is designed to learn continuously, improving its detection ability as it processes more data over time.

Technologies Used:

There are several key technologies are used. Machine learning algorithms like Naive Bayes and Support Vector Machines (SVM) help classify news articles as true or false based on their content. Natural Language Processing (NLP) techniques analyze the text to identify important features. Data preprocessing tools clean the data by removing unnecessary information, while feature extraction methods like Bag of Words and TF-IDF turn the text into numbers that the algorithms can understand. Finally, cloud computing or server technologies are often used to deploy the system, allowing it to analyze news articles in real time and handle large amounts of data.

Pros:

- Improved Accuracy.
- Real-Time Analysis.
- Improves Scalability.

Cons:

- Dependence on Quality Data.
- No reliable Complexity.
- False Positives and Negatives.

5. **Ahmed, Traore, and Saad (2017)**

Title: Detection of online fake news using n-gram analysis and machine learning techniques

Conference: Proceedings of the International Conference on Intelligent Secure and Dependable Systems in Distributed and Cloud Environments

Pages: 127-138

Year: 2017

Methodology:

The methodology for detecting fake news using n-gram analysis and machine learning involves several steps. First, a dataset containing both fake and real news articles is collected. Then, the text is cleaned by removing unwanted parts like HTML tags and common words. Next, n-gram analysis is applied, breaking the text into sequences of n words (like pairs of words), which helps capture the context. After that, the n-grams are turned into numerical data that machine learning algorithms can use. Various algorithms, such as Naive Bayes and Support Vector Machines, are trained to distinguish between fake and real news based on this data. Finally, the best model is used in a system that can classify new articles as fake or true, improving the detection of misinformation.

Technologies used:

In the detection of online fake news using n-gram analysis and machine learning, several important technologies are used. N-gram analysis breaks down text into sequences of words, capturing the context of the articles. Natural Language Processing (NLP) techniques clean and prepare the text by removing unnecessary elements. Machine learning algorithms like Naive Bayes and Support Vector Machines classify the news articles based on features extracted from the n-grams. Additionally, tools like Pandas and NumPy in Python help manage and process the data efficiently, while evaluation tools check the accuracy of the algorithms to ensure effective fake news detection.

Pros:

- Contextual Understanding.
- Automated Classification.
- Strong Adaptability.

Cons:

- Dependence on Quality Data.
- Complexity of Implementation.
- False Positives and Negatives.

2.1 Limitations Identified from Literature Survey

The limitations identified in the studies on fake news detection reveal several challenges faced by current methodologies and technologies. One significant limitation is the reliance on high-quality, labeled datasets for training machine learning models. If these datasets are biased or lack sufficient examples of fake news, the models may not generalize well, leading to poor performance in real-world applications. Additionally, many algorithms, including Naive Bayes and others utilized in these studies, are susceptible to overfitting, where they excel on training data but fail to accurately classify unseen information. The ever-evolving nature of language and misinformation strategies also presents a challenge, as models trained on outdated datasets may struggle to identify new forms of fake news or adapt to shifts in user behavior on social media platforms. Moreover, the risk of false positives and negatives can erode public trust in detection systems; legitimate articles may be incorrectly marked as fake, while deceptive content may escape detection altogether.

The complexity of natural language adds another layer of difficulty, as nuances, sarcasm, and cultural references can be misinterpreted by algorithms. The use of n-gram analysis, although beneficial, may overlook broader contextual elements such as sentiment and intent, leading to inaccurate classifications. Additionally, the computational demands of processing large datasets and training complex models can be prohibitive for smaller organizations, hindering their ability to deploy effective solutions. Scalability also poses a challenge, as many current approaches may struggle to maintain performance when dealing with the vast volumes of rapidly generated news content on social media. Furthermore, the interdisciplinary nature of fake news detection often means that approaches may lack integration of insights from fields like psychology and sociology, which could enhance detection strategies.

User interaction and feedback mechanisms are often underutilized, missing opportunities for crowdsourced input and public engagement, which are crucial for fostering awareness about critical media consumption. Additionally, the potential for manipulation of detection systems by malicious actors highlights the need for continuous improvement and adaptation of methodologies. In summary, while advancements in machine learning and n-gram analysis show promise for fake news detection, addressing these limitations requires a multi-faceted approach that encompasses technical improvements, a deeper understanding of human behavior, user engagement, and ethical implications of algorithmic decision-making. Collaboration across disciplines, ongoing research, and active participation from users and stakeholders will be essential in developing more robust and effective systems to combat the spread of misinformation in today's digital landscape.

2.2 Research Objectives

The research objectives for predicting fake news using sentiment analysis and machine learning algorithms extend beyond merely improving detection accuracy; they aim to create a comprehensive, multifaceted framework that addresses the complexities of misinformation in the digital age. A crucial objective is to refine and enhance the predictive capabilities of existing models, allowing for a more nuanced understanding of the factors that contribute to the propagation of fake news. This includes examining various features such as language patterns, article structures, and the contextual relevance of the content, thereby enabling the algorithms to identify subtle indicators of deception that might be overlooked by traditional methods.

Furthermore, the implementation of a diverse array of machine learning algorithms allows for comparative analysis, which not only assesses their effectiveness in classifying digital news articles but also informs the development of hybrid models that combine the strengths of different techniques. By iterating on these models, the research seeks to create a system that can adapt to evolving trends in misinformation, thus maintaining its effectiveness as new types of fake news emerge. This adaptability is essential, given the rapidly changing landscape of social media and the internet, where the nature of misinformation can shift dramatically over time.

In addition to improving accuracy and adaptability, another key objective is to provide actionable insights derived from the analytical processes. This involves generating reports and visualizations that highlight the characteristics of fake news, helping stakeholders—including news organizations, policymakers, and the general public—understand the dynamics of misinformation and its impact on society. By disseminating these findings, the research aims to foster a more informed public discourse around news consumption and media literacy, ultimately empowering users to critically assess the information they encounter.

Moreover, the focus on enhancing sentiment analysis represents a significant advancement in detecting fake news, as emotional manipulation is often a tactic employed in misleading narratives. By improving sentiment detection capabilities, the research aims to differentiate between content designed to provoke emotional responses and objective reporting, thereby offering a deeper layer of analysis that can enhance the accuracy of fake news detection. This dual focus on textual and emotional cues not only broadens the scope of what can be detected but also aligns with current trends in how news is consumed and shared, particularly in environments like social media where emotional engagement is a key driver of content virality.

Overall, the overarching goal of these research objectives is to contribute to the development of intelligent systems that can autonomously identify and mitigate the effects of fake news, thereby supporting the integrity of information in the digital realm. By bridging the gap between technical advancements and practical applications, this research aspires to influence policy, encourage responsible media practices, and ultimately cultivate a digital ecosystem where credible news can thrive amid the challenges posed by misinformation.

2.3 Product Backlog

S.No	**USER STORIES OF FAKE NEWS IDENTIFICATION**
US 1.	As a user, I want to efficiently gather and organize relevant datasets from various sources so that I can prepare the data for further analysis and model training.
US 2.	As user, I want to Extract and engineer relevant features from the collected data.
US 3.	As a user, I want to search for news articles, sources, and other users on the platform to see if they are credible, using fake news detection.
US 4.	As a user, I want to validate the accuracy of the machine learning models on test data so that I can ensure the reliability of the fake news detection system.
US 5.	As a user, I want to analyze the effectiveness of different algorithms in detecting fake news so that I can provide future research guidance.
US 6.	As a user, I want to receive detailed insights on the credibility of news articles based on the analysis so that I can make informed decisions.
US 7.	As a user, I want to monitor the sources of news articles flagged as fake so that I can take action to block or warn about unreliable sources.
US 8.	As a user, I want to continuously update and improve the training datasets with new and diverse examples so that the detection system remains accurate over time.

Table 2.1 Product Backlog

The product backlog of Fake News Identification was configured using the MS planner Agile Board which is represented in the following Figure 2.1. The Product Backlog consists of the complete user stories of Fake News Identification.

Each user story consists of necessary parameters like MoSCoW prioritization, Functional and non-functional parameters, detailed acceptance criteria with linked tasks.

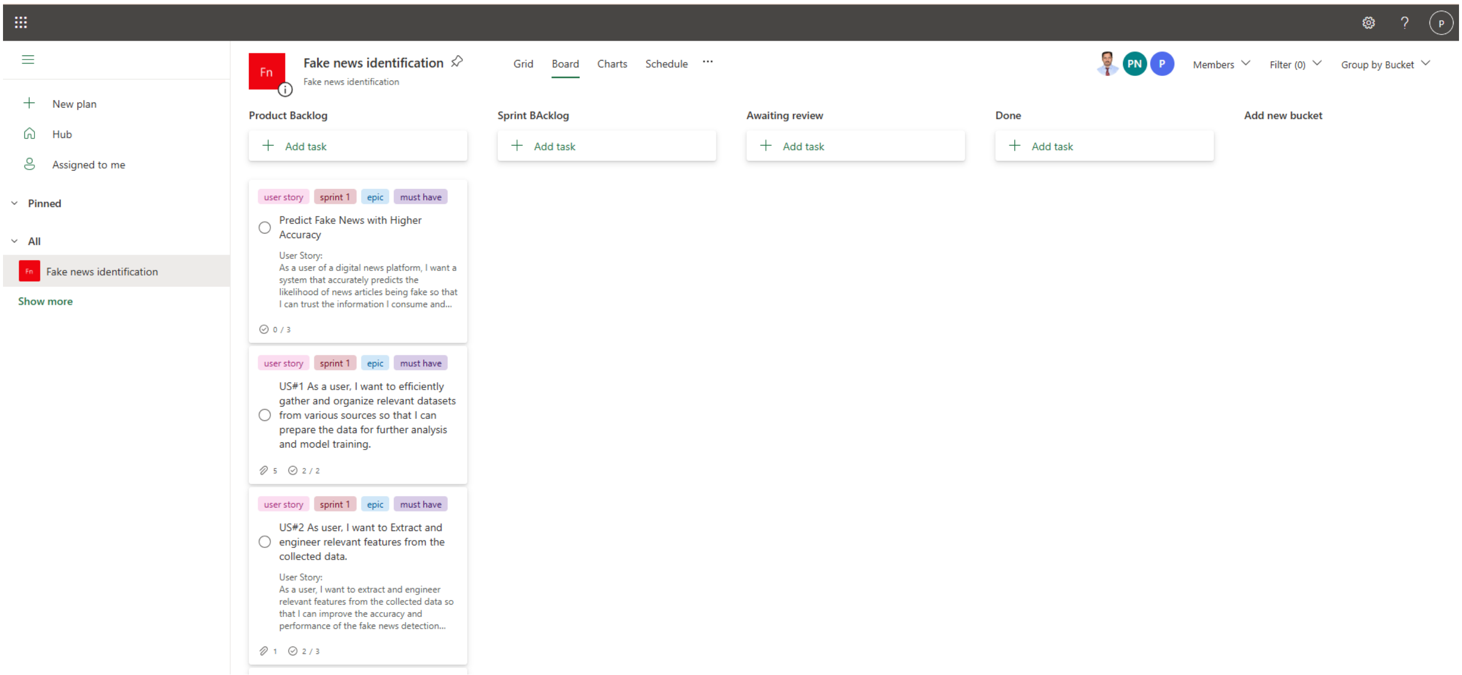


Figure 2.1 MS Planner Board of Fake News Identification

CHAPTER 3

SPRINT PLANNING AND EXECUTION

3.1 SPRINT 1

3.1.1 OBJECTIVES WITH USER STORIES OF SPRINT 1:

The Goal of the first sprint is to efficiently gather and organize relevant datasets and extract the relevant features from the collected data.

Table 3.1 Detailed User Stories of sprint 1

S.NO	Detailed User Stories
US 1	As a user, I want to efficiently gather and organize relevant datasets from various sources so that I can prepare the data for further analysis and model training.
US 2	As user, I want to Extract and engineer relevant features from the collected data.
US 3	As a user, I want to search for news articles, sources, and other users on the platform to see if they are credible, using fake news detection.

Planner Board representation of user stories are mentioned below figures 3.1,3.2 and 3.3

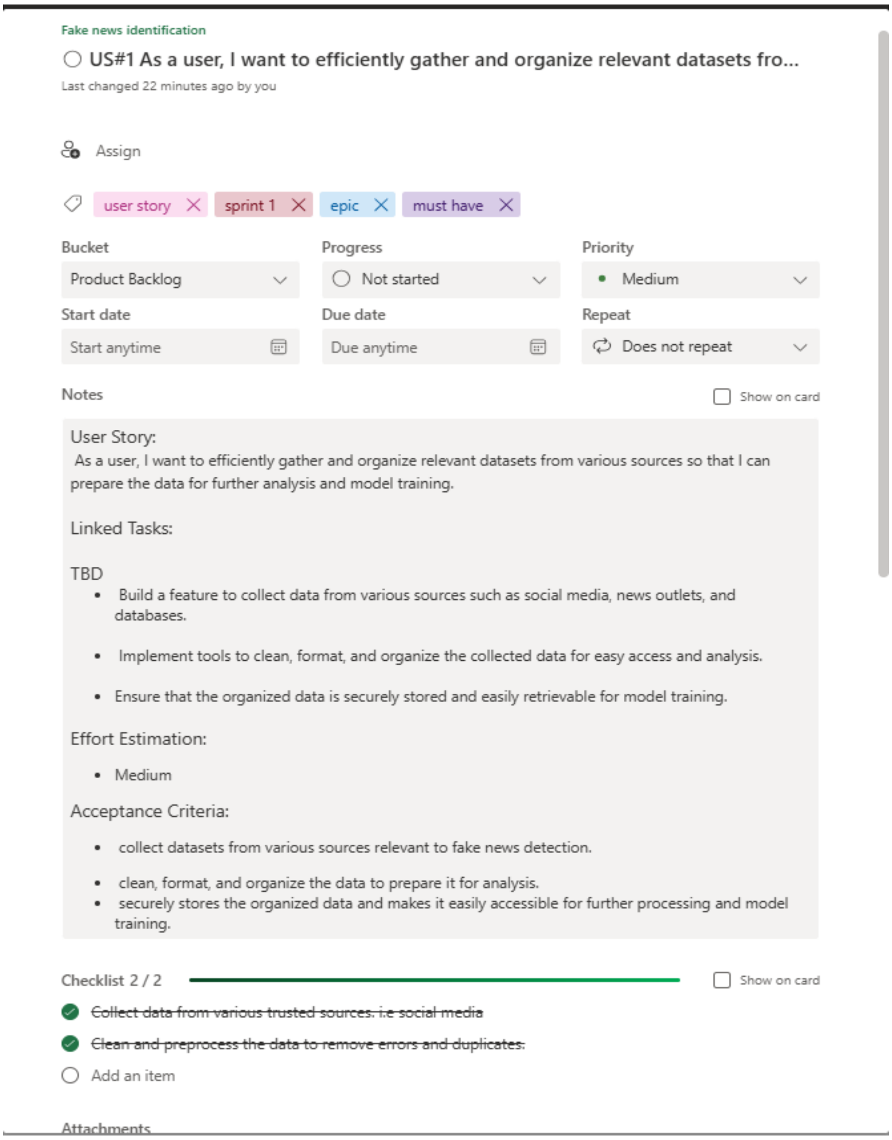


Figure 3.1 user story for collecting the datasets

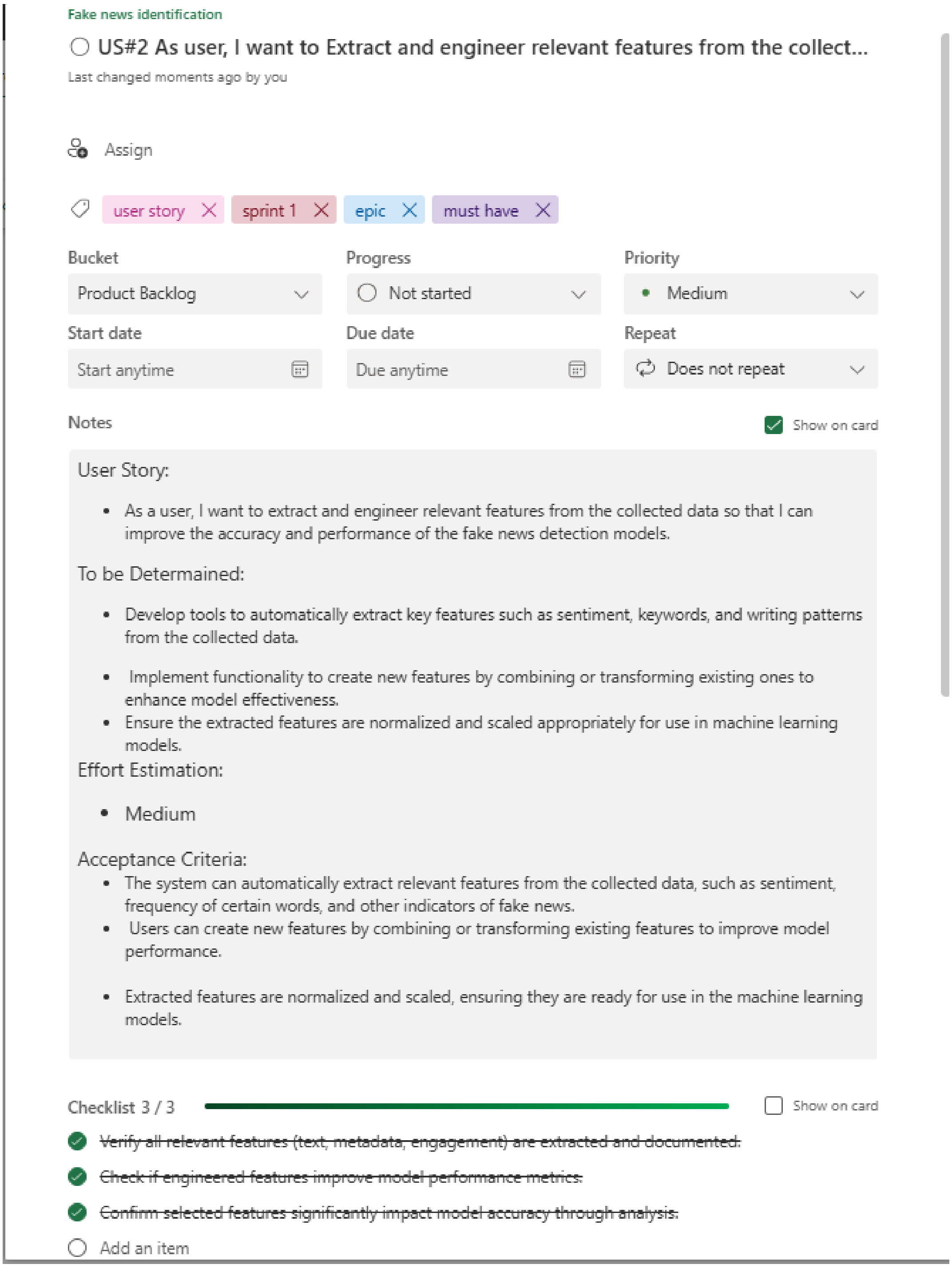


Figure 3.2 User story for feature extraction

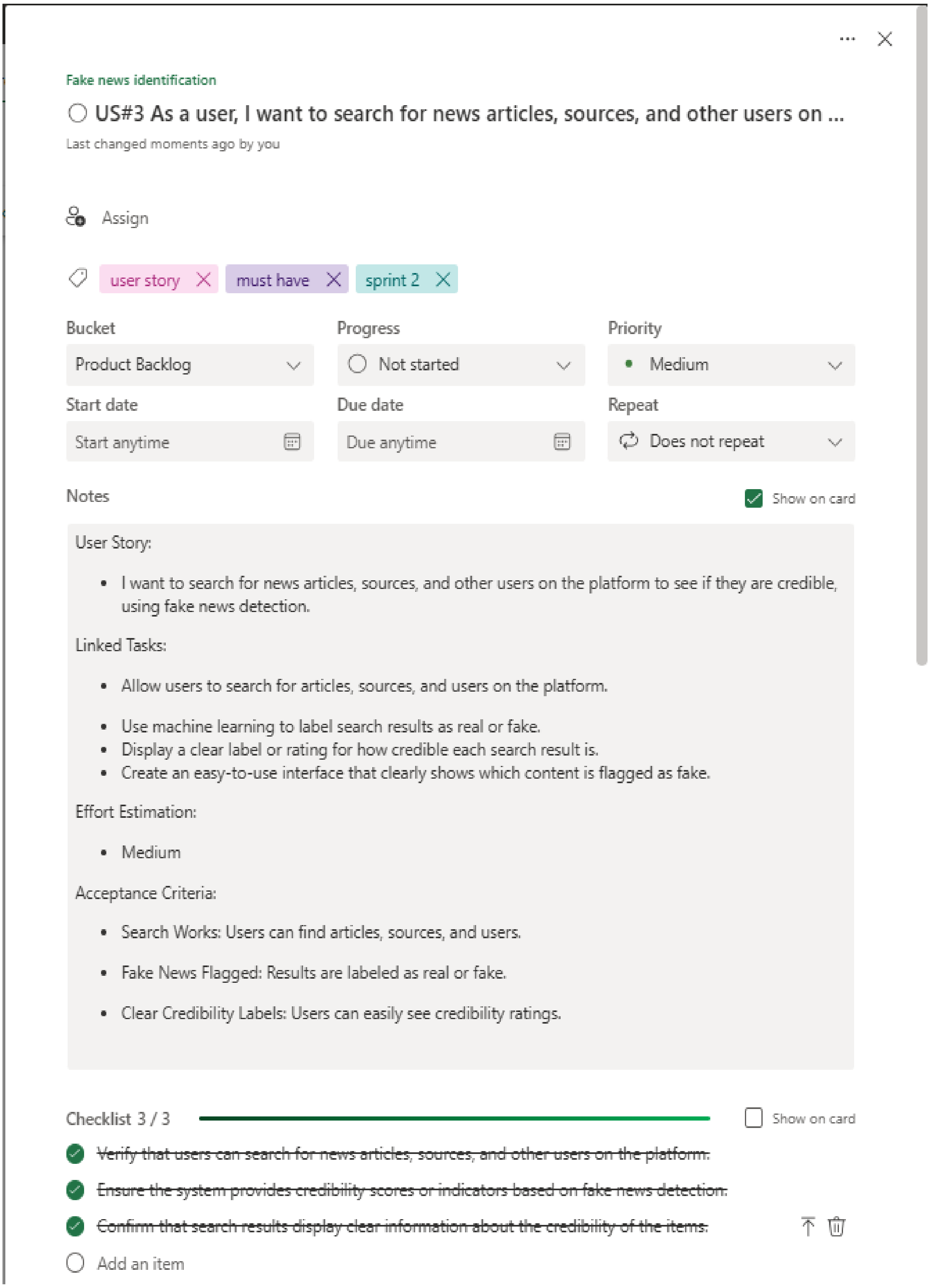


Figure 3.3 user story for to search news article platform to see if they are credible

3.1.2 FUNCTIONAL DOCUMENT

3.1.2.1 Introduction:

In today's digital world, fake news spreads quickly, especially on social media. This makes it difficult for users to determine which news is reliable. A fake news detection system helps users verify if content is trustworthy by analyzing news articles, sources, and users who share content. For better reliability, these systems need machine learning (ML) models that are accurate in identifying fake news. By allowing users to check ML models' accuracy with test data, the system aims to provide dependable results, helping users make informed decisions and supporting a more trustworthy online environment.

3.1.2.2 Product Goal:

The primary goal is to create a reliable fake news detection system that helps users evaluate the trustworthiness of online content. By making it easy to verify news articles and sources, the system reduces the spread of misinformation. This system also lets users validate ML model accuracy, ensuring the results are accurate. Ultimately, the goal is to build a trusted platform where users can safely navigate digital content and detect misinformation effectively.

3.1.2.3 Demography:

Users: Everyday consumers, journalists, researchers, students, organizations, fact-checkers, and moderators.
Location: The platform serves a global audience but is especially useful in regions with high internet usage, like North America, Europe, and rapidly growing online regions in Asia and South America.

3.1.2.4 Business Processes:

1. User Interaction and Search:
Users can search for articles, sources, and users to check their credibility. The system provides credibility scores and flags, allowing users to review results and give feedback for improvements.

2. Machine Learning Model Validation:
Users, such as researchers, can test the ML models' accuracy on updated datasets, reviewing performance metrics like precision and recall. This builds transparency and trust in the system's results.

3. Content Monitoring and Moderation:
The system continuously monitors content for misinformation, scanning articles and posts in real-time and alerting moderators if needed.

4. Data Collection and Model Training:
Data is regularly collected and labeled to train the ML models, ensuring the system stays updated to handle new misinformation trends.

5. User and Business Feedback Loop:
User feedback on search results and detection accuracy, as well as business insights, help improve and adapt the system to users' needs.

3.1.2.5 Features:

1. Credibility Search and Verification:
Users can search for articles, sources, and users to assess their credibility. The system provides credibility scores or flags with explanations.
- User Story: As a user, I want to verify credibility to avoid misinformation.

2. Machine Learning Model Accuracy Validation:
Advanced users can validate the ML models by testing accuracy on provided datasets, reviewing metrics like precision and recall.
- User Story: As a user, I want to validate model accuracy to trust the system's results.

3. Real-time Misinformation Alerts:
Users receive alerts when potential misinformation is detected, allowing them to review content credibility and find reliable alternatives.
- User Story: As a user, I want real-time alerts for misinformation so I can make informed decisions.

3.1.2.6 Authorization Matrix :

Table 3.2 Authorization Matrix

Role	Permissions	Description
Regular User	Search & Verify	Search news, view credibility scores, and get misinformation alerts.
Advanced User	Search, Verify & Model Testing	All Regular User permissions, plus test ML model accuracy.
Moderator	Monitor & Moderate Content	Review flagged content, approve or dismiss misinformation alerts.
Data Scientist	Manage & Train Models	Access to update, test, and improve ML models with new data.
Admin	Full Access	Access to all features, including managing users, models, and content.

3.1.2.7 Assumptions:

1. Reliable Training Data Availability:
Access to current and accurate datasets for training ML models is essential.

2. User Engagement in Verification:
Users are motivated to verify credibility and help prevent misinformation.

3. Continuous Improvement of ML Models:
Regular updates to the ML models will be made based on new data and feedback.

4. Human Oversight:
A team of moderators will review flagged content and handle cases requiring human judgment.

3.1.3 ARCHITECTURE DOCUMENT

3.1.3.1 Application

1. Microservices:

- Architecture: The system should be broken down into microservices, each handling a specific aspect of the fake news detection process.
- Components:
  - Data Ingestion Service: Gathers data from various news sources and social media platforms.
  - Preprocessing Service: Cleans, tokenizes, and normalizes the text data.
  - Sentiment Analysis Service: Analyzes the sentiment of the news articles and social media posts.
  - Fake News Detection Service: Applies machine learning algorithms to predict the likelihood of news being fake.
  - Notification Service: Alerts users or other systems when fake news is detected.
  - API Gateway: Manages requests and routes them to the appropriate microservices.

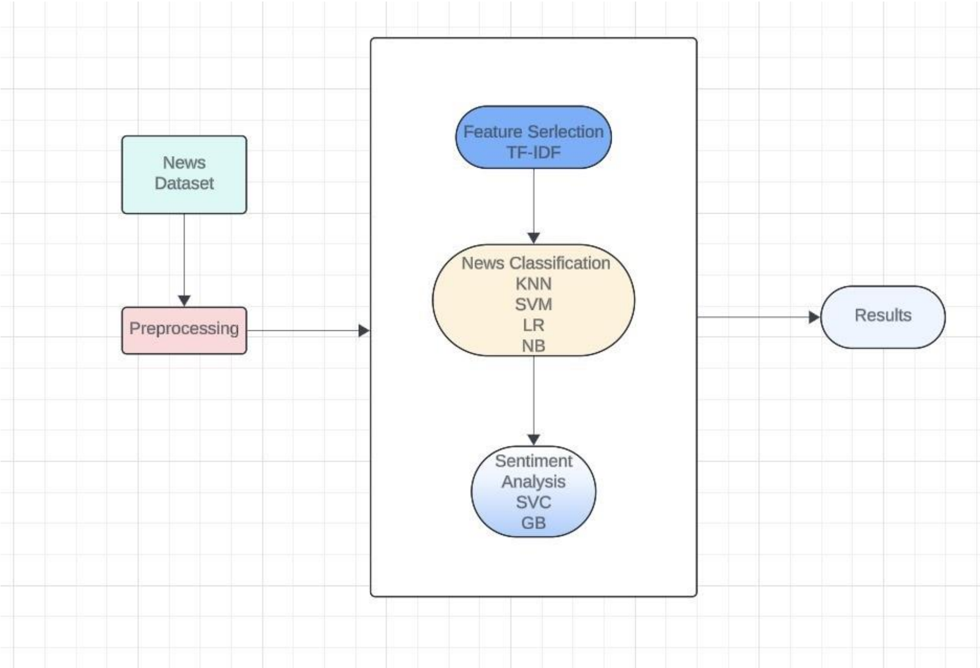
2. **Event-Driven:**

- Events: The system should react to specific events such as new data ingestion, detection of fake news, or updates to the machine learning model.
- Event Handlers:
  - Trigger re-analysis of articles when the model is updated.
  - Trigger notifications when fake news is detected.
  - Scale services dynamically based on the influx of data.

3. **Serverless:**

- Functionality: Use serverless functions for scalable, event-driven processes like:
  - Processing incoming data streams in real-time.
  - Running periodic model updates.
  - Executing lightweight tasks such as sending notifications.
- Advantages:
- Cost efficiency as you only pay for the compute time you use.
- Easy to scale and manage without the need for infrastructure provisioning.

3.1.3.2 System Architecture :



3.1.3.3 Data Exchange Contract:

1. **Frequency of Data Exchanges**:
   - **Real-Time**: Data ingestion from social media platforms and news websites.
   - **Batch**: Periodic updates from data providers, model training data.
   - **On-Demand**: Model retraining or reanalysis of specific articles.

2. **Data Sets**:
   - **Training Data**: Labeled datasets of articles as real or fake.
   - **Sentiment Analysis Data**: Datasets for training sentiment analysis models.
   - **News Articles**: Streamed or batch-loaded news articles from various sources.
   - **Social Media Posts**: Tweets, posts, and comments related to news articles.

3. **Mode of Exchanges**:
   - **API**: RESTful APIs for real-time data ingestion and results retrieval.
   - **File**: Batch files (CSV, JSON) for training data and periodic updates.
   - **Queue**: Message queues (e.g., Kafka, RabbitMQ) for handling event-driven data processing.
   - **Database**: Direct database connections for internal services, especially for data retrieval and model results storage.

3.1.4 OUTCOME OF OBJECTIVE

Preprocessing

Missing Values

```
[4]: df_train.isnull().sum()
```

```
[4]: Class Index    0
     Title          0
     Description    0
     dtype: int64
```

```
[5]: df_test.isnull().sum()
```

```
[5]: Class Index    0
     Title          0
     Description    0
     dtype: int64
```

```
[6]: df=pd.concat([df_train,df_test],axis=0)
     df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 127600 entries, 0 to 7599
Data columns (total 3 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Class Index  127600 non-null  int64
 1   Title        127600 non-null  object
 2   Description  127600 non-null  object
dtypes: int64(1), object(2)
memory usage: 3.9+ MB
```

Figure 3.5 Data Preprocessing

Feature Extraction

```
[5]: from sklearn.feature_extraction.text import TfidfVectorizer
     tfidf_vectorizer = TfidfVectorizer(max_df=0.7,ngram_range=(1,2),stop_words='english')
     X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
     X_test_tfidf = tfidf_vectorizer.transform(X_test)
     print(X_train_tfidf)
```

```
  (0, 74193)	0.1420279039481175
  (0, 357987)	0.1303533551275847
  (0, 23756)	0.15085935920327462
  (0, 137391)	0.15085935920327462
  (0, 409393)	0.15085935920327462
  (0, 377192)	0.15085935920327462
  (0, 357996)	0.14569328905221093
  (0, 1055825)	0.1391848103827418
  (0, 1115369)	0.13686183379705383
  (0, 610439)	0.11974117944958325
  (0, 336491)	0.13169576364599014
  (0, 943965)	0.15085935920327462
  (0, 358008)	0.13686183379705383
  (0, 508713)	0.14569328905221093
  (0, 221828)	0.15085935920327462
  (0, 640514)	0.10015385041391117
  (0, 402231)	0.09317775608036212
  (0, 437312)	0.15085935920327462
  (0, 358000)	0.15085935920327462
  (0, 282641)	0.30171871840654924
  (0, 837855)	0.2406150873100072
  (0, 608664)	0.14569328905221093
  (0, 896382)	0.14569328905221093
  (0, 409502)	0.15085935920327462
  (0, 981276)	0.09440589390724279
  :	:
  (102079, 109937)	0.10760036778126689
  (102079, 1147300)	0.08721610227415928
  (102079, 13132)	0.08627170590606784
  (102079, 378089)	0.10511548672933903
  (102079, 377998)	0.19035287093416065
  (102079, 1126585)	0.07532266241608981
  (102079, 478244)	0.09470221859570901
  (102079, 378220)	0.15954730434554285
  (102079, 5071)	0.090476425548231
  (102079, 804025)	0.07240897603781826
  (102079, 109896)	0.08545918709827292
  (102079, 563656)	0.092193382902547
  (102079, 579945)	0.1453067499108596
  (102079, 843789)	0.06132462560512056
  (102079, 394670)	0.0740386093042695
  (102079, 968890)	0.06321357934287058
  (102079, 301088)	0.07335359164253069
  (102079, 174076)	0.07549624319992211
  (102079, 640185)	0.09188568625794899
  (102079, 387958)	0.05389741120781163
  (102079, 662833)	0.05618018293781947
  (102079, 1146654)	0.05606523184849747
  (102079, 1010375)	0.06500294720040516
```

Figure 3.6 Feature Extraction

3.1.5 SPRINT RETROSPECTIVE

	Sprint Retrospective			
Liked	**Learned**	**Lacked**	**Longed For**	
Share aspects of the sprint that you enjoyed or found particularly effective.	*Discuss lessons learned, whether they are related to processes, technical aspects, or teamwork.*	*Identify areas where the team felt a lack of resources, support, or information.*	*Discuss any desires or expectations that the team had but were not met during the sprint.*	*Guidelines*
The search feature worked well, allowing users to find articles, sources, and users easily.	Gained insights into the complexity of fake news detection and credibility labeling.	A more robust dataset to improve the fake news detection accuracy.	More user feedback on the usability of credibility labels and fake news flags.	Search returns relevant results and labels them as "Real" or "Fake" accurately.

Figure 3.7 sprint retrospective

3.2 SPRINT 2

3.2.1 OBJECTIVES WITH USER STORIES OF SPRINT 2

The Goal of the Second sprint is to Identifying to validate the accuracy of the machine learning models on test data so that I can ensure the reliability of the fake news detection system.

S.No	**Detailed User Stories**
US 4	As a user, I want to validate the accuracy of the machine learning models on test data so that I can ensure the reliability of the fake news detection system.
US 5	As a user, I want to analyze the effectiveness of different algorithms in detecting fake news so that I can provide future research guidance.
US 6	As a user, I want to receive detailed insights on the credibility of news articles based on the analysis so that I can make informed decisions.

Table 3.3 Detailed User Stories of sprint 2

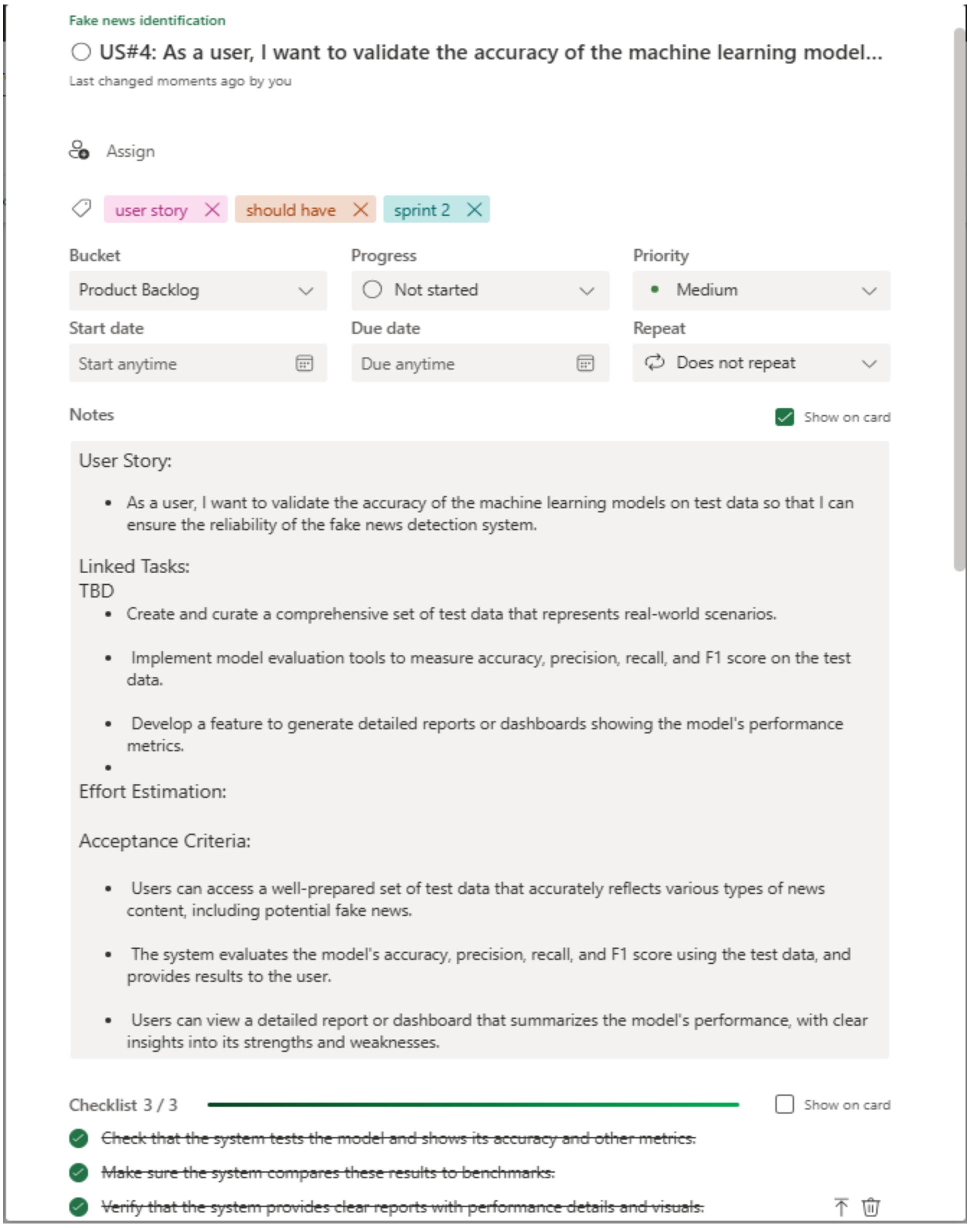


Figure 3.8 User story for Identifying Machine Learning Model

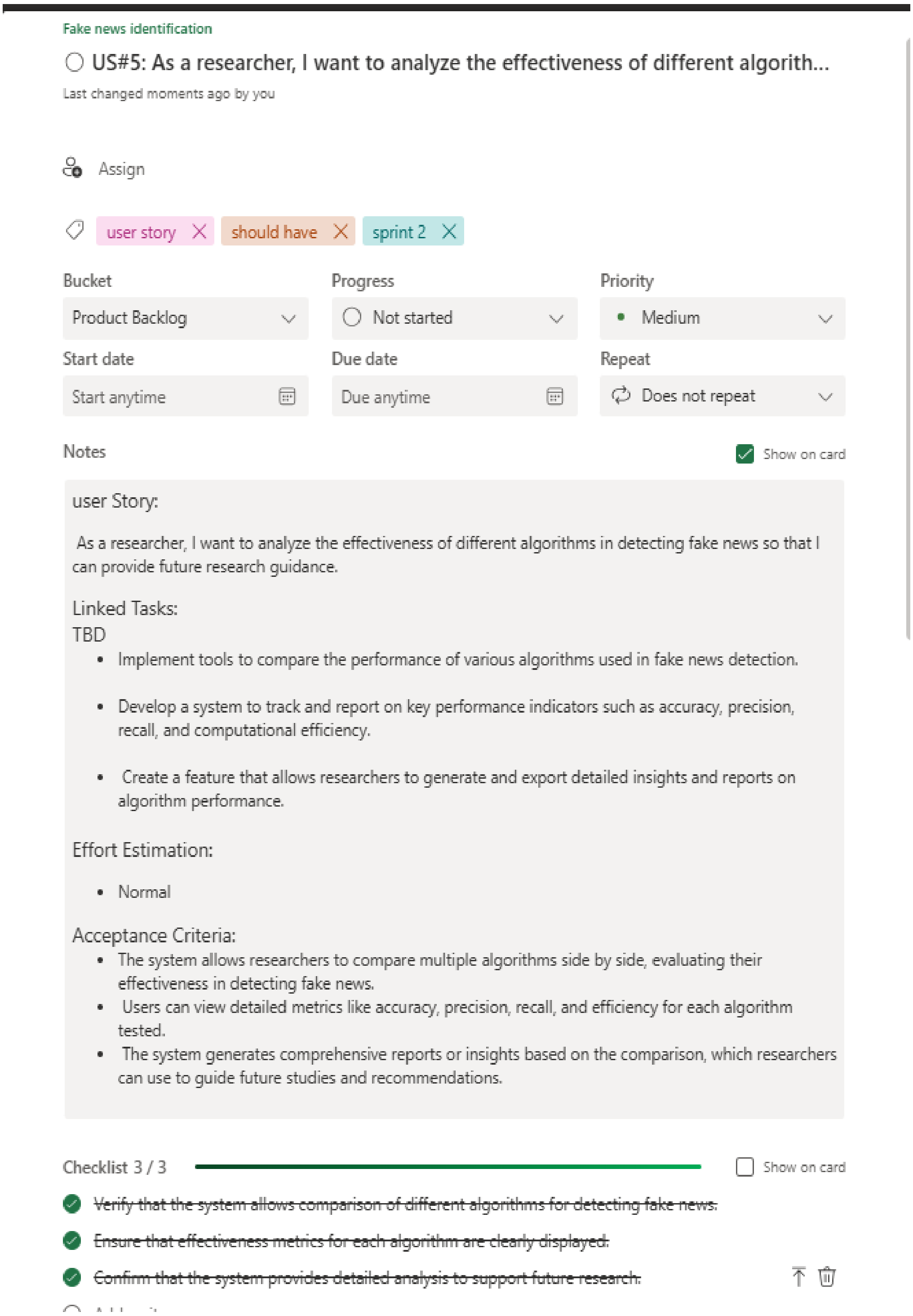


Figure 3.9 user story for analyze the effectiveness of different algorithms in detecting fake news

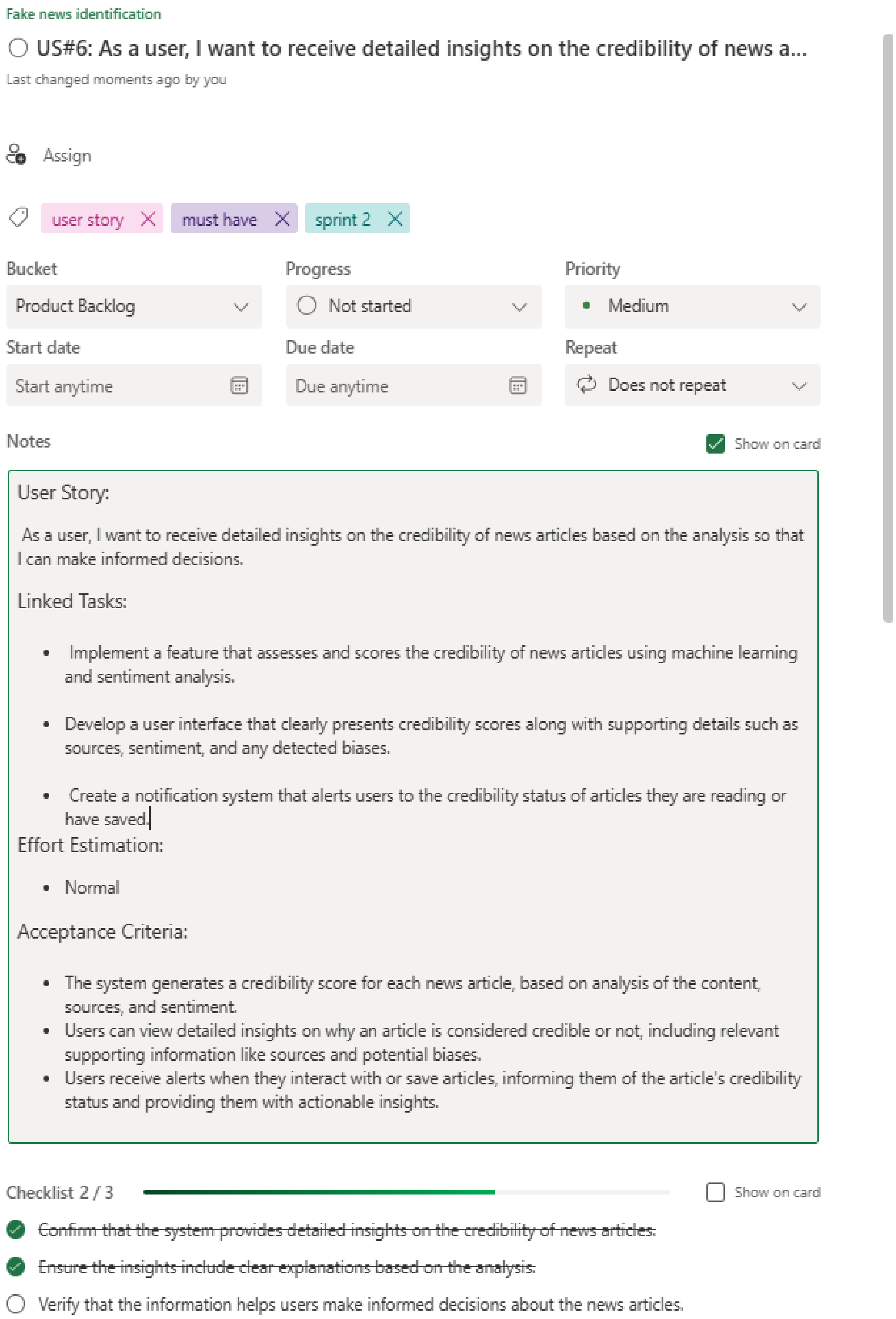


Figure 3.10 user story for credibility of news article

3.2.2 FUNCTIONAL DOCUMENT

3.2.2.1 Introduction:

With the rapid spread of fake news, finding the best algorithm to detect misinformation is essential. This platform supports researchers in testing different machine learning algorithms to find the most accurate ones for identifying fake news. Researchers can compare how algorithms, like NLP and deep learning, perform against fake news to see which methods are most effective. This helps improve future fake news detection systems and provides insights on which methods handle complex cases and adapt to new patterns in misinformation.

3.2.2.2 Product Goal:

The goal is to provide a platform for researchers to analyze and compare various machine learning algorithms for fake news detection. By testing deep learning, NLP, and other classification methods, researchers can determine which ones deliver accurate and dependable results. The platform focuses on key performance metrics like accuracy, precision, and adaptability to new misinformation types. This will support advancements in fake news detection and guide future research for more reliable detection tools.

3.2.2.3 Demography

Users:

- Researchers and data scientists focused on fake news and machine learning.
- Academics, students, and data analysts in news organizations interested in detecting misinformation.
- Developers of AI systems focused on content moderation for social media and news.

Location:

- Worldwide, especially regions with high research activity like North America, Europe, and Asia.

3.2.2.4 Business Processes:

1. Algorithm Evaluation and Comparison:
- Researchers can choose and test different algorithms on labeled datasets.
- The platform calculates metrics like accuracy, precision, and recall to compare performance.

2. Performance Report Generation:
- Automated reports show detailed results, with visuals like charts for easy understanding.
- Reports can be customized based on specific metrics or datasets.

3. Research Guidance and Insights:
- Provides recommendations on which algorithms work best for certain misinformation types.
- Suggests research directions based on recent findings and data trends.

4. Collaboration and Feedback:
- Researchers can share reports and collaborate on findings.
- Users provide feedback, helping improve the platform’s algorithm library.

5. Data Management and Security:
- Secure storage for user-uploaded data.
- Version control to keep track of updates to algorithms and datasets.

3.2.2.5 Features

1. Algorithm Performance Evaluation:
- Researchers can evaluate and compare different algorithms for detecting fake news.
- Includes metrics like accuracy, precision, and recall.

User Story: As a researcher, I want to test various algorithms on fake news detection so I can identify the most effective ones for my work.

2. Comprehensive Report Generation:
- Generates reports with performance metrics and visuals to highlight algorithm strengths and weaknesses.

User Story: As a researcher, I want detailed reports on algorithm performance to understand their strengths and limitations better.

3. Dataset Upload and Management:
- Allows users to upload custom datasets to test algorithms.
- Provides secure storage and format support for personalized analysis.

User Story: As a researcher, I want to upload my datasets to test algorithms on data that matches my specific research needs.

3.2.2.6 Authorization Matrix:

Role	Upload Datasets	Run Algorithm Tests	Generate Reports	Access Recommendations	Share Findings	Give Feedback	Manage Data Versions
Researcher	Yes, own datasets only	Yes, on own and public datasets	Yes, custom reports	View insights for own work	Share with others	Provide feedback	View own dataset versions
Data Scientist	Yes, all datasets	Yes, all datasets	Detailed reports for all	Create new recommendations	Share publicly	Provide & review feedback	Full version control
Platform Admin	Yes, manage all datasets	Configure algorithms for platform	Platform-wide reports	Update insights and recommendations	Manage sharing settings	Review feedback	Full access to security & backups
Student/Academic User	Yes, own datasets only	Yes, on own & public datasets	Basic reports	Limited insights	Share with assigned groups	Provide feedback	View public dataset versions

3.2.2.7Assumptions

1. High-Quality Datasets:
- Researchers will have access to labeled datasets essential for evaluating algorithms. Without these, it’s hard to assess algorithm performance across misinformation types.

2. Knowledge of ML Metrics:
- Users are assumed to understand machine learning concepts and performance metrics like accuracy and recall.

3. Algorithm Adaptability:
- The algorithms can be updated to handle new types of misinformation over time.

4. Reliable Internet Access:
- Consistent internet access is needed as the platform is cloud-based, and interruptions may affect performance analysis.

These foundational assumptions ensure that researchers can effectively analyze and improve fake news detection algorithms on the platform.

3.2.3 ARCHITECTURE DOCUMENT

3.2.3.1 Application :

An ER (Entity-Relationship) diagram for a fake news detection system using sentiment analysis and machine learning can help in structuring the data relationships and interactions within the system. This system typically involves several entities, such as News Articles, Users, Comments, Sentiments, and Classifications. The News Articles entity would include attributes like `article_id`, `title`, `content`, `source`, and `date published`, where each article is analyzed for fake news likelihood. The Users entity contains user details, including `user_id`, `name`, and `engagement history`, representing how users interact with the articles. The Comments entity connects to both Users and News Articles, allowing the system to evaluate user responses or comments linked to specific articles.

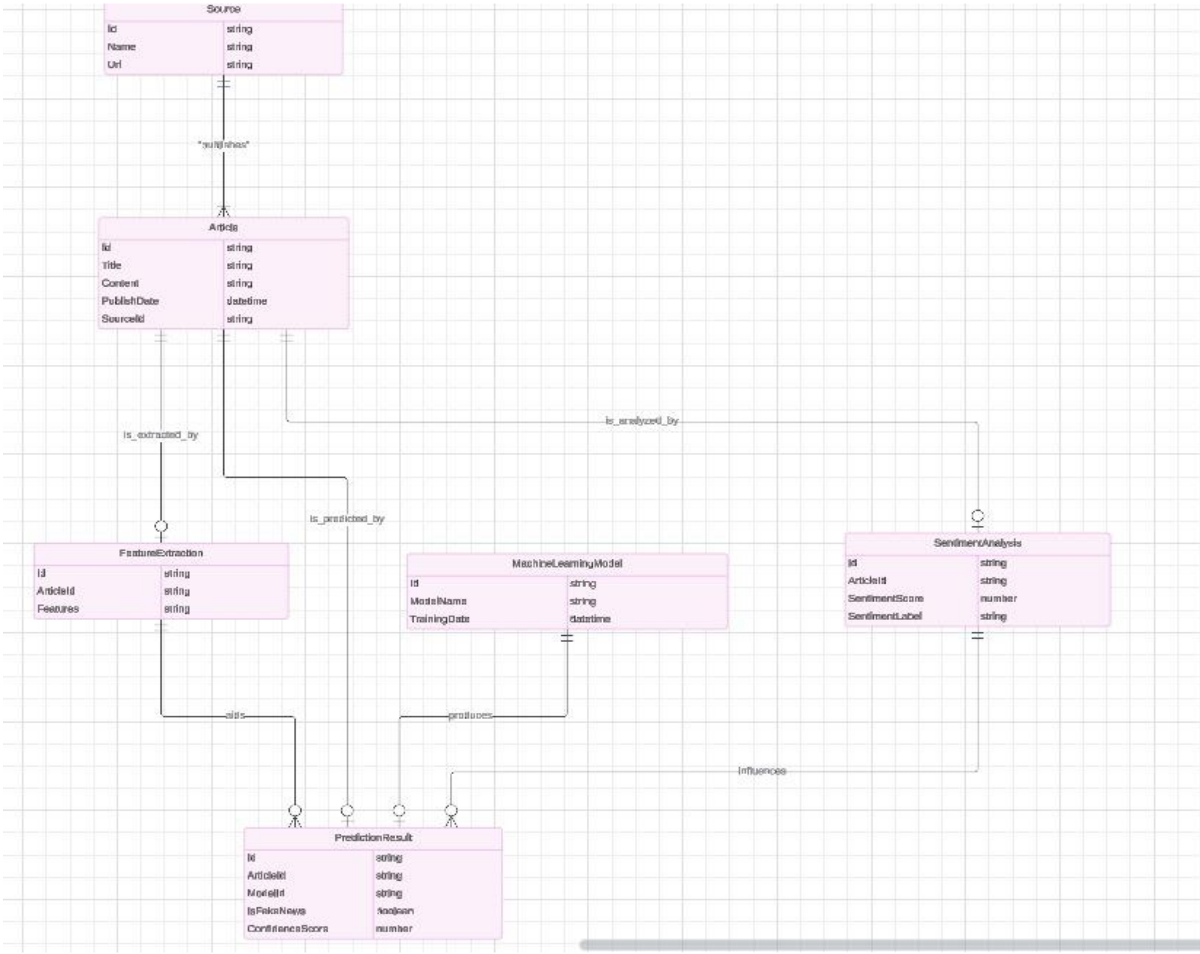


Figure 3.11 Entity Relationship Diagram

The system leverages a Sentiments entity to record the results of sentiment analysis on both the article content and user comments. Attributes such as `sentiment score` and `sentiment type` (e.g., positive, negative, neutral) help in assessing the general tone associated with an article. Another critical entity, Classifications, stores the output of machine learning models, which can include `classification_id`, `algorithm used`, `confidence score`, and `label` (e.g., fake, real). These models process text features, sentiment scores, and other metadata to make predictions about an article's credibility. Relationships between these entities support efficient data flow: for instance, Users can engage with multiple News Articles through Comments, and each article can have an associated Sentiment and Classification. The ER diagram for this system thus provides a structured foundation, helping in implementing a comprehensive, AI-powered fake news detection system that combines sentiment analysis and machine learning insights.

3.2.3.3. Training the Model :

To train a model for fake news detection using sentiment analysis, we start by collecting a dataset of news articles labeled as either real or fake, along with any user comments that can indicate sentiment. Next, we clean and preprocess the text, removing unnecessary elements, and breaking it down into simpler forms. We then analyze sentiment by scoring each article and comment as positive, negative, or neutral, which can help reveal general tone. Important features for training might include text patterns, sentiment scores, and metadata like the publisher's credibility. We then choose a model, such as logistic regression or a deep learning model like BERT, and train it using an 80-20 split of data for training and testing. The model's performance is evaluated on the test data using accuracy and other metrics, with tuning to improve results. Finally, we deploy the trained model, allowing it to detect fake news in real-time and continually monitor and update its accuracy over time. This combination of machine learning and sentiment analysis enables a robust system for spotting potentially misleading information.

3.2.4 OUTCOME OF OBJECTIVE :

```
[2]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.corpus import stopwords
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score,classification_report,confusion_matrix
import warnings
import pickle
warnings.filterwarnings('ignore')
```

```
[2]: df_train=pd.read_csv("train_preprocessed_lemmatized.csv")
df_test=pd.read_csv("test_preprocessed_lemmatized.csv")
df=pd.concat([df_train,df_test],axis=0)
df.head()
```

[2]:

	Unnamed: 0	text	Class Index
0	0	wall st bear claw back black reuter reuters sh...	3
1	1	carlyle look toward commercial aerospace reute...	3
2	2	oil economy cloud stock outlook reuters reuter...	3
3	3	iraq halt oil export main southern pipeline re...	3
4	4	oil price soar alltime record pose new menace ...	3

Train Test Splitting(80-20%)

```
[3]: from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(df["text"],df["Class Index"],test_size=0.2,random_state=42)
```

Figure 3.12 Training the dataset

3.2.4.1 Classification Algorithms:

Logistic Regression

```
[5]: from sklearn.linear_model import LogisticRegression
nc_lr=LogisticRegression(C=1,solver="saga")
nc_lr.fit(X_train_tfidf, y_train)
pickle.dump(nc_lr,open("nc_lr.pkl","wb"))
y_pred_lr = nc_lr.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_lr)
print("Accuracy:", accuracy)
```

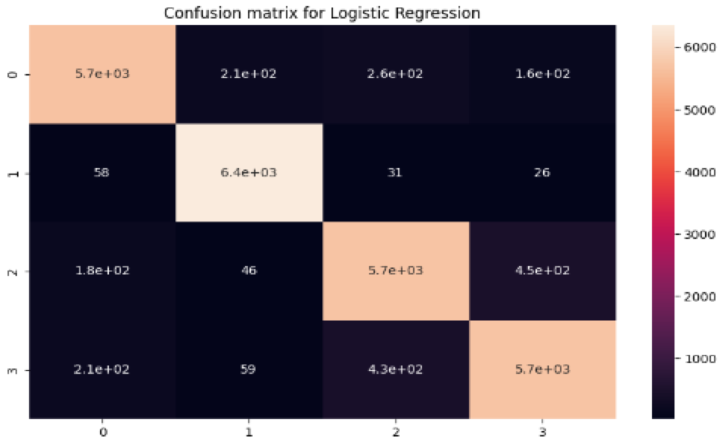
```
Accuracy: 0.9170846394984326
```

```
[6]: print("Classification Report:\n",classification_report(y_test,y_pred_lr))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_lr),annot=True)
plt.title("Confusion matrix for Logistic Regression")
plt.show()
```

```
Classification Report:
              precision    recall  f1-score   support

           1       0.93      0.90      0.91      6283
           2       0.95      0.98      0.97      6466
           3       0.89      0.89      0.89      6370
           4       0.90      0.89      0.89      6401

    accuracy                           0.92     25520
   macro avg       0.92      0.92      0.92     25520
weighted avg       0.92      0.92      0.92     25520
```



SVM(Support Vector Machine)

```
[7]: from sklearn.svm import SVC
nc_svm = SVC()
nc_svm.fit(X_train_tfidf, y_train)
pickle.dump(nc_svm,open("nc_svm.pkl","wb"))
y_pred_svm = nc_svm.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_svm)
print("Accuracy:", accuracy)
```

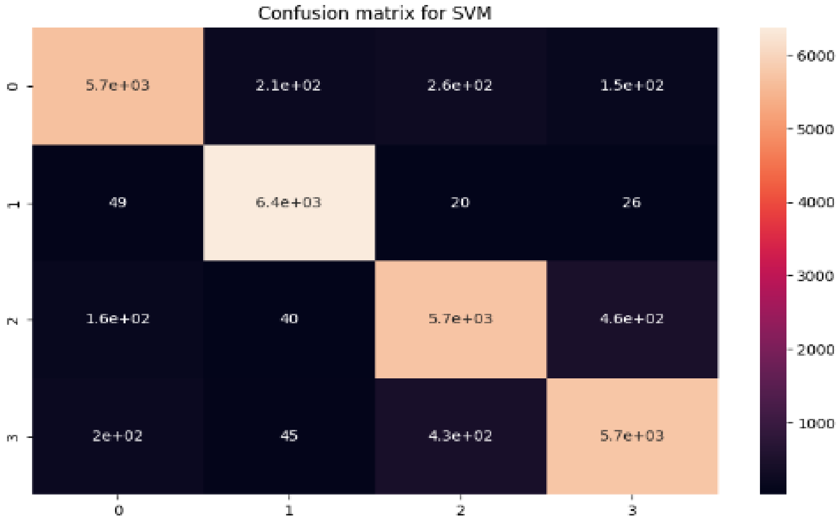
```
Accuracy: 0.9198667711598746
```

```
[8]: print("Classification Report:\n",classification_report(y_test,y_pred_svm))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_svm),annot=True)
plt.title("Confusion matrix for SVM")
plt.show()
```

```
Classification Report:
               precision    recall  f1-score   support

           1       0.93      0.90      0.92      6283
           2       0.96      0.99      0.97      6466
           3       0.89      0.90      0.89      6370
           4       0.90      0.90      0.90      6401

    accuracy                           0.92     25520
   macro avg       0.92      0.92      0.92     25520
weighted avg       0.92      0.92      0.92     25520
```



K-Nearest Neighbors

```
[9]: from sklearn.neighbors import KNeighborsClassifier
nc_knn = KNeighborsClassifier(n_neighbors=43)
nc_knn.fit(X_train_tfidf, y_train)
pickle.dump(nc_knn,open("nc_knn.pkl","wb"))
y_pred_knn = nc_knn.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_knn)
print("Accuracy:", accuracy)
```

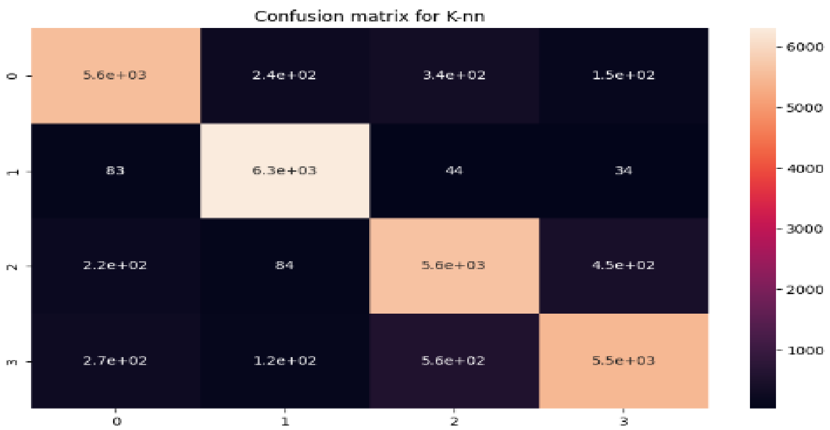
```
Accuracy: 0.8987852664576802
```

```
[10]: print("Classification Report:\n",classification_report(y_test,y_pred_knn))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_knn),annot=True)
plt.title("Confusion matrix for K-nn")
plt.show()
```

```
Classification Report:
               precision    recall  f1-score   support

           1       0.91      0.88      0.90      6283
           2       0.93      0.98      0.95      6466
           3       0.86      0.88      0.87      6370
           4       0.90      0.85      0.87      6401

    accuracy                           0.90     25520
   macro avg       0.90      0.90      0.90     25520
weighted avg       0.90      0.90      0.90     25520
```



Multinomial NaiveBayes

[11]:
```
from sklearn.naive_bayes import MultinomialNB
nc_nb = MultinomialNB(alpha=0.1)
nc_nb.fit(X_train_tfidf, y_train)
pickle.dump(nc_nb,open("nc_nb.pkl","wb"))
y_pred_nb = nc_nb.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_nb)
print("Accuracy:", accuracy)
```

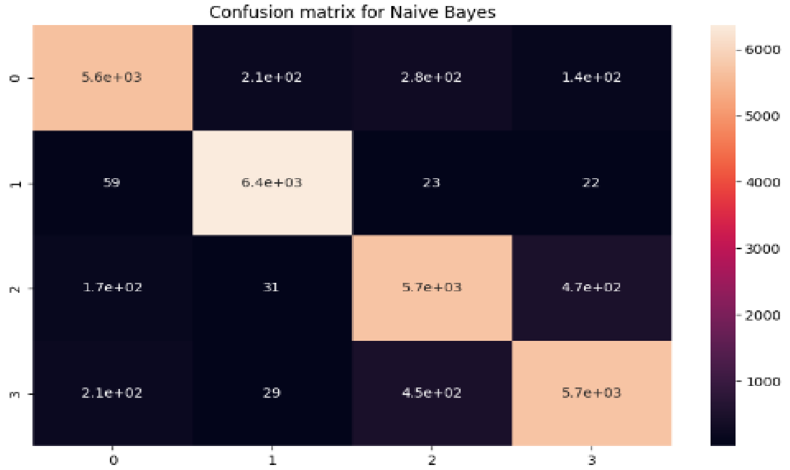
```
Accuracy: 0.9177115987460815
```

[12]:
```
print("Classification Report:\n",classification_report(y_test,y_pred_nb))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_nb),annot=True)
plt.title("Confusion matrix for Naive Bayes")
plt.show()
```

```
Classification Report:
               precision    recall  f1-score   support

           1       0.93      0.90      0.91      6283
           2       0.96      0.98      0.97      6466
           3       0.88      0.89      0.89      6370
           4       0.90      0.89      0.90      6401

    accuracy                           0.92     25520
   macro avg       0.92      0.92      0.92     25520
weighted avg       0.92      0.92      0.92     25520
```



[13]:
```
models = ['Naive Bayes', 'Logistic Regression', 'SVM', 'KNN']
accuracy = [accuracy_score(y_test, y_pred_nb)*100,
            accuracy_score(y_test, y_pred_lr)*100,
            accuracy_score(y_test, y_pred_svm)*100,
            accuracy_score(y_test, y_pred_knn)*100]
bar_width = 0.3
index = np.arange(len(models))

plt.bar(index, accuracy, bar_width, label='Accuracy')
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Accuracy Score')
plt.xticks(index , models)
plt.ylim(80, 100)
plt.legend()
plt.show()
```

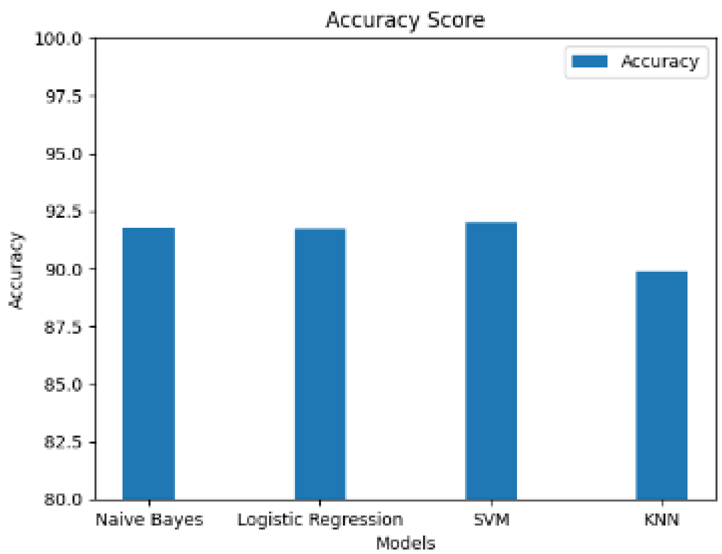


Figure 3.12.1 Accuracy of classification algorithms

The image shows a bar chart comparing the accuracy scores of four different machine learning models: Naive Bayes, Logistic Regression, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). Among these models, Naive Bayes, Logistic Regression, and SVM have similar accuracy scores, all around 90 92%. The KNN model, however, has a slightly lower accuracy, falling below the other three. This comparison indicates that while Naive Bayes, Logistic Regression, and SVM perform similarly well on the given dataset, KNN may not be as effective for this specific classification task.

3.2.4.2 Detection Algorithms:

[2]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
import pickle
from nltk.corpus import stopwords
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
from sklearn.metrics import confusion_matrix,classification_report
import warnings
warnings.filterwarnings("ignore")
```

[3]:
```python
df=pd.read_csv("fake_news_preprocessed_dataset.csv")
df.drop('Unnamed: 0',axis=1,inplace=True)
df.head()
```

[3]:

	data	label
0	law enforcement high alert follow threat cop w...	1
1	unbelievable obamas attorney general say charl...	1
2	bobby jindal raise hindu use story christian c...	0
3	satan russia unvelis image terrify new supernu...	1
4	time christian group sue amazon splc designati...	1

Train Test Splitting(80-20%)

[3]:
```python
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(df["data"],df["label"],test_size=0.2,random_state=42)
```

Feature Extraction

[4]:
```python
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_df=0.7,ngram_range=(1,2),stop_words='english')
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

Figure 3.12.2 Training and Feature Extraction

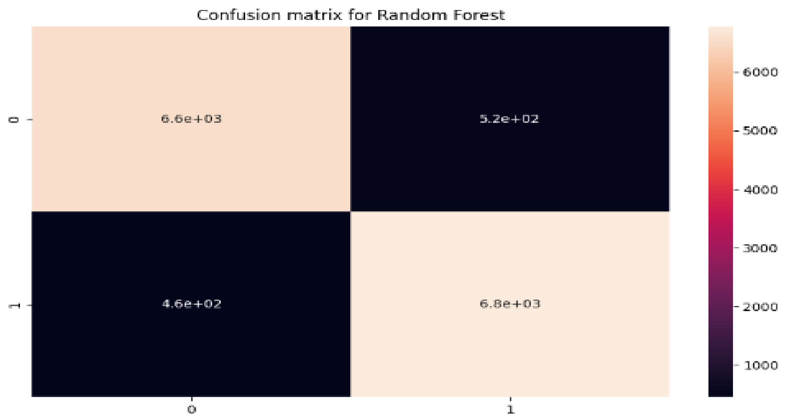
Random Forest

[5]:
```python
from sklearn.ensemble import RandomForestClassifier
fn_rf = RandomForestClassifier()
fn_rf.fit(X_train_tfidf,y_train)
pickle.dump(fn_rf,open("fn_rf.pkl","wb"))
y_pred_rf=fn_rf.predict(X_test_tfidf)
print("Accuracy:",accuracy_score(y_pred_rf,y_test)*100)
print("Classification Report:\n",classification_report(y_test,y_pred_rf))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_rf),annot=True)
plt.title("Confusion matrix for Random Forest")
plt.show()
```

```
Accuracy: 93.19960861056752
Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.93      0.93      7081
           1       0.93      0.94      0.93      7227

    accuracy                           0.93     14308
   macro avg       0.93      0.93      0.93     14308
weighted avg       0.93      0.93      0.93     14308
```



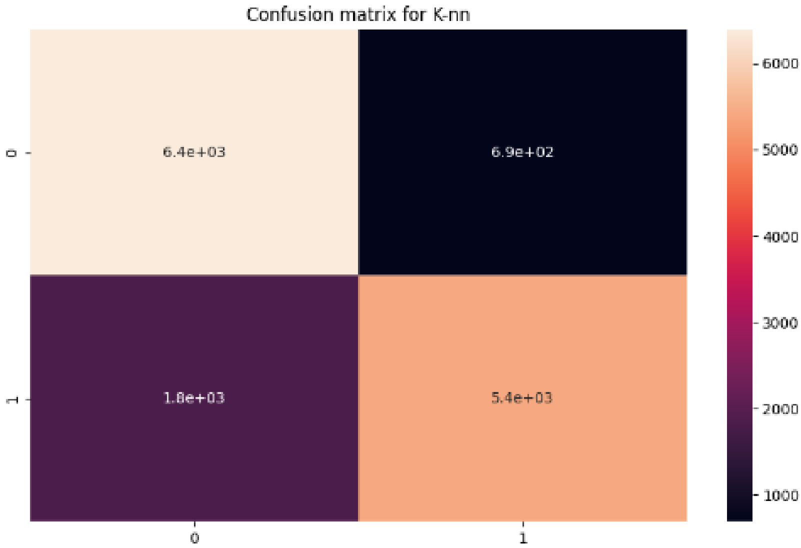
K-Nearest Neighbors

```
[6]: from sklearn.neighbors import KNeighborsClassifier
fn_knn = KNeighborsClassifier(n_neighbors=5)
fn_knn.fit(X_train_tfidf, y_train)
pickle.dump(fn_knn,open("fn_knn.pkl","wb"))
y_pred_knn = fn_knn.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_knn)
print("Accuracy:", accuracy*100)
print("Classification Report:\n",classification_report(y_test,y_pred_knn))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_knn),annot=True)
plt.title("Confusion matrix for K-nn")
plt.show()
```

```
Accuracy: 82.52725747833381
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.90      0.84      7081
           1       0.89      0.75      0.81      7227

    accuracy                           0.83     14308
   macro avg       0.83      0.83      0.82     14308
weighted avg       0.83      0.83      0.82     14308
```



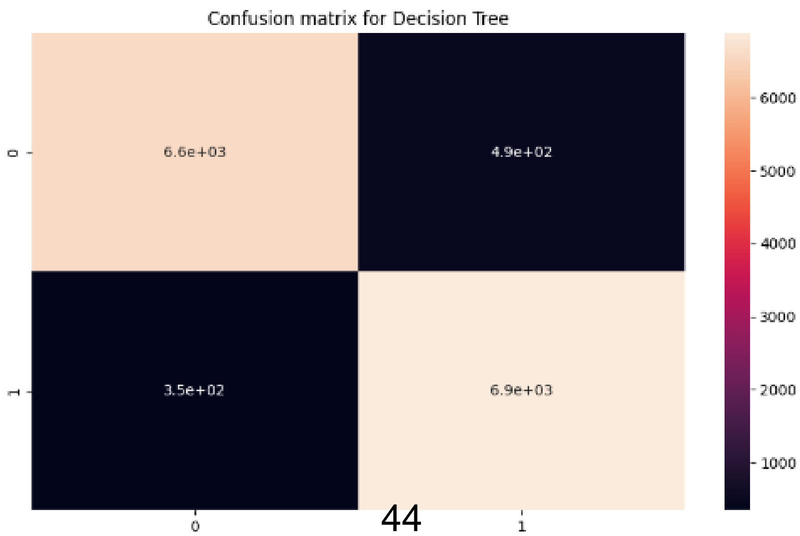
Decision Tree

```
[7]: from sklearn.tree import DecisionTreeClassifier
fn_dt = DecisionTreeClassifier()
fn_dt.fit(X_train_tfidf, y_train)
pickle.dump(fn_dt,open("fn_dt.pkl","wb"))
y_pred_dt = fn_dt.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_dt)
print("Accuracy:", accuracy*100)
print("Classification Report:\n",classification_report(y_test,y_pred_dt))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_dt),annot=True)
plt.title("Confusion matrix for Decision Tree")
plt.show()
```

```
Accuracy: 94.16410399776349
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.93      0.94      7081
           1       0.93      0.95      0.94      7227

    accuracy                           0.94     14308
   macro avg       0.94      0.94      0.94     14308
weighted avg       0.94      0.94      0.94     14308
```



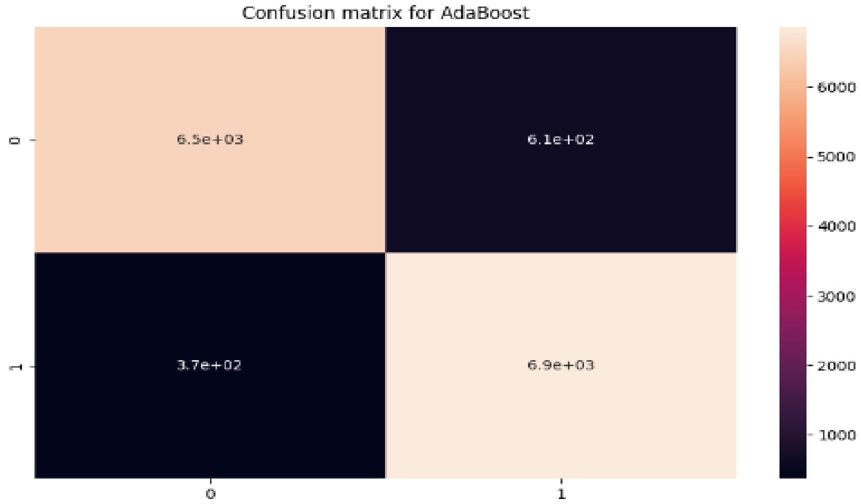
AdaBoost

```
[8]: from sklearn.ensemble import AdaBoostClassifier
fn_ab = AdaBoostClassifier()
fn_ab.fit(X_train_tfidf, y_train)
pickle.dump(fn_ab,open("fn_ab.pkl","wb"))
y_pred_ab = fn_ab.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_ab)
print("Accuracy:", accuracy*100)
print("Classification Report:\n",classification_report(y_test,y_pred_ab))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_ab),annot=True)
plt.title("Confusion matrix for AdaBoost")
plt.show()
```

```
Accuracy: 93.17165222253286
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.91      0.93      7081
           1       0.92      0.95      0.93      7227

    accuracy                           0.93     14308
   macro avg       0.93      0.93      0.93     14308
weighted avg       0.93      0.93      0.93     14308
```



Comparision

```
[10]: models = ['Random Forest', 'KNN', 'Decision Tree', 'AdaBoost']
accuracy = [accuracy_score(y_test, y_pred_rf)*100,
            accuracy_score(y_test, y_pred_knn)*100,
            accuracy_score(y_test, y_pred_dt)*100,
            accuracy_score(y_test, y_pred_ab)*100]
bar_width = 0.3
index = np.arange(len(models))

plt.bar(index, accuracy, bar_width, label='Accuracy')
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Accuracy Score')
plt.xticks(index , models)
plt.ylim(0, 100)
plt.legend()
plt.show()
```

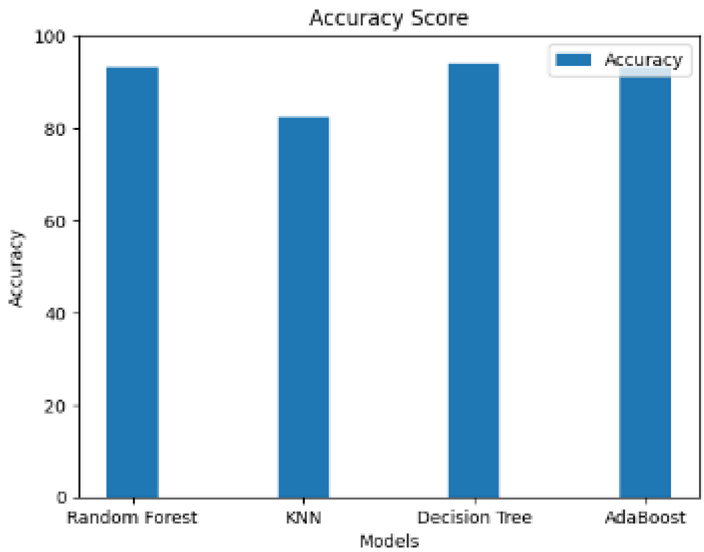


Figure 3.12.3 Accuracy of detection algorithms

The bar chart compares the accuracy scores of four different machine learning models: Random Forest, KNN, Decision Tree, and AdaBoost. Among these, Random Forest, Decision Tree, and AdaBoost show similar high accuracy, all above 90%, indicating strong performance in making correct predictions. The KNN model, however, has a slightly lower accuracy, around 82%, suggesting it is less effective compared to the other models in this case. Overall, the chart highlights that the ensemble methods like Random Forest and AdaBoost tend to perform better than KNN for this dataset.

3.2.4 SPRINT RETROSPECTIVE

	Sprint Retrospective			
Liked	**Learned**	**Lacked**	**Longed For**	
Share aspects of the sprint that you enjoyed or found particularly effective.	*Discuss lessons learned, whether they are related to processes, technical aspects, or teamwork.*	*Identify areas where the team felt a lack of resources, support, or information.*	*Discuss any desires or expectations that the team had but were not met during the sprint.*	*Guidelines*
[illegible]	[illegible]	[illegible]	[illegible]	labels
Machine learning model evaluations provided meaningful metrics like accuracy and F1 score.	Learned that precision and recall can vary significantly based on the type of content being tested.	Better integration between the ML model's performance dashboard and the user interface.	A more intuitive interface for users to interpret model performance reports and credibility scores.	report on false positives.

Figure 3.14 sprint retrospective

3.3 SPRINT 3

3.3.1 OBJECTIVES WITH USER STORIES OF SPRINT 3

The Goal of the Third sprint is to perform continuously update and improve the training datasets with new and diverse examples so that the detection system remains accurate over time.

S.No	Detailed User Stories
US 7	As a user, I want to monitor the sources of news articles flagged as fake so that I can take action to block or warn about unreliable sources.
US 8	As a user, I want to continuously update and improve the training datasets with new and diverse examples so that the detection system remains accurate over time.

Table 3.5 Detailed User Stories of sprint 3

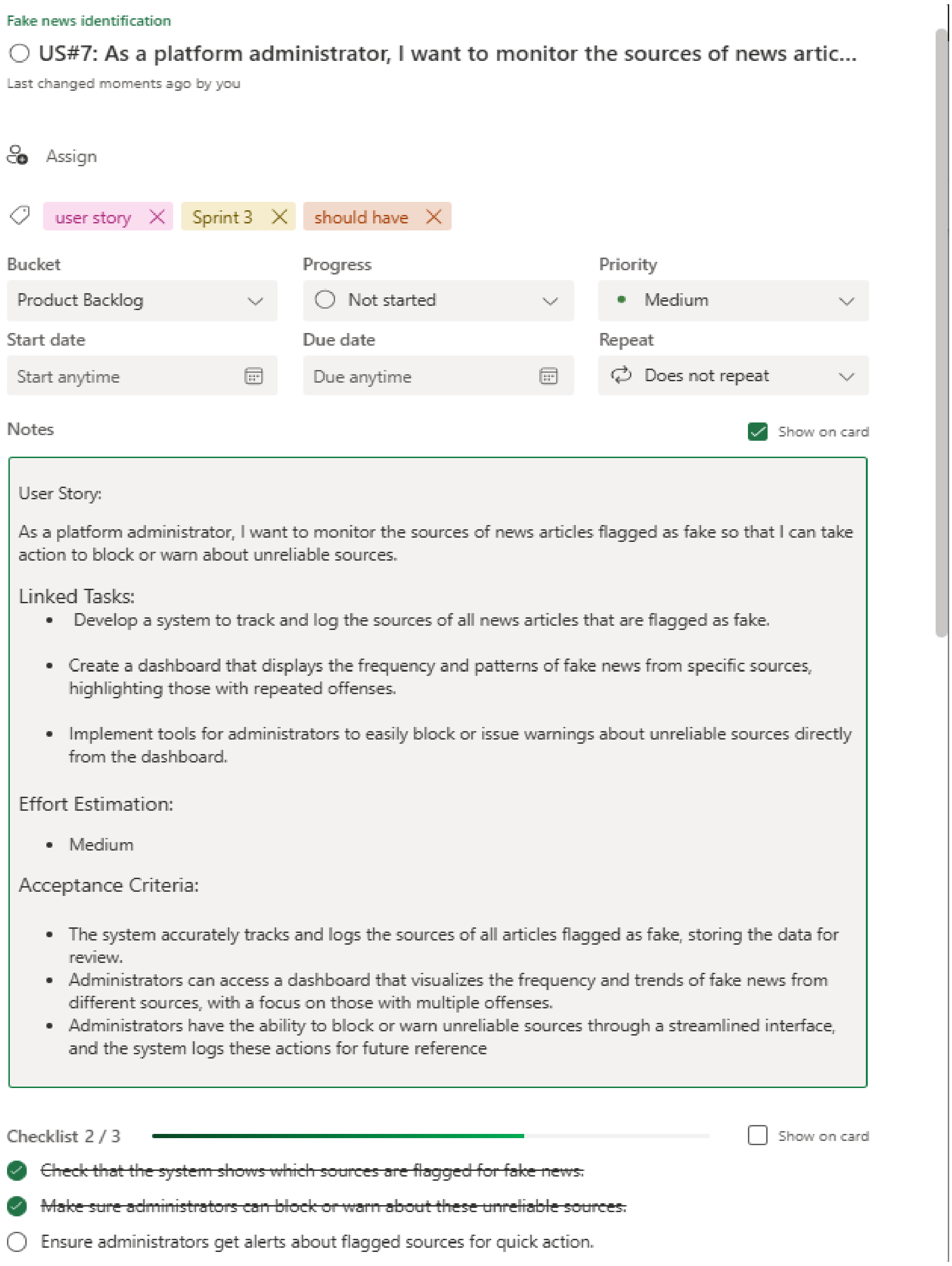


FIGURE 3.15 User story for monitoring the sources of news

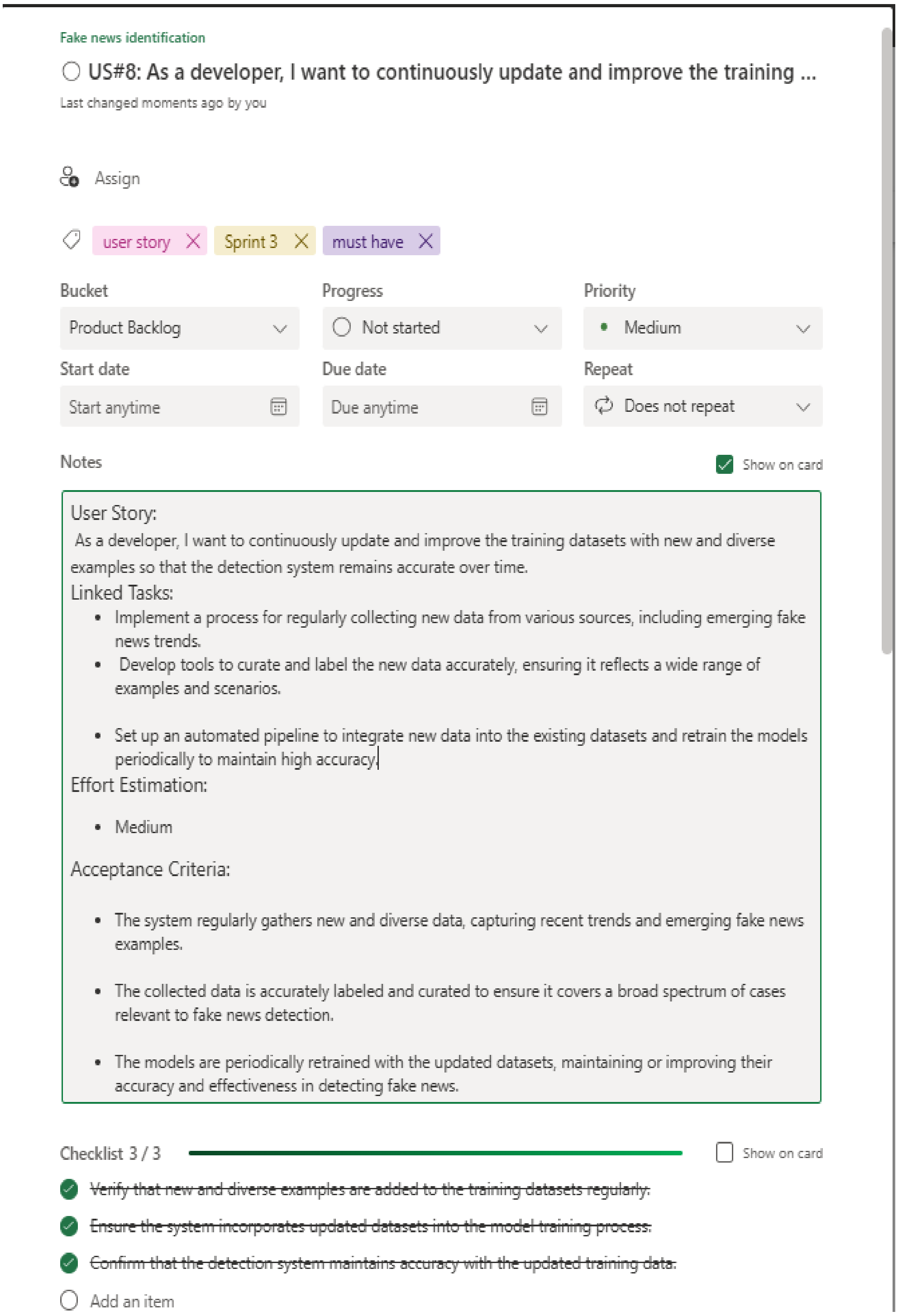


FIGURE 3.16 User story for updates and improves the datasets

3.3.2 FUNCTIONAL DOCUMENT

3.3.2.1. Introduction

The project aims to monitor and identify the sources of news articles flagged as fake, enabling users or organizations to take action by either blocking or warning about unreliable sources. This system leverages continuous data collection and machine learning to detect fake news, and it also updates the datasets regularly to ensure accurate detection over time.

3.3.2.2. Product Goal

The primary goal of the product is to provide users with a robust system that identifies fake news, tracks its sources, and allows users to take actions like blocking or flagging suspicious outlets. The system also focuses on keeping its detection methods accurate and up-to-date through continuous learning from new examples of fake news.

3.3.2.3. Demography

Users:

- Media organizations, digital platforms, government agencies, or individual users who want to filter or flag potentially false information.
- Users who seek reliable news and wish to block unreliable sources to avoid misinformation.

Location:

- Global reach with a focus on regions with high internet usage, social media activity, and where misinformation is prevalent, such as North America, Europe, and parts of Asia.

3.3.2.4. Business Processes

- News Source Monitoring: Continuously track news articles from various online sources.
- Fake News Detection: Use machine learning algorithms to identify potentially false or misleading articles.
- Actionable Alerts: Provide users with tools to block, flag, or warn about sources identified as unreliable.
- Continuous Dataset Updates: Regularly update the training datasets with new examples of fake news from various sources, ensuring the model adapts to evolving misinformation trends.

3.3.2.5. Features

3.3.2.5.1 Feature 1: News Source Monitoring

Description:

- The system continuously scans and collects news articles from a variety of sources, including websites, blogs, social media posts, and news aggregators. It flags content based on predetermined criteria and user reports to identify potential misinformation sources.

User Story:

- As a platform user or administrator, I want to be able to track the sources of flagged news articles so that I can block unreliable sources or warn others about their content.

3.3.2.5.2 Feature 2: Fake News Detection

Description:

- The system uses machine learning algorithms to detect patterns in news articles, identifying potentially fake or misleading content. These algorithms are trained on diverse datasets of known fake and real news and are continuously updated with new examples.

User Story:

- As an organization monitoring online content, I want to receive alerts when fake news is detected so that I can take swift action to address misinformation.

3.3.2.5.3 Feature 3: Continuous Learning and Dataset Updating

Description:

- The system periodically updates its training datasets with new examples of fake news, incorporating different sources, regions, and formats (text, images, videos). This ensures that the detection algorithms remain effective as misinformation evolves over time.

User Story:

- As an admin, I want the system to continuously improve its accuracy in detecting fake news by updating its datasets with new and diverse examples, so I can ensure reliable detection over time.

3.3.2.6 Authorization Matrix :

Role	News Source Monitoring	Fake News Detection	Receive Alerts	Block or Flag Sources	Update Datasets	System Configuration
Platform User	View flagged sources	View detected fake news	Yes	Flag or report sources	No	No
Organization Admin	Monitor all sources	Run detection on all sources	Yes	Block or flag sources for all users	Request updates	Limited configurations (e.g., alert settings)
System Admin	Full monitoring access	Configure and run detection	Yes	Block or flag at system level	Manage dataset updates	Full configuration access

Table 3.6 Authorization Matrix

3.3.2.7 Assumptions:

- The system will have access to a large volume of data from various sources for continuous monitoring and dataset updating.
- Users are likely to interact with the platform primarily through online dashboards or APIs.
- Misinformation trends will evolve, necessitating ongoing refinement and enhancement of machine learning models.
- The system will operate under applicable laws and regulations regarding online content moderation, privacy, and data security.

3.3.3 ARCHITECTURE DOCUMENT

3.3.3.1 Application:

This flowchart shows the process of detecting fake news using sentiment analysis and machine learning. Here's a simple explanation:

1. **Start:** The process begins with collecting data from various news sources.
2. **Preprocessing:** The collected data is then prepared, cleaned, and formatted for analysis.
3. **Sentiment Analysis:** Sentiment analysis is applied to assess the tone of the content. If the analysis shows a positive sentiment, the content is flagged as potentially fake.
4. **Machine Learning Model:** If sentiment alone does not indicate fake news, a machine learning model is used to further evaluate the content.
5. **Decision Point:** Based on the model's results, content is either marked for further review or classified as fake news if it meets the criteria.
6. **Storage and Reporting**: All results are stored in a database, and reports or alerts are generated to notify about flagged or classified content.

This system provides a structured approach to identifying fake news by combining sentiment analysis with machine learning for accuracy.

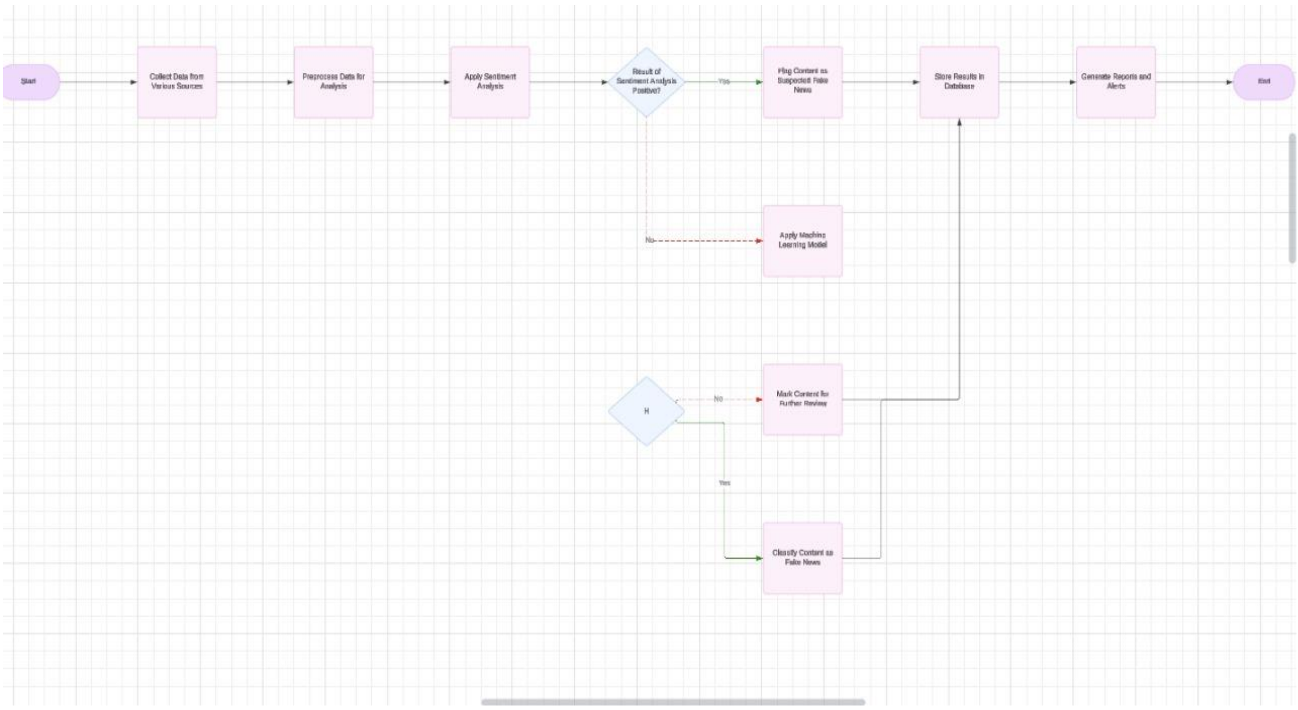


Figure 3.17 Activity Diagram

3.3.3.3 Testing the Model :

Testing the fake news detection model involves evaluating its performance on a separate dataset that was not used during training. This dataset contains labeled articles (either real or fake) to allow a fair assessment of the model's accuracy. Each article in the test data undergoes the same preprocessing and sentiment analysis steps as during training, after which the model predicts whether the article is fake or real. To evaluate the model, key metrics are used, including accuracy (the percentage of correct predictions), precision (the proportion of articles predicted as fake that are actually fake), recall (the proportion of actual fake articles that the model successfully identifies), and the F1 score, which combines precision and recall to handle imbalanced data. Additionally, any errors are analyzed to identify patterns in misclassification, which can guide improvements to the model. Thresholds for sentiment and confidence scores may also be adjusted to find an optimal balance between accurately detecting fake news and minimizing false positives. Finally, the test results are stored, and if the model is deployed, it will be monitored over time to ensure it continues performing well as new data becomes available. This testing process is essential for validating the model's effectiveness and preparing it for real-world use.

3.3.4 OUTCOME OF OBJECTIVE

```
[1]: import pandas as pd
```

```
[5]: df=pd.read_csv("J:\\mini project\\sentiment analysis\\all-data.csv",encoding="latin1",names=["label","data"])
print(df.shape)
```

```
(4846, 2)
```

```
[6]: df.head()
```

[6]:

	label	data
0	neutral	According to Gran , the company has no plans t...
1	neutral	Technopolis plans to develop in stages an area...
2	negative	The international electronic industry company ...
3	positive	With the new production plant the company woul...
4	positive	According to the company 's updated strategy f...

```
[7]: df.isnull().sum()
```

```
[7]: label    0
data     0
dtype: int64
```

```
[10]: df["label"].value_counts()
```

```
[10]: label
neutral     2879
positive    1363
negative     604
Name: count, dtype: int64
```

```
[ ]: import warnings
warnings.filterwarnings("ignore")
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
def preprocess_text(text):
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub(r'\d+', '', text)
    tokens = word_tokenize(text)
    tokens = [token.lower() for token in tokens]
    stop_words = set(stopwords.words('english'))
    tokens = [token for token in tokens if token not in stop_words]
    preprocessed_text = ' '.join(tokens)
    return preprocessed_text
df['data'] = df['data'].apply(preprocess_text)
```

```
[ ]: import spacy
nlp = spacy.load("en_core_web_sm")
def lemmatize_text(text):
    doc = nlp(text)
    lemmatized_text = " ".join([token.lemma_ for token in doc])
    return lemmatized_text
df['data'] = df['data'].apply(lemmatize_text)
df[["data","label"]].to_csv("sentimental_news_preprocessed_dataset.csv")
```

Figure 3.18 Preprocessing the model

```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.corpus import stopwords
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
from sklearn.metrics import confusion_matrix,classification_report
import warnings
warnings.filterwarnings("ignore")
import pickle
```

```
[2]: df=pd.read_csv("sentimental_news_preprocessed_dataset.csv")
df["label"]=df["label"].map({"neutral":0,"positive":1,"negative":2})
df.dropna(inplace=True)
df.head()
```

[2]:

	Unnamed: 0	data	label
0	0	accord gran company plan move production russi...	0
1	1	technopoli plan develop stage area less square...	0
2	2	international electronic industry company elco...	2
3	3	new production plant company would increase ca...	1
4	4	accord company update strategy year basware ta...	1

Train Test Splitting(80-20%)

```
[3]: from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(df["data"],df["label"],test_size=0.2,random_state=42)
```

```
[4]: from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_df=0.7,ngram_range=(1,2),stop_words='english')
sa_tf = tfidf_vectorizer.fit(X_train)
pickle.dump(sa_tf,open("sa_tf.pkl","wb"))
```

Feature Extraction

```
[19]: from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_df=0.7,ngram_range=(1,2),stop_words='english')
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

Figure 3.18.1 Testing and Feature Extraction

3.18.2 Sentimental Analysis Algorithms:

Logistic Regression

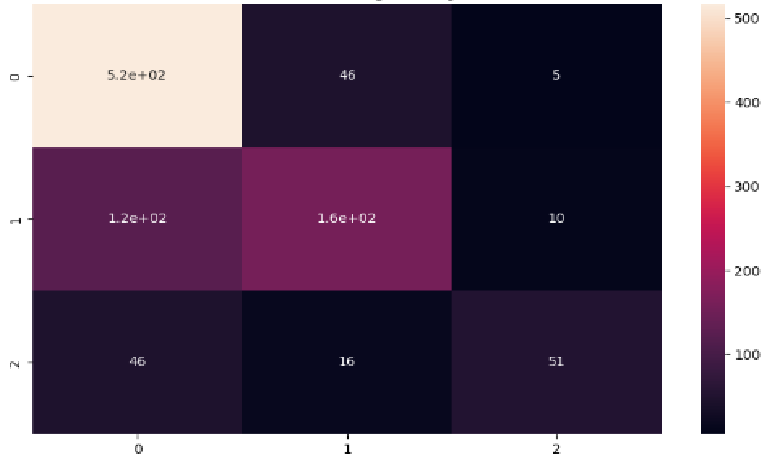
[31]:
```python
from sklearn.linear_model import LogisticRegression
sa_lr=LogisticRegression(C=3,solver="sag")
sa_lr.fit(X_train_tfidf, y_train)
pickle.dump(sa_lr,open("sa_lr.pkl","wb"))
y_pred_lr = sa_lr.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_lr)
print("Accuracy:", accuracy*100)
print("Classification Report:\n",classification_report(y_test,y_pred_lr))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_lr),annot=True)
plt.title("Confusion matrix for Logistic Regression")
plt.show()
```

```
Accuracy: 74.81940144478844
Classification Report:
               precision    recall  f1-score   support

           0       0.76      0.91      0.83       567
           1       0.72      0.55      0.62       289
           2       0.77      0.45      0.57       113

    accuracy                           0.75       969
   macro avg       0.75      0.64      0.67       969
weighted avg       0.75      0.75      0.73       969
```

Confusion matrix for Logistic Regression



Support Vector Classifier

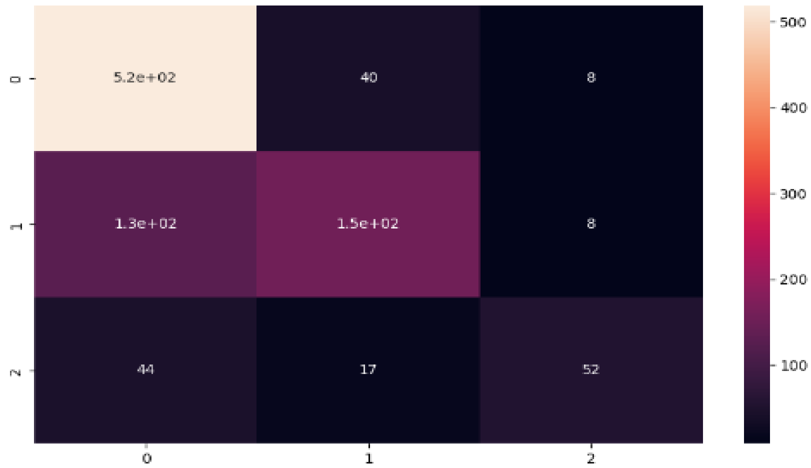
[38]:
```python
from sklearn.svm import SVC
sa_svm = SVC(C=1,kernel='sigmoid')
sa_svm.fit(X_train_tfidf, y_train)
pickle.dump(sa_svm,open("sa_svm.pkl","wb"))
y_pred_svm = sa_svm.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_svm)
print("Accuracy:", accuracy*100)
print("Classification Report:\n",classification_report(y_test,y_pred_svm))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_svm),annot=True)
plt.title("Confusion matrix for SVM")
plt.show()
```

```
Accuracy: 74.81940144478844
Classification Report:
               precision    recall  f1-score   support

           0       0.75      0.92      0.83       567
           1       0.73      0.53      0.62       289
           2       0.76      0.46      0.57       113

    accuracy                           0.75       969
   macro avg       0.75      0.64      0.67       969
weighted avg       0.75      0.75      0.73       969
```

Confusion matrix for SVM



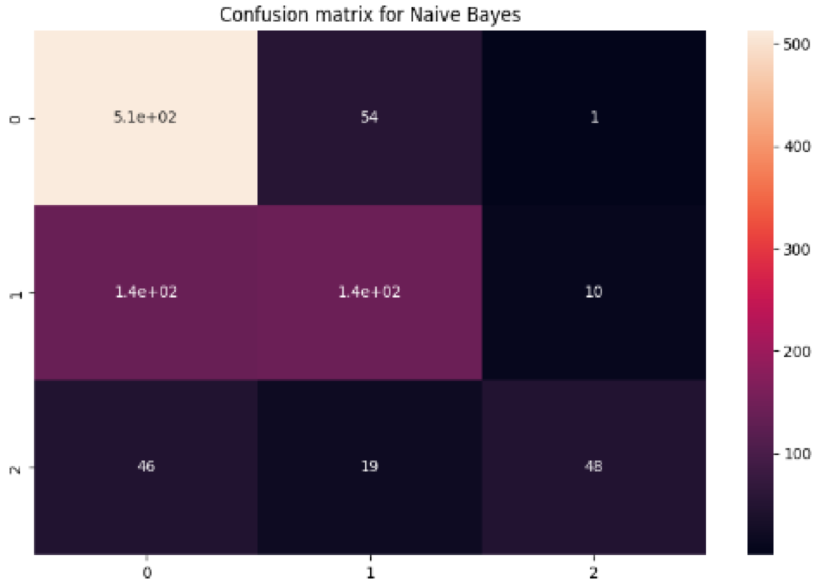
Naive Bayes

```
[45]:
from sklearn.naive_bayes import MultinomialNB
sa_nb = MultinomialNB(alpha=0.1)
sa_nb.fit(X_train_tfidf, y_train)
pickle.dump(sa_nb,open("sa_nb.pkl","wb"))
y_pred_nb = sa_nb.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_nb)
print("Accuracy:", accuracy*100)
print("Classification Report:\n",classification_report(y_test,y_pred_nb))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_nb),annot=True)
plt.title("Confusion matrix for Naive Bayes")
plt.show()
```

```
Accuracy: 72.34262125902993
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.90      0.81       567
           1       0.66      0.49      0.56       289
           2       0.81      0.42      0.56       113

    accuracy                           0.72       969
   macro avg       0.74      0.61      0.64       969
weighted avg       0.72      0.72      0.71       969
```



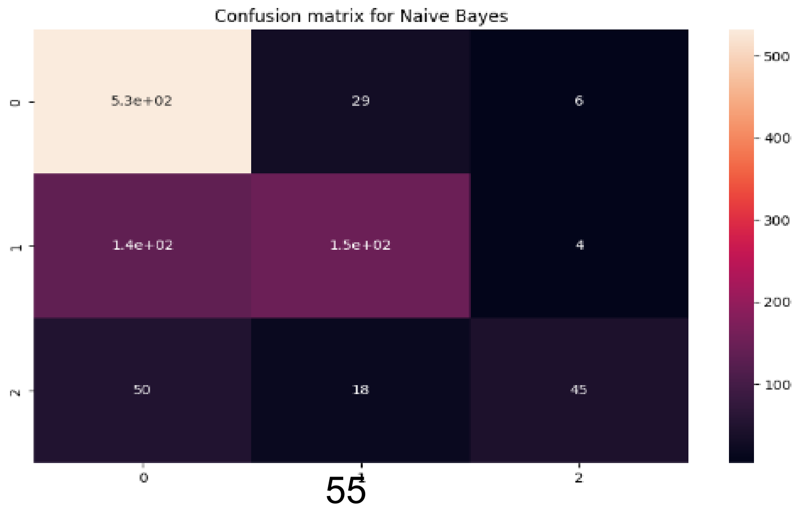
gradient boosting

```
[50]:
from sklearn.ensemble import GradientBoostingClassifier
sa_gb = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=4, random_state=42)
sa_gb.fit(X_train_tfidf, y_train)
pickle.dump(sa_gb,open("sa_gb.pkl","wb"))
y_pred_gb = sa_gb.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_gb)
print("Accuracy:", accuracy*100)
print("Classification Report:\n",classification_report(y_test,y_pred_gb))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_gb),annot=True)
plt.title("Confusion matrix for Naive Bayes")
plt.show()
```

```
Accuracy: 75.02579979360165
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.94      0.83       567
           1       0.76      0.52      0.62       289
           2       0.82      0.40      0.54       113

    accuracy                           0.75       969
   macro avg       0.77      0.62      0.66       969
weighted avg       0.76      0.75      0.73       969
```



Comparison :

```
[51]: models = ['Naive Bayes', 'Logistic Regression', 'SVM', 'Gradient Boost']
accuracy = [accuracy_score(y_test, y_pred_nb)*100,
            accuracy_score(y_test, y_pred_lr)*100,
            accuracy_score(y_test, y_pred_svm)*100,
            accuracy_score(y_test, y_pred_gb)*100]
bar_width = 0.3
index = np.arange(len(models))
plt.bar(index, accuracy, bar_width, label='Accuracy')
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Accuracy Score')
plt.xticks(index , models)
plt.ylim(0, 100)
plt.legend()
plt.show()
```

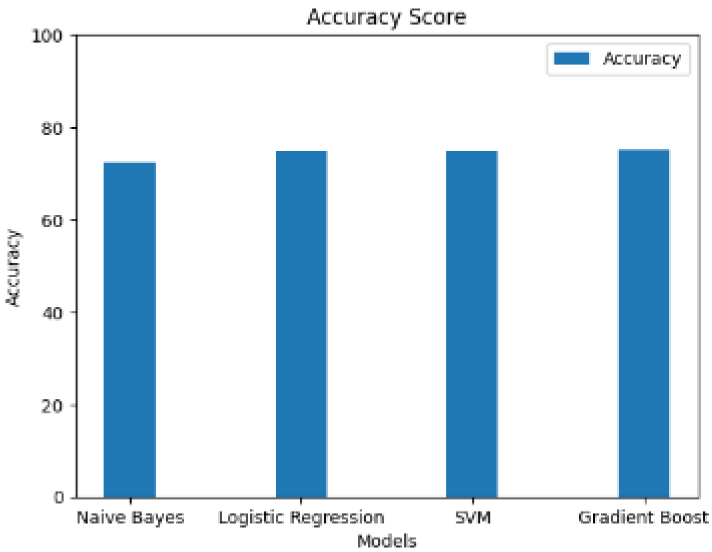


Figure 3.12.3 Accuracy of Sentimental algorithms

The bar chart compares the accuracy scores of four different models: Naive Bayes, Logistic Regression, SVM, and Gradient Boost. All models have similar accuracy, around 70-80%, indicating they perform comparably on this dataset. None of the models show a significant advantage over the others, suggesting that their ability to correctly classify instances is fairly even. However, there is still room for improvement, as accuracy is not close to 100%, indicating some misclassifications. Overall, these models provide a moderate level of prediction accuracy.

3.3.5 SPRINT RETROSPECTIVE

	Sprint Retrospective			
Liked	**Learned**	**Lacked**	**Longed For**	
Share aspects of the sprint that you enjoyed or found particularly effective.	*Discuss lessons learned, whether they are related to processes, technical aspects, or teamwork.*	*Identify areas where the team felt a lack of resources, support, or information.*	*Discuss any desires or expectations that the team had but were not met during the sprint.*	*Guidelines*
The ability to track and log sources of fake news articles was well-received, making the process more transparent.	We learned that having a dashboard to display patterns helped us identify repeated offenders efficiently.	We lacked automated blocking tools, which caused delays in action against unreliable sources.	Longed for more advanced features to automatically notify admins of frequent offenders and take proactive actions.	trends being captured and

Figure 3.20 Sprint retrospective

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Project Outcome

1. Objective

- The project aims to develop a reliable system for detecting fake news by leveraging sentiment analysis and machine learning algorithms. By analyzing the emotional tone and linguistic patterns of news articles, the system will identify misinformation and provide a classification that can help users and organizations take appropriate actions.

2. Data Processing and Feature Extraction

- Sentiment Analysis: Each news article is processed to extract emotional cues and sentiment scores (e.g., positive, negative, or neutral). This analysis helps capture emotional manipulation often used in fake news.
- Textual Feature Extraction: Using techniques such as term frequency-inverse document frequency (TF-IDF), bag-of-words, and natural language processing (NLP) features, the system will quantify and represent the text content for machine learning model input.
- Feature Transformation: Apply Fast Fourier Transform (FFT) and other transformations to enhance feature extraction, making patterns more distinguishable for the models.

3. Dataset Preparation

- Collect and label a dataset of news articles with both genuine and fake news sources.
- Split the dataset into training and testing sets to evaluate model performance on unseen data.

4. Model Comparison

- Support Vector Machine (SVM): Use SVM to classify news articles based on the extracted features. This model's results will help establish a baseline for fake news detection accuracy. However, SVM may require fine-tuning to handle complex, high-dimensional data effectively.
- k-Nearest Neighbors (k-NN): Experiment with a k-NN classifier, likely using k=3, to compare its performance with SVM. This model’s ability to capture underlying data patterns will help in distinguishing fake news based on proximity to known classes, although it may be computationally intensive with a larger dataset.
- Neural Network: Train a neural network with early stopping to prevent overfitting, as neural networks are suitable for capturing nuanced patterns in fake news articles. This model is expected to be powerful for complex detection tasks but will require significant computational resources.

5. Model Validation

- Cross-Validation: Implement k-fold cross-validation to assess the consistency and robustness of each model. This approach will ensure that models generalize well to unseen data.
- Early Stopping for Neural Networks: Apply early stopping to halt training once performance plateaus on the validation set, minimizing overfitting.

6. Real-Time Applicability

- The project explores the potential to integrate the most effective models into a real-time monitoring system, allowing for live analysis of incoming articles. Real-time applicability would be beneficial for media companies, content platforms, and fact-checkers by enabling proactive content moderation and response to misinformation.

7. Future Directions

- Hyperparameter Tuning: Future work could involve fine-tuning hyperparameters for both neural networks and SVM models to enhance their accuracy.
- Ensemble Methods: Explore ensemble techniques (e.g., stacking, boosting, bagging) to combine multiple models and improve overall prediction performance.
- Real-Time Data Analysis: Move towards utilizing real-time data, which could improve the reliability and responsiveness of the fake news detection system.

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

5.1 Conclusion

This project emphasizes the importance of credible and diverse information in the digital news landscape, addressing critical issues such as fake news, misinformation, and algorithmic bias. By utilizing a three-tiered approach that incorporates various machine learning techniques, including k-Nearest Neighbors (k-NN), Support Vector Machine (SVM), Logistic Regression (LR), and Naïve Bayes (NB), our system demonstrates the potential for accurate and robust fake news detection. Through sentiment analysis and other feature engineering techniques, the project successfully highlights patterns and indicators of misinformation, providing a strong foundation for accurate classification. The approach of dynamic model updating and continuous feedback integration ensures that the system remains relevant as it adapts to evolving news trends and emerging patterns in misinformation. This fake news detection system is a promising step toward building a more reliable, unbiased, and adaptable model that not only recognizes misinformation but also addresses the underlying complexities of modern news content. By continuously improving data collection methods, expanding model capabilities, and incorporating human feedback, this solution can significantly enhance digital trust and integrity in the information ecosystem.

5.2 Future Enhancements

1. Enhanced Data Collection and Diversity
Expanding data sources to include a broader array of viewpoints, regions, and media outlets will reduce bias and increase model reliability. Additionally, more diverse datasets will improve the model's ability to detect misinformation in different cultural and linguistic contexts, making it applicable on a global scale.

2. Advanced Feature Engineering
The use of more sophisticated feature engineering techniques, such as semantic analysis, advanced entity recognition, and deep sentiment analysis, will allow the model to capture complex relationships in the text. Such techniques can improve the accuracy of the detection system by providing deeper insight into the article's context, tone, and purpose.

3. Ensemble Learning and Hybrid Approaches
Leveraging ensemble learning by combining multiple machine learning algorithms (e.g., boosting or stacking) can improve prediction accuracy. The strength of each model can be used in tandem, resulting in a system that balances precision and recall more effectively across diverse news topics.

4. Dynamic Model Updating
Implementing a system for dynamic model updates will ensure that the fake news detection model remains relevant over time. Regular retraining on fresh data will allow the system to adapt to emerging trends in misinformation and digital content.

5. Human-in-the-Loop Validation
Introducing a human-in-the-loop approach, where expert validation is combined with machine predictions, will improve the reliability of classifications. Experts can review and verify predictions, especially for ambiguous or complex articles, contributing feedback that the model can use to learn and enhance future performance.

6. Scalability and Real-Time Detection
Enhancing the scalability of the system will enable real-time monitoring and classification of news content. This would be beneficial for media companies, social platforms, and government agencies aiming to flag or address misinformation as it emerges, making proactive interventions possible.

By implementing these future enhancements, the project can evolve into a comprehensive and adaptable system for detecting fake news, ensuring it remains responsive to the ever-changing digital news landscape. This will support a more informed public, uphold the integrity of digital content, and contribute to reducing the impact of misinformation in society.

REFERENCES

1) E. Aímeur, S. Amri and G. Brassard, "Fake news disinformation and misinformation in social media: a review", Social Network Analysis and Mining, vol. 13, no. 1, pp. 30, 2023.
2) H. Ahmed, I. Traore and S. Saad, "Detection of online fake news using n-gram analysis and machine learning techniques", Proceedings of the International Conference on Intelligent Secure and Dependable Systems in Distributed and Cloud Environments, pp. 127-138, 2017.
3) T. Wilson, J. Wiebe and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis", Proceedings of human language technology conference and conference on empirical methods in natural language processing, pp. 347-354, 2005.
4) D. N. S. B. Kavitha, P. V. G. D. Prasad Reddy and K. V. Rao, "Emotion Recognition in Tweets using Optimized Ensemble Classifiers", 2022 7th International Conference on Communication and Electronics Systems (ICCES), pp. 1728-1731, 2022.
5) P. Chikersal, S. Poria, E. Cambria, A. Gelbukh and C. E. Siong, "Modelling public sentiment in Twitter: using linguistic patterns to enhance supervised learning", International Conference on Intelligent Text Processing and Computational Linguistics, pp. 49-65, 2015.
6) O'Connor, R. Balasubramanian, B. R. Routledge and N. A. Smith, "From tweets to surveys: Linking message feeling to general assessment time series", Proc. Int. AAAI Conf. Weblogs Social Media, pp. 122-129, 2016.
7) T. Mullen and N. Collier, "Sentiment Analysis using Support Vector Machines with Diverse Information Sources", EMNLP, vol. 4, pp. 412-418, 2004.
8) R. Prabowo and M. Thelwall, "Sentiment analysis: A combined approach", Journal of Informetrics, vol. 3, no. 2, pp. 143-157, 2009.
9) Jain, A. Shakya, H. Khatter and A. K. Gupta, "A smart system for fake news detection using machine learning", 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), 2019.
10) P R Brewer, DG Young and M. Morreale, "The impact of real news about “fake news”: Intertextual processes and political satire", International Journal of Public Opinion Research, vol. 25, no. 3, pp. 323-343, 2013.

APPENDIX A

CODING

PRE PROCESSING FOR MACHINE LEARNING TECHNIQUES:

Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.corpus import stopwords
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score,classification_report,confusion_matrix
#For not getting warning messages
import warnings
warnings.filterwarnings('ignore')
```

Dataset

```
df_train=pd.read_csv("C:\\Users\\Sravya Kamineni\\Documents\\train_preprocessed_lemmatized.csv")
df_test=pd.read_csv("C:\\Users\\Sravya Kamineni\\Documents\\test_preprocessed_lemmatized.csv")
df=pd.concat([df_train,df_test],axis=0)
df.head()
```

	Unnamed: 0	text	Class Index
0	0	wall st bear claw back black reuter reuters sh...	3
1	1	carlyle look toward commercial aerospace reute...	3
2	2	oil economy cloud stock outlook reuters reuter...	3
3	3	iraq halt oil export main southern pipeline re...	3
4	4	oil price soar alltime record pose new menace ...	3

Train Test Splitting(80-20%)

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(df["text"],df["Class Index"],test_size=0.2,random_state=42)
```

Missing Values

```
df_train.isnull().sum()
```

```
Class Index    0
Title          0
Description    0
dtype: int64
```

```
df=pd.concat([df_train,df_test],axis=0)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 127600 entries, 0 to 7599
Data columns (total 3 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Class Index  127600 non-null  int64
 1   Title        127600 non-null  object
 2   Description  127600 non-null  object
dtypes: int64(1), object(2)
memory usage: 3.9+ MB
```

Data Balance Checking

```python
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
df["category"]=df["Class Index"].map({1:"World",2:"Sports",3:"Business",4:"Sci/Tech"})
count=pd.DataFrame(df["category"].value_counts()).reset_index()
sns.set_style('darkgrid')
plt.figure(figsize=(10, 6))
print(count.head())
sns.barplot(data=count, y='count', x='category', palette='Dark2')
plt.title('No. news in each category', loc='left', fontsize=20)
plt.xlabel("")
plt.ylabel("")
plt.show()
```

```
   category  count
0  Business  31900
1  Sci/Tech  31900
2    Sports  31900
3     World  31900
```

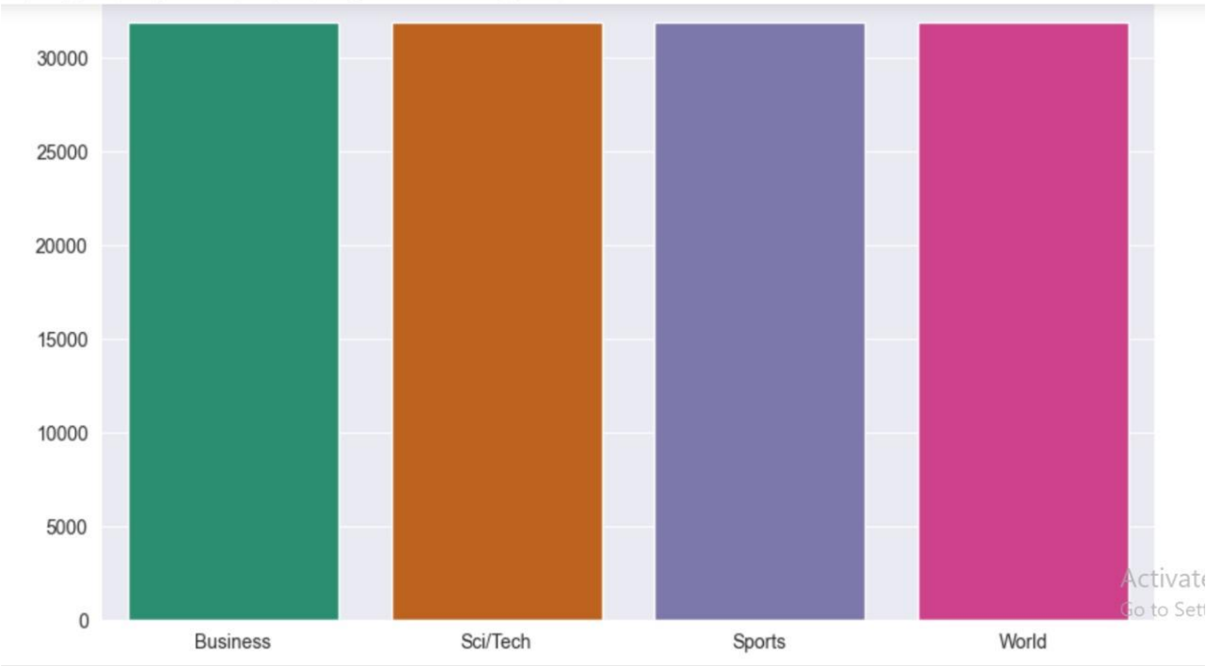


Fig 3.1 No. of News in Each Category

Tokenization

```python
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
def preprocess_text(text):
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub(r'\d+', '', text)
    tokens = word_tokenize(text)
    tokens = [token.lower() for token in tokens]
    stop_words = set(stopwords.words('english'))
    tokens = [token for token in tokens if token not in stop_words]
    preprocessed_text = ' '.join(tokens)
    return preprocessed_text
df_train['headline']=df_train['Title']+" "+df_train['Description']
df_train['headline'] = df_train['headline'].apply(preprocess_text)
df_test['headline']=df_test['Title']+" "+df_test['Description']
df_test['headline'] = df_test['headline'].apply(preprocess_text)
```

Lemmatization

```python
import spacy
nlp = spacy.load("en_core_web_sm")
def lemmatize_text(text):
    doc = nlp(text)
    lemmatized_text = " ".join([token.lemma_ for token in doc])
    return lemmatized_text
df_train['text'] = df_train['headline'].apply(lemmatize_text)
df_test['text'] = df_test['headline'].apply(lemmatize_text)
df_train[["text","Class Index"]].to_csv("train_preprocessed_lemmatized.csv")
df_test[["text","Class Index"]].to_csv("test_preprocessed_lemmatized.csv")
```

Feature Extraction

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_df=0.7,ngram_range=(1,2),stop_words='english')
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

Algorithms:
News Classification

LR

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(C=1,solver="saga")
lr.fit(X_train_tfidf, y_train)
y_pred_lr = lr.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_lr)
print("Accuracy:", accuracy)
print("Classification Report:\n",classification_report(y_test,y_pred_lr))
sns.heatmap(confusion_matrix(y_test,y_pred_lr))
```

SVM

```
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train_tfidf, y_train)
y_pred_svm = svm.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_svm)
print("Accuracy:", accuracy)
#previous:89.98
```

KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=43)
knn.fit(X_train_tfidf, y_train)
y_pred_knn = knn.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_knn)
print("Accuracy:", accuracy)
#previous:83.9
```

Multinomial Naïve Bayes

```
from sklearn.naive_bayes import MultinomialNB
nc_nb = MultinomialNB(alpha=0.1)
nc_nb.fit(X_train_tfidf, y_train)
pickle.dump(nc_nb,open("nc_nb.pkl","wb"))
y_pred_nb = nc_nb.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_nb)
print("Accuracy:", accuracy)
```

```
print("Classification Report:\n",classification_report(y_test,y_pred_nb))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_nb),annot=True)
plt.title("Confusion matrix for Naive Bayes")
plt.show()
```

Comparison

```
models = ['Naive Bayes', 'Logistic Regression', 'SVM', 'KNN']
accuracy = [accuracy_score(y_test, y_pred_nb)*100,
            accuracy_score(y_test, y_pred_lr)*100,
            accuracy_score(y_test, y_pred_svm)*100,
            accuracy_score(y_test, y_pred_knn)*100]
bar_width = 0.3
index = np.arange(len(models))

plt.bar(index, accuracy, bar_width, label='Accuracy')
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Accuracy Score')
plt.xticks(index , models)
plt.ylim(80, 100)
plt.legend()
plt.show()
```

Fake News Detection

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
fn_rf = RandomForestClassifier()
fn_rf.fit(X_train_tfidf,y_train)
pickle.dump(fn_rf,open("fn_rf.pkl","wb"))
y_pred_rf=fn_rf.predict(X_test_tfidf)
print("Accuracy:",accuracy_score(y_pred_rf,y_test)*100)
print("Classification Report:\n",classification_report(y_test,y_pred_rf))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_rf),annot=True)
plt.title("Confusion matrix for Random Forest")
plt.show()
```

KNN

```
from sklearn.neighbors import KNeighborsClassifier
fn_knn = KNeighborsClassifier(n_neighbors=5)
fn_knn.fit(X_train_tfidf, y_train)
pickle.dump(fn_knn,open("fn_knn.pkl","wb"))
y_pred_knn = fn_knn.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_knn)
print("Accuracy:", accuracy*100)
print("Classification Report:\n",classification_report(y_test,y_pred_knn))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_knn),annot=True)
plt.title("Confusion matrix for K-nn")
plt.show()
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
fn_dt = DecisionTreeClassifier()
fn_dt.fit(X_train_tfidf, y_train)
pickle.dump(fn_dt,open("fn_dt.pkl","wb"))
y_pred_dt = fn_dt.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_dt)
print("Accuracy:", accuracy*100)
print("Classification Report:\n",classification_report(y_test,y_pred_dt))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_dt),annot=True)
plt.title("Confusion matrix for Decision Tree")
plt.show()
```

AdaBoost

```
from sklearn.ensemble import AdaBoostClassifier
fn_ab = AdaBoostClassifier()
fn_ab.fit(X_train_tfidf, y_train)
pickle.dump(fn_ab,open("fn_ab.pkl","wb"))
y_pred_ab = fn_ab.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_ab)
print("Accuracy:", accuracy*100)
print("Classification Report:\n",classification_report(y_test,y_pred_ab))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_ab),annot=True)
plt.title("Confusion matrix for AdaBoost")
plt.show()
```

Comaprision

```
models = ['Random Forest', 'KNN', 'Decision Tree', 'AdaBoost']
accuracy = [accuracy_score(y_test, y_pred_rf)*100,
            accuracy_score(y_test, y_pred_knn)*100,
            accuracy_score(y_test, y_pred_dt)*100,
            accuracy_score(y_test, y_pred_ab)*100]
bar_width = 0.3
index = np.arange(len(models))

plt.bar(index, accuracy, bar_width, label='Accuracy')
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Accuracy Score')
plt.xticks(index , models)
plt.ylim(80, 100)
plt.legend()
plt.show()
```

Sentimental Analysis

LR

```
from sklearn.linear_model import LogisticRegression
sa_lr=LogisticRegression(C=1,solver="saga")
sa_lr.fit(X_train_tfidf, y_train)
pickle.dump(sa_lr,open("sa_lr.pkl","wb"))
y_pred_lr = sa_lr.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_lr)
print("Accuracy:", accuracy*100)
print("Classification Report:\n",classification_report(y_test,y_pred_lr))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_lr),annot=True)
plt.title("Confusion matrix for Logistic Regression")
plt.show()
```

Support Vector Classifier

```
from sklearn.svm import SVC
sa_svm = SVC()
sa_svm.fit(X_train_tfidf, y_train)
pickle.dump(sa_svm,open("sa_svm.pkl","wb"))
y_pred_svm = sa_svm.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_svm)
print("Accuracy:", accuracy*100)
print("Classification Report:\n",classification_report(y_test,y_pred_svm))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_svm),annot=True)
plt.title("Confusion matrix for SVM")
plt.show()
```

Naïve Bayes

```
from sklearn.naive_bayes import MultinomialNB
sa_nb = MultinomialNB(alpha=0.1)
sa_nb.fit(X_train_tfidf, y_train)
pickle.dump(sa_nb,open("sa_nb.pkl","wb"))
y_pred_nb = sa_nb.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_nb)
print("Accuracy:", accuracy*100)
print("Classification Report:\n",classification_report(y_test,y_pred_nb))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_nb),annot=True)
plt.title("Confusion matrix for Naive Bayes")
plt.show()
```

Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier
sa_gb = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)
sa_gb.fit(X_train_tfidf, y_train)
pickle.dump(sa_gb,open("sa_gb.pkl","wb"))
y_pred_gb = gb_clf.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred_gb)
print("Accuracy:", accuracy*100)
print("Classification Report:\n",classification_report(y_test,y_pred_gb))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,y_pred_gb),annot=True)
plt.title("Confusion matrix for Naive Bayes")
plt.show()
```

Comparision

```
models = ['Naive Bayes', 'Logistic Regression', 'SVM', 'Gradient Boost']
accuracy = [accuracy_score(y_test, y_pred_nb)*100,
            accuracy_score(y_test, y_pred_lr)*100,
            accuracy_score(y_test, y_pred_svm)*100,
            accuracy_score(y_test, y_pred_gb)*100]
bar_width = 0.3
index = np.arange(len(models))
plt.bar(index, accuracy, bar_width, label='Accuracy')
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Accuracy Score')
plt.xticks(index , models)
plt.ylim(80, 100)
plt.legend()
plt.show()
```

Output

ACCURACY:

News Classification:

Algorithms	Accuracy
Knn	89.87
Svm	91.98
LR	91.70
NB	91.77

Table 4.1 Accuracy for News Classification

Fake News Detection:

Algorithms	Accuracy
Knn	82.52
Decision Tree	94.16
AdaBoost	93.17
Random Forest	93.19

Table 4.2 Accuracy for News detection

Sentiment Analysis:

Algorithms	Accuracy
LR	74.81
SVC	74.82
Naïve Bayes	72.34
Gradient Boosting	75.02

Table 4.3 Accuracy for Sentiment Analysis

REFERENCES

[1] Ning, X., and Lovell, M. R., “On the Sliding Friction Characteristics of Unidirectional Continuous FRP Composites,” ASME J. Tribol., 124(1), pp. 5-13, 2002.

[2] Barnes, M., “Stresses in Solenoids,” J. Appl. Phys., 48(5), pp. 2000–2008, 2001.

[3] Jones, J., (2000), Contact Mechanics, Cambridge University Press, Cambridge, UK, Chap. 6.

[4] Lee, Y., Korpela, S. A., and Horne, R. N., “Structure of Multi-Cellular Natural Convection in a Tall Vertical Annulus,” Proc. 7th International Heat Transfer Conference, U. Grigul et al., eds., Hemisphere, Washington, DC, 2, pp. 221–226, 1982.

[5] Hashish, M., “600 MPa Waterjet Technology Development,” High Pressure Technology, PVP-Vol. 406, pp. 135-140, 2000.

[6]Watson, D. W., “Thermodynamic Analysis,” ASME Paper No. 97-GT-288, 1997. [7] Tung, C. Y., (1982), “Evaporative Heat Transfer in the Contact Line of a Mixture,” Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, NY.