НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Система виявлення фейкових новин з використанням методів машинного навчання»

Виконала:
студентка IV курсу, групи ІС-12
Мельникова Катерина Олександрівна _____

Керівник:
Асистент кафедри
Коломоєць Сергій Олексійович _____

Рецензент:
професор каф. ШІ, д.т.н., професор
Чумаченко Олена Іллівна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2025 р.

ЗАВДАННЯ

на дипломний проєкт студентці

Мельниковій Катерині Олександрівні

1. Тема проєкту «Система виявлення фейкових новин з використанням методів машинного навчання», керівник проєкту Коломоєць Сергій Олексійович, асистент кафедри, затверджені наказом по університету від «23» травня 2025р. № 1705-с

2. Термін подання студентом проєкту: «9» червня 2025р.

3. Вихідні дані до проєкту: мова програмування Python, фреймворк FastAPI, бібліотеки: scikit-learn, tensorflow, nltk, датасет ISOT

4. Зміст пояснювальної записки:

1) загальні положення;
2) вибір технологій розробки;
3) математичне забезпечення;
4) розробка інформаційної системи;
5) тестування та огляд готової системи.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо):

1) діаграма варіантів використання;
2) діаграма послідовностей;
3) діаграма компонентів;
4) діаграма діяльності.

6. Дата видачі завдання: 15.03.2025

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Аналіз предметної області	15.03.2025	
2.	Аналіз готових рішень	20.03.2025	
3.	Формування вимог до системи	09.04.2025	
4.	Математичне забезпечення	18.04.2025	
5.	Вибір технологій розробки	27.04.2025	
6.	Розробка інформаційної системи	15.05.2025	
7.	Тестування системи та оцінка результатів класифікації на тестових даних	29.05.2025	
8.	Висновки	01.06.2025	

Студентка Катерина МЕЛЬНИКОВА

Керівник Сергій КОЛОМОЄЦЬ

АНОТАЦІЯ

Система виявлення фейкових новин з використанням методів машинного навчання.

Проєкт містить 68 с. тексту, 22 рисунки, 1 таблицю, посилання на 15 літературних джерел, додатки та 4 конструкторські документи.

Ключові слова: фейкові новини, машинне навчання, класифікація тексту, векторизація, обробка природної мови, нейронні мережі, TF-IDF, LSTM, SVM, Байєсів класифікатор, логістична регресія, Bi-LSTM, GRU, випадкові ліси.

Об'єктом дослідження є інформаційна система для автоматичного виявлення фейкових новин на основі аналізу текстових даних.

Метою роботи є підвищення точності автоматичної класифікації новин на достовірні та фейкові за допомогою сучасних методів машинного навчання.

У дипломному проєкті розроблено та порівняно ефективність класичних моделей (Naive Bayes, Logistic Regression, SVM, Random Forest) і нейронних мереж (LSTM, GRU, Bi-LSTM) для задачі виявлення фейкових новин. Реалізовано повний цикл обробки даних: очищення тексту, векторизація (TF-IDF), токенізація, підготовка до навчання та оцінювання результатів за допомогою метрик точності, повноти та F1-міри. Також було розроблено зручний вебінтерфейс для взаємодії користувача з системою.

Результати дипломного проєкту можуть бути використані в медіасервісах, агрегаторах новин або фактчекінгових платформах для автоматичного попереднього фільтрування потенційно фейкової інформації.

SUMMARY

Fake news detection system using machine learning methods.

The project consists of 68 pages of text, 22 figures, 1 table, references to 15 literary sources, appendices, and 4 design documents.

Keywords: fake news, machine learning, text classification, vectorization, natural language processing, neural networks, TF-IDF, LSTM, SVM, Naive Bayes classifier, logistic regression, Bi-LSTM, GRU, random forest.

The object of the study is an information system for automatic fake news detection based on text data analysis.

The aim of the project is to improve the accuracy of automatic classification of news as true or fake using modern machine learning methods.

The thesis develops and compares the effectiveness of classical models (Naive Bayes, Logistic Regression, SVM, Random Forest) and neural networks (LSTM, GRU, Bi-LSTM) for the task of fake news detection. A complete data processing pipeline is implemented: text cleaning, vectorization (TF-IDF), tokenization, training preparation, and result evaluation using precision, recall, accuracy and F1-score metrics. A user-friendly web interface was also developed to enable user interaction with the system.

The results of this thesis can be applied in media services, news aggregators, or fact-checking platforms to automatically pre-filter potentially fake information.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІС12.220БАК.005 ПЗ	Пояснювальна записка	68		
6	А3	ІС12.220БАК.005 Д1	Система виявлення фейкових	1		
7			новин з використанням мето-			
8			дів машинного навчання.			
9			Діаграма варіантів використа-			
10			ння.			
11	А3	ІС12.220БАК.005 Д2	Система виявлення фейкових	1		
12			новин з використанням мето-			
13			дів машинного навчання.			
14			Діаграма послідовностей.			
15						
16	А3	ІС12.220БАК.005 Д3	Система виявлення фейкових	1		
17			новин з використанням мето-			
18			дів машинного навчання.			
19			Діаграма компонентів.			
20						
21	А3	ІС12.220БАК.005 Д4	Система виявлення фейкових	1		
22			новин з використанням мето-			
23			дів машинного навчання.			
24			Діаграма діяльності.			
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІС12.220БАК.005 ТП		
Розроб.		Мельникова К.О.			Система виявлення фейкових новин з використанням методів машинного навчання. Відомість проєкту	Літ.	Аркуш 1 / Аркушів 1
Керівн.		Коломоєць С.О.				КПІ ім. Ігоря Сікорського. Група ІС-12	
Затв.							

Пояснювальна записка
до дипломного проєкту
на тему: «Система виявлення фейкових новин з використанням методів машинного навчання»

Київ – 2025 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API — Application Programming Interface

Bi-LSTM — Bidirectional Long Short-Term Memory

FN — False Negative

FP — False Positive

GRU — Gated Recurrent Unit

LSTM — Long Short-Term Memory

TN — True Negative

TP — True Positive

ВСТУП

В сучасних умовах стрімкого розвитку інформаційних технологій та поширеності соціальних мереж люди отримали широкий та майже миттєвий доступ до будь-якої інформації. Проте, разом з очевидними перевагами людство стикнулося також з менш очевидними негативними наслідками цього явища одним з яких є поширення фейкової інформації. Ні для кого не секрет, що далеко не вся інформація подана у ЗМІ є правдивою та цінною, проте, далеко не кожен здатен та має необхідну кількість ресурсів щоб критично оцінити представлену перед ним інформацію.

Особливо гостро дана проблема постала під час повномасштабної війни в Україні. Дезінформація широко використовується для дестабілізації ситуації в суспільстві, поширення паніки, дискредитації діяльності державних та благодійних установ і спроб вплинути на внутрішню та міжнародну позицію України. В таких умовах здатність швидко відрізняти фейкові новини від достовірних є критично важливою.

У зв’язку з великими обсягами інформації та впливом людського фактору ручна перевірка правдивості представлених новин стає малоефективною. Це створює передумови для автоматизації даного процесу. Метою даної роботи є аналіз існуючих рішень та розробка системи, що буде здатна автоматично виокремлювати фейкові новини на основі проаналізованого тексту.

Для вирішення поставленої задачі буде застосовано різноманітні методи машинного навчання, зокрема нейронні мережі, що показують високу ефективність у задачах обробки природної мови. Завдяки здатності виявляти приховані закономірності у текстах, моделі на основі штучних нейронних мереж можуть успішно розпізнавати фейкові повідомлення, враховуючи як лексичні, так і стилістичні особливості тексту.

Окрім нейронних мереж в рамках даної роботи також буде розглянуто різноманітні класичні моделі машинного навчання, зокрема: логістична регресія, метод опорних векторів, випадкові ліси та наївний Байєсів класифікатор. Не

зважаючи на свою відносну простоту дані методи показали свою безумовну ефективність в задачах обробки природньої мови та навчання з вчителем.

Окрім впровадження та порівняльного аналізу різноманітних методів машинного навчання для здійснення класифікації фейкових новин, також планується імплементація вебзастосунку, що дозволить користувачам легко взаємодіяти з натренованими моделями та швидко отримувати результати класифікації.

Загалом, даний дипломний проєкт матиме не лише наукове, але й практичне значення для суспільства та може бути використана в багатьох сферах таких як: медіа, освіта та державне управління а також у громадському секторі для підвищення інформаційної грамотності населення, забезпечення доступу до достовірних джерел та зміцнення критичного мислення.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметної області

З розвитком інформаційних технологій стрімко зросла кількість каналів поширення інформації, що в свою чергу призвело до спрощення процесу впливу на суспільну думку, хід прийняття рішень в суспільстві та формування політики. Новинний контент поширюється як через традиційні медіа, так і через цифрові платформи: новинні сайти, соціальні мережі та месенджери.

Основна діяльність в цій сфері полягає у створенні, редагуванні, публікації та поширенні новинних повідомлень. У більшості випадків оцінка правдивості інформації виконується вручну редакторами, журналістами або навіть кінцевими споживачами. Однак зростання обсягів новин, швидкість їх появи та різноманітність джерел поширення інформації, що не завжди мають високий рівень довіри, ускладнюють процес верифікації. У зв’язку з цим виникає потреба в автоматизованій системі, здатній аналізувати текст новинного повідомлення та давати оцінку його достовірності. Процес діяльності такої системи включає наступні етапи:

1) збір новинних матеріалів із відкритих джерел;
2) попередню обробку текстів (очищення та нормалізація датасету);
3) застосування моделей машинного навчання до оброблених даних;
4) класифікація нових даних.

Розроблена система дозволить спростити процес оцінки правдивості новин та підвисить швидкість реагування на дезінформацію.

1.2 Постановка задачі

1.2.1 Призначення системи

Розроблювана система призначена для автоматизованої класифікації новинних повідомлень з метою виявлення фейкових матеріалів. Вид діяльності, що автоматизується, — аналітична обробка текстової інформації, а також моніторинг

достовірності новинного контенту, що публікується в мережі в інформаційних ресурсах (у вебвиданнях та соціальних мережах).

Об’єктами автоматизації є:

– тексти новин, що надходять з відкритих джерел;

– процес перевірки достовірності новини;

– процес прийняття рішення щодо наявності ознак неправдивості.

Система має забезпечити можливість автоматичного аналізу тексту, попередньої обробки (очищення, нормалізації), застосування алгоритмів машинного навчання (в тому числі нейронних мереж), класифікації повідомлень, а також формування результатів, придатних для подальшої перевірки або використання людиною.

Основна мета автоматизації — зменшення впливу людського фактору та підвищення ефективності та швидкості виявлення фейкової інформації у великому потоці новинного контенту.

1.2.2 Цілі та задачі розробки

Мета розробки: підвищення ефективності виявлення фейкових новин шляхом створення системи автоматизованої класифікації текстів з використанням методів машинного навчання та нейронних мереж а також порівняння та оцінка ефективності різноманітних методів машинного навчання для вирішення поставленої задачі.

Завдання, що вирішуються в рамках проєкту:

1) провести аналіз предметної області та огляд типових ознак фейкових новин;

2) проаналізувати існуючі методи машинного навчання для автоматичної класифікації новин;

3) обґрунтувати вибір методів машинного навчання, які можуть бути ефективними для вирішення задачі класифікації новин;

4) підготувати релевантний датасет;

5) сформулювати вимоги до інформаційної системи;

6) розробити архітектуру системи виявлення фейкових новин;

7) реалізувати прототип системи, що здійснює автоматичну класифікацію новинних текстів;

8) провести тестування розробленої системи на реальних або наближених до реальності даних;

9) оцінити якість класифікації за допомогою відповідних метрик (accuracy, precision, recall, F1-score).

1.3 Огляд наявних аналогів

До найбільш поширених методів машинного навчання, що використовуються для класифікації фейкової інформації відносяться: логістична регресія, дерева рішень, випадкові ліси, метод опорних векторів (SVM) а також нейронні мережі, зокрема LSTM, Bi-LSTM, BERT і трансформери.

Попри активні розробки, більшість існуючих рішень мають низку обмежень. Часто системи орієнтовані на конкретне джерело або формат даних (наприклад, твіт, заголовок чи коротку новинну замітку). Крім того, деякі підходи не враховують семантичні й стилістичні особливості мови, що знижує їх ефективність при зміні джерел або тематики.

Детальний опис готових рішень наведено в таблиці 1.1.

Таблиця 1.1 — Порівняльний аналіз існуючих методів машинного навчання

Автор(-и)	Метод(-и)	Дані	Результати дослідження
Jumana Jouhar, Anju Pratap, Neharin Tijo, Meenakshi Mony [1]	Логістична регресія, Дерева рішень, Випадкові ліси, Gradient Boosting,	ISOT fake news dataset	Найкращі результати класифікації продемонстрував алгоритм XGBoost, забезпечивши найвищі показники Testing Accuracy, Recall, F1-Score

Автор(-и)	Метод(-и)	Дані	Результати дослідження
	Passive Aggressive Classifier, XGBoost		та AUC. Проте, всі інші методи також показали непогані результати зі значеннями усіх метрик >0.97
Monther Aldwairi, Ali Alwahedi [2]	Bayes Net, Логістична регресія, Дерева рішень, Випадкові ліси	Дані текстів і заголовків постів з соцмереж	Всі класифікатори показали досить високі значення застосованих метрик (Precision, Recall, F-Score і ROC) >0.94 проте найкращі результати показала логістична регресія зі значенням Precision 99.4%
Rakesh Dutta, Mukta Majumder [3]	BERT, BERT + LSTM, BERT + LSTM (+ attention layer), BERT + ABiLSTM	COVID-19 rumor dataset	У ході дослідження найкращі результати були отримані за допомогою моделі BERT + ABiLSTM (80% Precision та F-measure, 81% Recall та Accuracy). Модель в основному складається з трьох рівнів: BERT (для векторизації), BiLSTM (витягує семантичні та часові характеристики з контексту) і Attention Layer (відповідає за визначення слів, які

Автор(-и)	Метод(-и)	Дані	Результати дослідження
			семантично пов’язані з контекстом, щоб сприяти загальному розумінню тексту).
Vysotska Victoria, Przystupa Krzysztof, Chyrun Lyubomyr, Vladov Serhii, Ushenko Yuriy, Uhryn Dmytro, Hu Zhengbing [4]	Логістична регресія, дерева рішень, KNN, випадкові ліси, багатошаровий перцептрон, наївний байєсівський класифікатор.	Власний корпус текстів онлайн повідомлень з прикладами фейків та пропаганди (точне джерело не вказано)	Усі протестовані класифікатори показали високі результати, точність класифікації перевищила 93% у більшості випадків. Найкращу ефективність при класифікації фейків продемонстрували алгоритм k-найближчих сусідів (точність 99.9%) та логістична регресія (99.88%). У випадку розпізнавання нефейкових новин найвищі результати показали баєсівський класифікатор (99.8%) та багатошаровий перцептрон (99.79%).
Putra Wanda, Mohammad Diqi [5]	Generative round network (GRN)	Indonesian "Fake News" dataset	У ході дослідження GRN показав кращі результати ніж GAN (97,98% Precision, 99,39% Recall, 98,69% F1). GRN — модифікована GAN-

Автор(-и)	Метод(-и)	Дані	Результати дослідження
			архітектура з додатковою round-функцією у генераторі та дискримінаторі для кращого розрізнення фейкових новин.

Результати порівняльного аналізу проведеного в даному підрозділі показують, що різні методи машинного навчання мають різну ефективність в залежності від кожного конкретного датасету — його мови, тематики, структури текстів та джерела даних.

Методи машинного навчання (логістична регресія, дерева рішень, KNN і т.п.) зазвичай є ефективними, коли присутня чітка структура даних та велика кількість чітко визначних ознак. Також використання подібних методів є доцільним коли необхідно швидко отримати результат з обмеженими ресурсами або на невеликих наборах даних. Крім цього, такі методи також мають свої переваги у випадках коли важлива інтерпретація результатів, оскільки вони часто забезпечують зрозумілі моделі, де кожен параметр можна легко інтерпретувати.

Нейронні мережі, зокрема глибокі моделі, такі як LSTM, BiLSTM або трансформери зазвичай показують набагато кращі результати при обробці великих за обсягом неструктурованих даних. Ці методи є набагато ефективнішими, коли потрібно враховувати складні взаємозв’язки в даних. Такі як: контекстні залежності в текстах або складні патерни, що не можна виявити через прості ознаки. Проте, нейронні мережі вимагають значних обчислювальних ресурсів і часто потребують великих обсягів даних для досягнення оптимальних результатів.

Також важливо враховувати те як кожен метод справляється з класифікацією фейкових і справжніх новин, так як в деяких випадках певні моделі можуть краще справлятися з виявленням фейкових новин, але при цьому давати гірші результати

класифікації справжніх новини, і навпаки. Тому важливо враховувати значення різних метрик при оцінці моделей.

В рамках дипломного проєкту, доцільним є власна розробка та порівняльний аналіз моделей, орієнтованих на задану предметну область і тип вхідних даних.

1.4 Формулювання вимог до системи

1.4.1 Вимоги до системи в цілому

В даному розділі визначимо вимоги до структури та функціонування системи в цілому. Систему що розробляється можна розбити на наступні підсистеми:

– підсистема обробки тексту забезпечує попередню обробку текстових даних: очищення, нормалізацію та токенізацію;

– підсистема класифікації виконує аналіз тексту та класифікує його як справжню або фейкову новину за допомогою алгоритмів машинного навчання;

– підсистема взаємодії з користувачем призначена для забезпечення інтерфейсу для введення тексту і відображення результатів аналізу.

Основною характеристикою системи є модульність. Кожна підсистема може бути легко модифікована або замінена без впливу на інші підсистеми.

Основним режимом функціонування є інтерактивний режим. В даному режимі аналіз даних відбувається за запитом користувача;

Вимоги до надійності системи:

1) використання перевірених бібліотек і фреймворків для обробки тексту та побудови моделей машинного навчання (таких як scikit-learn, TensorFlow, spaCy та інші);

2) обробка помилок та інших виняткових ситуацій (наприклад введення некоректного вхідного тексту завершується виведенням відповідного повідомлення і коректним завершенням роботи програми);

3) коректна обробка помилок та ведення логування для подальшої їх обробки;

4) забезпечення простого перезапуску системи без необхідності ручного відновлення стану, даних або необхідності перенавчання моделі.

1.4.2 Вимоги до функціональних характеристик

Система, що розробляється має виконувати всі перераховані нижче функції.

Попередня обробка тексту. Перед тим як аналізувати дані необхідно привести їх в зручний для обробки формат. Це включає в себе очищення даних від зайвого «шуму» (такого як пунктуація, частини html-коду та інші зайві символи), нормалізація (приведення тексту до нижнього регістру та видалення стоп-слів) і токенізація.

Векторизація. Векторизація — це процес перетворення даних, що знаходяться у «сирому» текстовому форматі на вектори чисел, що можуть бути сприйняті і оброблені моделлю.

Класифікація. Дана функція є основною у системі ідентифікації фейкових новин. На цьому етапі використовуються попередньо натреновані моделі, що аналізуючи текст отриманий від користувача видають відповідне рішення щодо того чи є подана новина фейковою.

Відображення результатів. Система має чітко і зрозуміло виводити результат у вигляді текстового повідомлення з класифікацією новини.

При цьому, система має працювати достатньо швидко та якісно: середній час аналізу одного тексту має бути до 3 секунд та точність класифікації на тестових даних не менше 85%.

1.4.3 Вимоги до видів забезпечення

Математичне забезпечення. Для побудови моделі що вирішуватиме поставлену задачу класифікації планується застосування таких підходів:

1) наївний байєсівський класифікатор — простий але ефективний алгоритм машинного навчання що базується на теоремі Байєсса та припущенні щодо незалежності ознак;

2) логістична регресія — базовий метод для бінарної класифікації текстів;

3) метод опорних векторів (SVM) — модель, здатна визначати оптимальні гіперплощини, що розділяють класи текстів;

4) випадковий ліс (Random Forest) — модель, що використовує ансамбль дерев рішень;

5) нейронні мережі — для захоплення складніших нелінійних залежностей у текстах, зокрема із застосуванням простих багатошарових перцептронів та рекурентних нейронних мереж (LSTM, Bi-LSTM та GRU).

Планується провести порівняльний аналіз вищезгаданих методів за наступними метриками: precision, recall, accuracy та F1-score, та обрати найбільш ефективний підхід для подальшого впровадження в систему.

Крім того, для підготовки даних передбачено використання базових алгоритмів обробки тексту:

– очищення тексту від шумів (спецсимволів, стоп-слів тощо);

– токенізація;

– векторизація за допомогою методів TF-IDF, Word2Vec та інших.

Інформаційне забезпечення. Інформаційне забезпечення системи передбачає роботу з текстовими даними без залучення зовнішніх баз даних. Для навчання моделей буде використано відкриті датасети новин, що містять готові розмічені дані. Вхідні та проміжні дані оброблятимуться безпосередньо в оперативній пам’яті та зберігатимуться у файловій системі у форматі .csv.

Програмне забезпечення. Основні вимоги до програмного забезпечення:

– мови програмування: Python 3.10+ для розробки моделей та аналізу даних:

– бібліотеки для машинного навчання: scikit-learn — для реалізації базових моделей (логістична регресія, Random Forest, SVM) та інструментів попередньої обробки даних; TensorFlow та PyTorch — для створення і навчання нейронних мереж, зокрема моделей на базі LSTM;

– інструменти роботи з даними: Pandas, NumPy — для обробки та аналізу даних;

– інструменти візуалізації: Matplotlib, Seaborn — для побудови графіків результатів моделювання і аналізу ефективності роботи алгоритмів;

– середовище розробки: VS Code.

Висновки до розділу 1

В даному розділі було здійснено аналіз предметної області системи виявлення фейкових новин, проведено аналіз існуючих аналогів, визначено цілі та задачі розробки а також вимоги до системи в цілому, її функціональних характеристик а також інформаційного та математичного забезпечення.

Визначено, що система, яка розробляється, має автоматизувати попередній аналіз текстових новин і виявляти потенційно фейкові повідомлення. Це дозволить зменшити навантаження на фахівців з фактчекінгу, а також підвищити загальний рівень інформаційної гігієни в суспільстві. Основною метою є створення інструменту, що на основі тексту новини зможе з високою точністю визначити, чи є вона фейковою. Для цього передбачено використання методів машинного навчання, зокрема моделей нейронних мереж.

Також було сформульовано ключові задачі, що буде необхідно вирішити під час реалізації проєкту: зібрати та підготувати релевантний датасет, провести аналіз ознак фейкових новин, обрати оптимальні алгоритми для класифікації, розробити прототип системи та перевірити його ефективність. З огляду на те, яку шкоду може завдавати фейкова інформація, особливо в умовах війни, створення подібної системи є не просто актуальним, а дійсно необхідним.

Окрім цього було проведено аналіз існуючих аналогів, результати якого показали, що різні методи машинного навчання мають різну ефективність в залежності від кожного конкретного датасету — його мови, тематики, структури текстів та джерела даних.

Методи машинного навчання (логістична регресія, дерева рішень, KNN і т.п.) зазвичай є ефективними, коли присутня чітка структура даних та велика кількість чітко визначних ознак. Також використання подібних методів є доцільним коли необхідно швидко отримати результат з обмеженими ресурсами або на невеликих наборах даних. Крім цього такі методи також мають свої переваги у випадках коли важлива інтерпретація результатів, оскільки вони часто забезпечують зрозумілі моделі, де кожен параметр можна легко інтерпретувати.

Нейронні мережі, зокрема глибокі моделі, такі як LSTM, BiLSTM або трансформери зазвичай показують набагато кращі результати при обробці великих за обсягом та неструктурованих даних. Подібні методи є набагато ефективнішими, коли потрібно враховувати складні взаємозв’язки в даних, такі як: контекстні залежності в текстах або складні патерни, що не можна виявити через прості ознаки. Проте, нейронні мережі вимагають значних обчислювальних ресурсів і часто потребують великих обсягів даних для досягнення оптимальних результатів.

Також важливо враховувати те як кожен метод справляється з класифікацією фейкових і справжніх новин, так як в деяких випадках певні моделі можуть краще справлятися з виявленням фейкових новин, але при цьому давати гірші результати класифікації справжніх новини, і навпаки. Тому важливо враховувати значення різних метрик при оцінці моделей.

В рамках дипломного проєкту, доцільним є власна розробка та порівняльний аналіз моделей, орієнтованих на задану предметну область і тип вхідних даних.

2 ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ

2.1 Опис обраних технологій розробки

В даному розділі визначимо основні технології розробки та технології, що якнайкраще відповідали б вимогам системи, що розробляється, визначним у розділі 3. Основними критеріями вибору технологій були: підтримка відповідних алгоритмів, зручність інтеграції, документація, відповідність визначеним вимогам а також ефективність при роботі з текстовими даними.

Мова програмування Python. Серед основних причин вибору мови програмування Python як основної мови застосунку є широка підтримка бібліотек машинного навчання та зручність синтаксису, що роблять Python однією з найкращих мов програмування для вирішення задач машинного навчання та аналізу даних. Python має одні з найкращих і найвідоміших бібліотек, що покривають всі етапи побудови системи обробки та аналізу даних до побудови моделей і створення API. Завдяки зрозумілій структурі Python дозволяє швидко створювати прототипи, експериментувати з алгоритмами та адаптувати код до нових вимог, що особливо важливо в науково-дослідницьких проєктах. Крім того, важливим критерієм була підтримка асинхронності, що було особливо важливо у створенні вебінтерфейсу застосунку.

Бібліотеки для машинного навчання. Бібліотеки для машинного навчання були підібрані з урахуванням визначених вимог до задач класифікації текстів, простоти використання, гнучкості налаштувань і наявності інструментів для повного циклу розробки моделей.

Scikit-learn — одна з найпопулярніших бібліотек для реалізації класичних алгоритмів машинного навчання. Дана бібліотека підтримує широкий спектр моделей серед яких методи, необхідні для виконання дослідження: логістична регресія (Logistic Regression), метод опорних векторів (SVM), випадкові ліси (Random Forest) та наївний байєсівський класифікатор (Naïve Bayes). Крім того, scikit-learn забезпечує інструменти для розділення датасету на навчальну та тестову вибірки (train_test_split), створення конвеєрів обробки даних (Pipeline), збереження

моделей, крос-валідації (cross_val_score), побудови confusion matrix та обчислення таких метрик як accuracy, percision, recall та F1-score.

TensorFlow та Keras — використовувалися для створення і навчання нейронних мереж, зокрема LSTM (Long Short-Term Memory), Bi-LSTM (Bidirectional LSTM) та GRU (Gated Recurrent Unit). Keras є високорівневим API, що вбудований у TensorFlow, і дозволяє легко та інтуїтивно будувати моделі.

Joblib — використовується для збереження і завантаження об'єктів, зокрема моделей машинного навчання, великих масивів NumPy та словників. У рамках проєкту joblib було використано для збереження навчених моделей після тренування з метою подальшого використання їх в API-сервісі без необхідності повторного навчання.

Бібліотеки для обробки природної мови. Бібліотеки для обробки природної мови були обрані з урахуванням необхідності якісної попередньої обробки текстових даних, очищення, нормалізації, лематизації, векторизації та підготовки вхідних даних для подачі в моделі машинного навчання.

NLTK (Natural Language Toolkit) — одна з найбільш відомих бібліотек для NLP в Python. Використовувалась для токенізації, видалення стоп-слів та інших базових операцій. Містить вбудовані корпуси текстів, словники та мовні ресурси, що допомагають швидко налаштувати базову обробку тексту.

Re (Regular Expressions) — стандартна бібліотека Python для роботи з регулярними виразами. Використовувалась для очищення тексту, видалення чисел, розділових знаків, спецсимволів та зайвих пробілів.

Зберігання, обробка та візуалізація даних. Для зберігання, обробки, візуалізації та аналізу текстових і числових даних були використані перераховані нижче бібліотеки.

Pandas — одна з найпопулярніших бібліотек для аналізу даних у Python. Використовувалась для читання та збереження даних у форматі .csv, очищення даних, обробки пропущених значень та групування даних.

Numpy — базова бібліотека для числових обчислень у Python. Забезпечувала швидкі операції з векторними та матричними даними, які використовувалися у

процесі векторизації текстів та при підготовці даних для нейронних мереж. NumPy також була основою для інших бібліотек (наприклад, pandas, scikit-learn, TensorFlow).

Matplotlib — базова бібліотека для побудови графіків і візуалізації даних у Python. У рамках проєкту використовувалась для візуалізації обсягів даних та матриць невідповідності.

FastAPI. Для створення вебінтерфейсу системи було використано фреймворк FastAPI. Однією з головних переваг FastAPI є підтримка асинхронного програмування, що дозволяє ефективно обробляти велику кількість запитів одночасно, забезпечуючи швидку та стабільну роботу застосунку навіть при підвищеному навантаженні. Крім цього, FastAPI має інтеграцію з бібліотекою Pydantic, що дозволяє автоматично перевіряти вхідні дані та генерувати документацію API у форматі OpenAPI (Swagger UI), що спрощує розробку та тестування застосунку.

Docker. Для швидкого та легкого запуску застосунку було обрано інструментарій для управління ізольованими Linux-контейнерами Docker. Він дозволив створити ізольований контейнер що містив всі необхідні файли застосунку та версії бібліотек і іншого програмного забезпечення, що дозволило уникнути проблем з сумісністю та конфліктів версій програмного забезпечення.

Makefile. Для скорочення та автоматизації виконання часто використовуваних команд, таких як встановлення необхідних залежностей чи збірка застосунку було використано makefile — текстовий файл, у якому описано набір команд із зручними псевдонімами, що дозволяють запускати складні команди простою інструкцією з термінала.

Висновки до розділу 2

У цьому розділі було розглянуто й обґрунтовано вибір інструментів та технологій, що стали основою для побудови системи виявлення фейкових новин. В процесі прийняття рішень враховувались точність, продуктивність,

масштабованість, зручність інтеграції, а також підтримка сучасних алгоритмів обробки тексту. Мова програмування Python була обрана як основна мова реалізації, не лише через велику кількість бібліотек, а й через гнучкість, інтуїтивність та здатність швидко адаптуватися під нові завдання. Крім цього, беззаперечною перевагою даної мови є наявність бібліотек, таких як Scikit-learn, TensorFlow та NLTK, що дозволили охопити увесь спектр необхідних підходів — від класичних алгоритмів та засобів обробки тексту до нейромережевих моделей. Для створення користувацького інтерфейсу було обрано FastAPI, що забезпечує зручну інтеграцію з моделями класифікації.

Не менш важливою частиною стало впровадження інструментів для розгортання — Docker і Makefile. Контейнеризація завдяки Docker значно спростила розгортання на різних середовищах, мінімізуючи ризики пов’язані з несумісністю бібліотек чи налаштувань операційної системи. Makefile же допоміг прибрати зайву рутину — замість довгих команд в терміналі, усе зводиться до простих інструкцій.

Також слід наголосити, що обрані технології розробки не лише відповідають функціональним вимогам, а й мають активну спільноту користувачів та багату документацію, що стало важливим чинником в процесі реалізації системи.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

У сучасному інформаційному просторі проблема розповсюдження фейкових новин стала особливо актуальною. Масове поширення неправдивої інформації через соціальні мережі та новинні сайти має серйозні соціальні, політичні та економічні наслідки. Тому виникає потреба в автоматизованих інструментах, здатних швидко та ефективно визначати достовірність новинного повідомлення.

Задача виявлення фейкових новин формулюється як задача бінарної класифікації текстів, де кожен текст повинен бути автоматично віднесений до одного з двох класів:

– «фейкова новина» (fake) — новина, що містить неправдиву або спотворену інформацію;

– «правдива новина» (real) — новина, що відповідає дійсності.

На вхід системи подається текст новини. Система повинна здійснити:

– попередню обробку тексту;

– представлення його у вигляді формального вектору ознак;

– класифікацію на основі навченої моделі.

У межах роботи ставиться мета:

– проаналізувати існуючі методи виявлення фейкових новин;

– реалізувати систему класифікації текстових повідомлень з використанням методів машинного навчання;

– навчити обрані моделі на датасеті, що містить розмічені дані;

– оцінити точність роботи системи на тестових даних.

Задача поєднує в собі етапи обробки природної мови (NLP), побудови векторного представлення текстів, вибору ефективного алгоритму класифікації та оцінювання його якості.

3.2 Математична постановка задачі

Задача виявлення фейкових новин формалізується як задача бінарної класифікації на основі текстових даних.

Нехай маємо навчальну вибірку:

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}, \quad (3.1)$$

де: $x_i \in R^d$ — вектор ознак, який відповідає і-й новині після перетворення тексту у числове представлення (наприклад, TF-IDF або векторне вбудовування слів/речень); $y_i \in \{0,1\}$ — цільова змінна (0, якщо новина правдива, та 1, якщо фейкова); N — кількість прикладів у вибірці; d — розмірність простору ознак.

Мета — знайти таку функцію класифікації:

$$f(x;\ \theta): R_d \ \rightarrow\ [0,1], \quad (3.2)$$

яка для довільного вхідного вектору ознак x прогнозує ймовірність того, що новина є фейковою. Параметри θ залежать від вибраного методу.

Формулювання задачі зводиться до мінімізації функції втрат (loss function), яка оцінює відхилення прогнозованого значення від фактичного. Для заданої задачі бінарної класифікації використовується бінарна крос-ентропійна функція втрат L(θ):

$$\mathrm{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} y_i * \log\big(p(y_i)\big) + (1 - y_i) * \log\big(1 - p(y_i)\big) \quad (3.3)$$

Задача навчання зводиться до пошуку такої функції f та параметрів θ, що мінімізують функцію втрат:

$$\Theta^* = argmin\, L(\theta) \quad (3.4)$$

3.3 Обґрунтування методу розв’язання

Для вирішення поставленої задачі було обрано наступні методи машинного навчання:

Naive Bayes. Це один із базових статистичних підходів до класифікації текстів, який ґрунтується на застосуванні теореми Байєса з припущенням незалежності ознак. Даний метод зазвичай показує високу швидкість навчання та передбачення, проте може показувати гірші результати на даних зі складними текстовими залежностями через припущення щодо незалежності ознак.

Логістична регресія. Це ще один класичний підхід до розв’язання задачі бінарної класифікації. Даний метод дозволяє моделювати ймовірність приналежності тексту до певного класу за допомогою лінійної комбінації ознак. Метод є простим у реалізації, добре масштабується на великі датасети та дозволяє легко інтерпретувати вплив окремих ознак. Проте не враховує порядок слів у тексті та через лінійність моделі є досить обмеженим у вираженні складних залежностей.

Метод опорних векторів. В основі даного методу лежить побудова максимальної роздільної гіперплощини між класами. Цей метод особливо ефективний при великій кількості ознак і малому обсязі даних, що робить його доречним у текстових задачах після векторизації. Завдяки використанню ядрових функцій (kernal functions), SVM здатен моделювати нелінійні залежності, однак навчання може бути тривалим при великій кількості даних, а інтерпретація результатів складною.

Метод випадкових лісів (Random Forest). Даний метод є ансамблевим методом, що використовує велику кількість дерев рішень, кожне з яких приймає часткове рішення, а фінальний прогноз формується більшістю голосів. Його ключова перевага полягає у стабільності результатів, меншій чутливості до шуму та оцінці важливості ознак. Проте складність структури моделі ускладнює її інтерпретацію, а навчання потребує більше обчислювальних ресурсів порівняно з одиничними моделями.

LSTM — це клас рекурентних нейронних мереж, розроблений для подолання проблеми зникнення або вибухового градієнта у рекурентних нейронних мережах. LSTM-мережі мають спеціальні механізми контролю потоку інформації через так звані «гейти» (входу, забування та виходу), що дозволяють ефективно зберігати або забувати інформацію про попередні стани.

GRU — це спрощений варіант LSTM, який зберігає основні властивості контролю інформаційного потоку, але має менше параметрів. GRU об'єднує механізми запам'ятовування та забування в один гейт і тим самим зменшує обчислювальну складність. Ця особливість робить GRU доцільним у випадках, коли обсяг даних обмежений або ресурси обчислень є критичними.

Bi-LSTM є розширенням класичної LSTM. Він складається з двох LSTM-шарів: один обробляє послідовність у прямому напрямку (від початку до кінця), а інший — у зворотному (від кінця до початку). Така архітектура дозволяє моделі враховувати як попередній, так і наступний контекст кожного елемента вхідної послідовності, що особливо корисно для задач обробки природної мови, де значення слова часто залежить від його оточення. Завдяки цьому Bi-LSTM забезпечує кращу точність у порівнянні зі звичайним LSTM у багатьох NLP-завданнях, зокрема в розпізнаванні мови, аналізі тональності та виявленні фейкових новин.

3.4 Опис методу розв'язання

Naive Bayes є класифікатором, заснованим на теоремі Байєса, що дозволяє оцінити ймовірність належності документа до певного класу, якщо відомі ймовірності появи слів у цьому класі. Формально, якщо є клас C і вектор x то клас документа обчислюється за формулою:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}, \tag{3.5}$$

де $P(c|x)$ — ймовірність того, що спостереження x належить до класу c, з урахуванням спостережених даних; $P(x|c)$ — ймовірність спостерігати такі ознаки x, якщо відомо, що приклад належить до класу c; $P(c)$ — початкова (апріорна) ймовірність класу c, тобто, як часто цей клас трапляється в загальній популяції, ще до врахування будь-яких ознак; $P(x)$ — загальна ймовірність спостерігати такі ознаки x незалежно від класу.

Припускаючи незалежність ознак, ймовірність P(x | c) можна розкласти на добуток умовних ймовірностей для кожного окремого слова:

$$P(x_1, x_2, \dots, x_n|c) = P(x_1|c) * P(x_2|c) * \dots * P(x_n|c) \quad (3.6)$$

Це значно спрощує обчислення та дозволяє моделі працювати з великими текстовими корпусами. Для навчання методу розраховуються умовні ймовірності кожного слова для кожного класу, а також апріорна ймовірність класу P(C), що визначає розподіл класів у навчальному наборі.

Логістична регресія — це метод, який дозволяє моделювати ймовірність того, що об’єкт належить до певного класу, використовуючи лінійну комбінацію вхідних ознак. Для кожного документа векторизований текст подається на вхід моделі у вигляді числового вектора ознак $x=(x_1, x_2, ..., x_n)$ для якого обчислюється зважена сума:

$$Z = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n, \quad (3.7)$$

де w_i — ваги, які навчаються під час тренування моделі, а w_0 — зсув (bias).

Щоб отримати ймовірність належності до позитивного класу, результат лінійної комбінації пропускається через сигмоїдну функцію активації**.**

Сигмоїдну функцію активації зображено на рисунку 3.1.

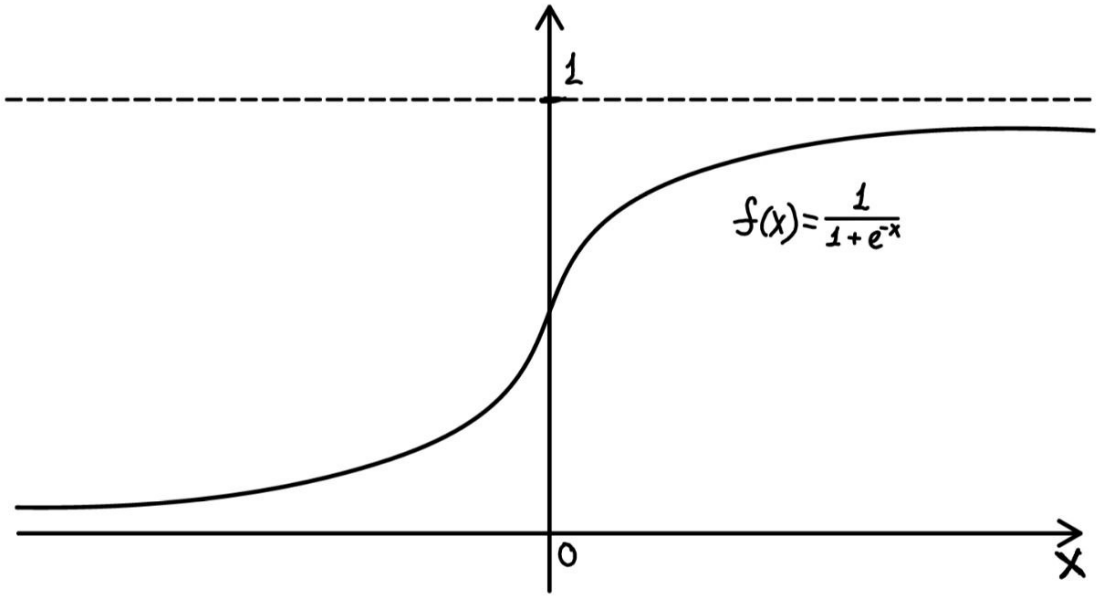

Рисунок 3.1 — Сигмоїдна функція активації

Якщо отримане значення перевищує певний поріг (зазвичай 0.5), то текст класифікується як фейковий або реальний відповідно.

SVM — Метод опорних векторів шукає оптимальну гіперплощину, яка розділяє простір ознак, щоб мінімізувати помилки класифікації та максимізувати зазор між прикладами різних класів. Формально, для двокласової задачі SVM намагається знайти вектор ваг w та зсув b такі що задовольняли б умову:

$$y_i(w^T x_i + b) \geq 1 \; \forall \; \text{i}, \tag{3.8}$$

де x_i — вхідний вектор ознак, $y_i \in \{-1, 1\}$ — клас мітки, а гіперплощина $w^T x_i + b = 0$ розділяє дані.

Оптимізаційна задача зводиться до мінімізації норми вектора $||w||^2$, що відповідає максимізації зазору.

У задачах класифікації текстів використовується лінійне ядро, оскільки кожен документ представлений як високо розмірний розріджений вектор і в такому просторі лінійна межа зазвичай є достатньо ефективною. SVM дозволяє точно визначити ті документи, що найбільше впливають на формування межі, так звані

опорні вектори. Саме ці приклади є ключовими для прийняття рішення і визначають класифікаційні властивості моделі.

Для нових текстів передбачення класу здійснюється шляхом підстановки їх векторного представлення у знайдену гіперплощину, і на основі знаку результату визначається клас. Завдяки своїй здатності працювати з високо вимірними та розрідженими даними, SVM є ефективним інструментом для задач, що вимагають точної класифікації текстової інформації.

Метод випадкових лісів (Random Forest) є ансамблевим методом машинного навчання, який поєднує велику кількість незалежних дерев рішень для побудови стабільної та точної класифікаційної моделі. Основна ідея методу полягає в тому, щоб зменшити ризик перенавчання, притаманний окремим деревам, за рахунок колективного голосування багатьох слабких моделей.

На етапі навчання створюється множина дерев, кожне з яких тренується на випадковій підмножині навчальних даних, відібраній із заміщенням (методом бутстрепа). Крім того, при побудові кожного вузла дерева випадковим чином відбирається підмножина ознак, з яких вибирається найкраща для поділу. Ці два джерела випадковості у виборі об'єктів і ознак забезпечують високу різноманітність дерев, що зменшує загальну дисперсію моделі.

LSTM — це тип нейронної мережі, побудований на базі рекурентної архітектури, в якому кожен нейрон має внутрішню структуру, що дозволяє контролювати, яку інформацію зберігати, а яку забувати.

Структурно кожен блок LSTM складається з:

– forget gate (забувальний вентиль): вирішує, яку частину попереднього стану пам’яті варто забути;

– input gate (вхідний вентиль): визначає, яку нову інформацію додати до стану пам’яті;

– output gate (вихідний вентиль): формує вихід на основі поточного стану пам’яті.

Математично ці ворота реалізуються через сигмоїдні та тангенсоїдні функції активації. Будову LSTM детально зображено на рисунку 3.2.

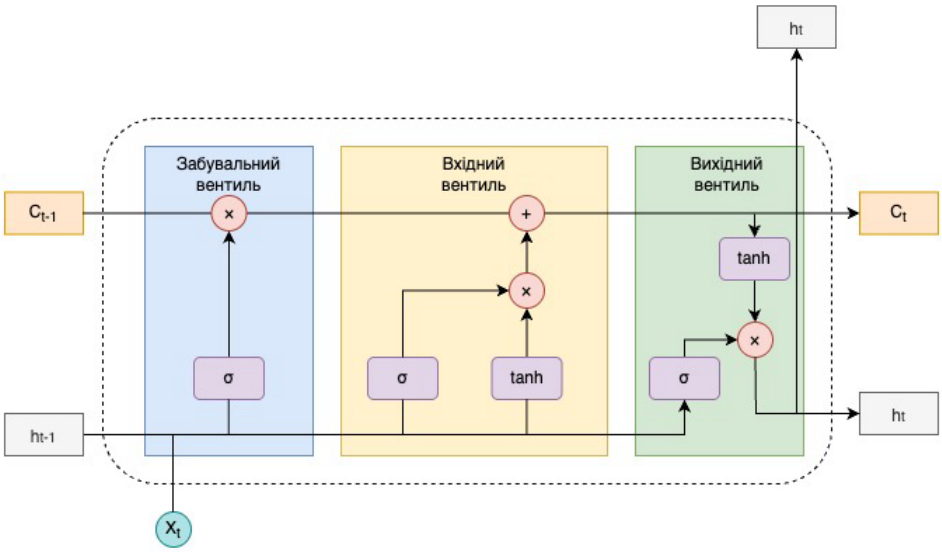

Рисунок 3.2 — Будова LSTM

У задачах обробки природної мови текст спершу перетворюється у послідовність векторів, яка подається на вхід LSTM. Мережа обробляє текст послідовно, зчитуючи по одному слову за раз, що дозволяє їй накопичувати контекст на кожному кроці. Після проходження всього тексту отриманий вихідний вектор може бути поданий у фінальний щільний шар (Dense) з функцією активації softmax або sigmoid для класифікації тексту як фейкового або справжнього.

Завдяки здатності пам’ятати контекст і враховувати порядок слів, LSTM моделі значно перевершують класичні статистичні алгоритми в задачах, де має значення структура та зміст тексту.

GRU — це тип рекурентної нейронної мережі, в якій кожна комірка має внутрішню структуру з двома елементами керування: update gate (вентиль уточнення) та reset gate (вентиль скидання). Ці “ворота” відповідають за контроль потоку інформації в часі: update gate вирішує, скільки інформації з попереднього стану слід передати далі, а reset gate скільки минулої інформації треба «забути».

GRU є спрощеною та менш ресурс затратною версією LSTM, що в більшості випадків не поступається їй у функціональності.

Будову GRU детально зображено на рисунку 3.3.

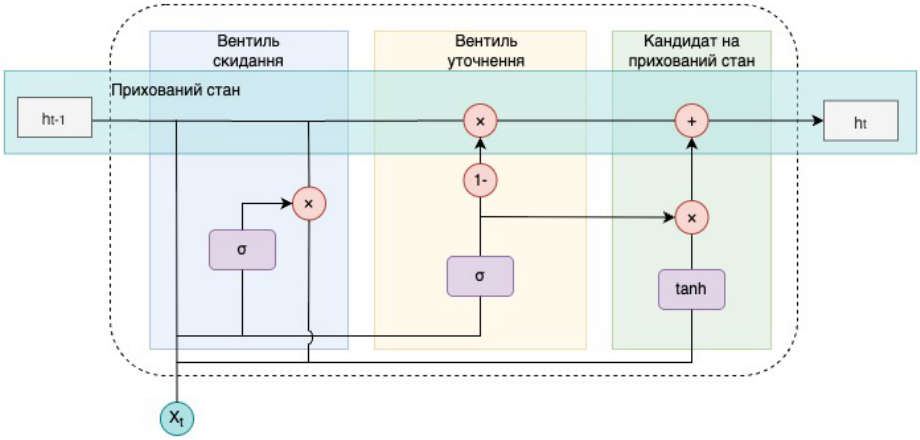

Рисунок 3.3 — Будова GRU

Bi-LSTM складається з двох LSTM-мереж: одна обробляє текст у прямому порядку (зліва направо), а інша у зворотному (справа наліво). Для кожного елемента послідовності обчислюється два вихідних вектори: h_{t1} з прямого проходу та h_{t2} із зворотного. Фінальне представлення на кожному кроці зазвичай отримується шляхом конкатенації:

$$h = [h_{t1}; h_{t2}] \tag{3.9}$$

Це дозволяє отримати вектор, який містить інформацію про контекст з обох сторін від слова. Перед подачею до Bi-LSTM модель текст проходить стандартну обробку: токенізацію, перетворення в послідовності індексів, та векторизацію. Послідовність векторів подається на вхід Bi-LSTM, яка повертає послідовність двосторонніх контекстуалізованих векторів. Один з останніх шарів нейронної мережі приймає ці вектори та передбачає клас (фейкова чи справжня новина), найчастіше у випадку бінарної класифікації через активацію sigmoid.

Будову Bi-LSTM детально зображено на рисунку 3.4.

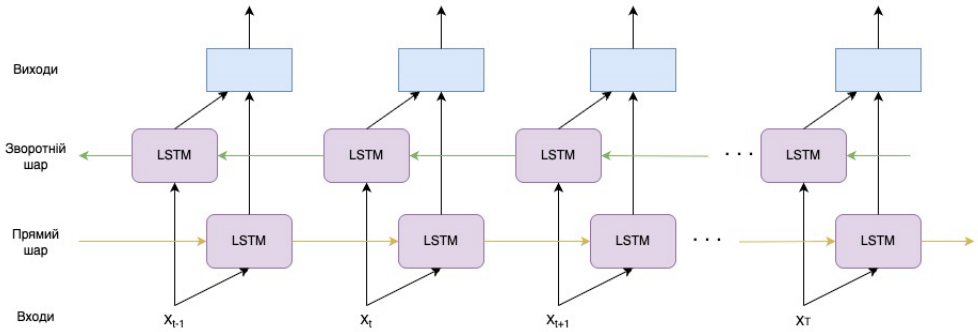

Рисунок 3.4 — Будова Bi-LSTM

Висновки до розділу 3

У цьому розділі було представлено математичне забезпечення системи виявлення фейкових новин, що включає формалізацію задачі, обґрунтування вибору методів машинного навчання та опис їхньої роботи. Задача класифікації новин як фейкових або справжніх була зведена до задачі бінарної класифікації тексту, де кожне повідомлення репрезентується у вигляді вектору ознак.

Для розв’язання цієї задачі було розглянуто низку алгоритмів: як класичних (Naive Bayes, логістична регресія, метод опорних векторів, випадкові ліси), так і моделей глибокого навчання (LSTM, GRU, Bi-LSTM). Класичні методи демонструють хорошу швидкодію, простоту реалізації та інтерпретованість, проте обмежені у виявленні складних контекстуальних залежностей. Натомість нейронні мережі, зокрема рекурентні архітектури, дозволяють враховувати порядок слів, граматичну структуру та семантику, що суттєво підвищує точність класифікації на реальних даних, попри вищі обчислювальні витрати.

4 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Структура системи

Розроблена система має модульну архітектуру, що забезпечує гнучкість, масштабованість і спрощене тестування окремих компонентів. Структура включає такі основні модулі:

Вебзастосунок. Призначений для взаємодії користувача із системою та реалізований у вигляді вебінтерфейсу з використанням html, JavaScript, Python та FastAPI. Функціональність: введення тексту новини, обробка даних та відображення результату класифікації. При побудові вебзастосунку використано принцип Server-Side Rendering (SSR), що забезпечує швидке завантаження сторінки, та спрощену інтеграцію з серверною логікою. HTML-розмітка формується на стороні сервера після обробки запиту та надсилається вже у готовому вигляді до браузера користувача. Це також дає змогу краще контролювати захист даних та обробку помилок.

Модуль попередньої обробки даних. Цей модуль виконує очищення тексту перед класифікацією. Серед основних кроків, виконуваних в цьому модулі: приведення до нижнього регістру, видалення спеціальних символів, HTML-розмітки, стоп-слів, токенізація та векторизація.

Модуль класифікації. Відповідає за аналіз вхідного тексту та визначення його як «фейковий» або «правдивий». В цьому модулі використані такі методи машинного навчання та нейронні мережі як: Logistic Regression, SVM, Random Forest, LSTM, Bi-LSTM та GRU.

Модуль збереження результатів. Відповідає за збереження, вже натренованих моделей і токенізаторів у форматах .pkl, .h5 та .joblib.

Вебчастина. Середня ланка між UI та класифікатором. Приймає запити від інтерфейсу, обробляє їх та надсилає результат.

4.1.1 Функціональна модель системи

Актори:

– користувач — основний актор, який взаємодіє з інтерфейсом системи, його роль полягає у введенні тексту новини для аналізу та отриманні результату;

– розробник — технічний актор, відповідальний за підтримку, оновлення і вдосконалення системи, зокрема машинних моделей;

– вебзастосунок — проміжна ланка між користувачем і внутрішніми модулями, передає введені дані, ініціює обробку та відображає результати;

– моделі МН (машинного навчання) — їх можна представити як окремого логічного актора, задача якого полягає у класифікації новин.

Діаграма варіантів використання представлена на кресленику ІС12.220БАК.005 Д1.

4.1.2 Модель вхідних даних

За основу дослідження було обрано набір даних про фейкові новини ISOT — це збірка кількох тисяч фейкових новин та правдивих статей, отриманих з різних легітимних новинних сайтів та сайтів, позначених Politifact.com як ненадійні.

Даний датасет містить 2 .csv файли з фейковими і правдивими новинами кожен з яких в свою чергу містить наступні колонки: “title”, “text”, “subject” та “date”. В процесі попередньої обробки даних для більш зручного навчання моделей було сформовано новий датасет, було опущено колонки “title”, “subject” та “date”, залишено колонку “text” та додано нову колонку “label” що містить значення 1 або 0 в залежності від правдивості тексту новини.

4.1.3 Передавання та обробка даних

Процес обробки даних складається з кількох етапів: попередньої обробки текстів новин, передавання вхідних даних від користувача до серверної частини, класифікації тексту та формування результату.

Вхідні дані: текст новини (рядок символів, введений користувачем у формі на вебсторінці); назва моделі (система підтримує декілька моделей для вибору).

Попередня обробка. Перед початком навчання моделі виконується очищення текстів новин у датасеті. Основні етапи обробки: видалення пунктуації, видалення чисел, перетворення тексту у нижній регістр, токенізація, видалення стоп-слів, об’єднання очищених токенів назад у текстову форму. Очищені тексти зберігаються у новому .csv файлі у вигляді масиву об’єктів з двома колонками: text — очищений текст новини; label — бінарна мітка (1 — правдива новина, 0 — фейкова).

Передавання даних від клієнта до сервера. Користувач взаємодіє з HTML-формою, розміщеною на головній сторінці (/). Після введення тексту новини та натискання кнопки аналізу форма надсилає POST-запит на маршрут /predict, серверна частина (FastAPI) приймає цей запит та передає текст до відповідної ML-моделі далі асинхронно у фоновому потоці виконується обробка даних і після класифікації формується результат у вигляді JSON-об’єкта, що містить: назву моделі (model), класифікаційний результат (prediction) та опціонально для нейронних мереж проміжний вихід моделі (model_output).

Вихідні дані: JSON-відповідь із результатом класифікації (правдива / фейкова новина), яка може бути відображена на клієнтському інтерфейсі.

4.1.4 Архітектура програмного забезпечення

Специфікація функцій. Для забезпечення повноцінної роботи системи виявлення фейкових новин реалізовано набір функцій, що охоплюють обробку текстових даних, класифікацію за допомогою машинного навчання, вивід

результатів та роботу вебсервісу. Нижче наведено короткий опис їх призначення та основних параметрів.

Classification_summary(y_test, y_pred, cmap="Purples", digits=5, title="Confusion Matrix"). Ця функція призначена для аналізу ефективності моделі. Вона виводить розширений звіт класифікації (точність, повнота, F-міра) та будує матрицю неточностей для бінарної класифікації. Параметри: y_test: справжні мітки класів; y_pred: передбачені мітки класів; cmap: палітра кольорів для візуалізації матриці; digits: кількість знаків після коми в звіті; title: заголовок діаграми.

Apply_model(model, text). Функція застосовує класичну модель машинного навчання (Logistic Regression, Random Forest, Naïve Bayes або Logistic Regression) до вхідного тексту, попередньо очищеного, векторизація відбувається всередині збереженої моделі. Параметри: model: об'єкт моделі; text: текст новини (str). Повертає: передбачену мітку (0 або 1).

Apply_nn(model, text). Застосовує нейронну мережу до вхідного тексту. Передбачає завантаження токенізатора, токенізацію, паддінг та передбачення. Параметри: model: завантажена нейромережа (LSTM, Bi-LSTM або GRU); text: вхідний текст. Повертає: кортеж з ймовірністю та бінарною міткою (0 або 1).

Clean_text(text). Функція для попередньої обробки тексту — видаляє знаки пунктуації, числа, стоп-слова та переводить текст до нижнього регістру. Параметри: text — необроблений текст новини. Повертає: очищений текст (str).

FakeNewsModelManager. Клас-менеджер для зручної роботи з різними моделями. Завантажує моделі з файлів, визначає їх тип (нейронна мережа чи класичний алгоритм) та виконує передбачення. Методи: __init__(model_paths: Dict[str, str], vectorizer_path: str): конструктор, завантажує моделі та векторизатор; predict(text: str, model_name: str): виконує передбачення, автоматично визначаючи, який метод застосувати.

Result_serializer(model, result). Форматує результат класифікації у вигляді словника, зручного для виводу на інтерфейсі або передачі через API. Параметри: model: назва моделі. result: результат класифікації (число або кортеж). Повертає: серіалізований словник з результатами.

Root(request: Request). Обробник запиту до головної сторінки вебзастосунку. Відповідає за повернення HTML-шаблону інтерфейсу.

Predict_news(req: NewsItem). Асинхронний обробник запиту на передбачення. Приймає текст новини та назву моделі, виконує передбачення у фоновому потоці та повертає результат користувачу у форматі JSON.

Діаграма послідовностей представлена на кресленику ІС12.220БАК.005 Д2. А діаграма компонентів — на кресленику ІС12.220БАК.005 Д3.

4.2 Попередня обробка і підготовка вхідних даних

Попередня обробка вхідних даних є одним з найважливіших етапів створення системи, що передбачає використання нейронних мереж. Неправильно оброблені дані можуть спричинити неточність результатів та вплинути на роботу моделі. Зайвий шум у даних може призводити до перенавчання та запам’ятовування моделлю нерелевантних закономірностей у даних. Відсутні ж чи некоректні дані (наприклад, дати до 1970 в датасеті що містить інформацію про онлайн новини чи суцільні числові дані в колонці з текстом новини) можуть сприяти тому що модель навчиться неправильно чи перенавчиться. Саме тому важливим кроком на цьому етапі є послідовна та логічна обробка вхідних даних.

Обробка текстових даних включає наступні етапи:

1) завантаження датасетів з реальними та фейковими новинами;

2) додавання міток класу (1 — реальні новини, 0 — фейкові);

3) вибір релевантних колонок (колонки з текстом новини) і видалення всіх інших: дати та заголовку, що є менш релевантними і можуть спричиняти шум і погіршувати роботу моделей;

4) об’єднання датасетів та випадкове перемішування даних, що дозволяє уникнути упередження моделі, так як початковий датасет представляв собою два датасети з фейковими та справжніми новинами, що були об’єднані на другому кроці то до перемішування записи кожного класу були розташовані рівномірно один за одним (спершу фейкові новини а потім справжні), що могло призвести до

того що деякі моделі б почали вловлювати закономірності у порядку подання даних, а не в самих даних, що б призвело до спотворення результатів роботи моделей;

5) застосування функції clean_text для очищення тексту до всіх записів, дана функція була розроблена в ході виконання завдання і включає такі етапи як: видалення пунктуації та цифр з тексту, перетворення тексту до нижнього регістру, токенізація тексту (поділ тексту на окремі слова), видалення стоп-слів, об’єднання вже очищених слів в один рядок, видалення порожніх рядків, видалення дублікатів, збереження вже обробленого датасету в окремий файл.

Структуру очищених даних та розподіл новин в очищеному датасеті наведено на рисунках 4.1 та 4.2 відповідно.

	text	label
0	st century wire says ben stein reputable profe...	0
1	washington reuters us president donald trump r...	1
2	reuters puerto rico governor ricardo rossello ...	1
3	monday donald trump embarrassed country accide...	0
4	glasgow scotland reuters us presidential candi...	1

Рисунок 4.1 — Структура очищених даних

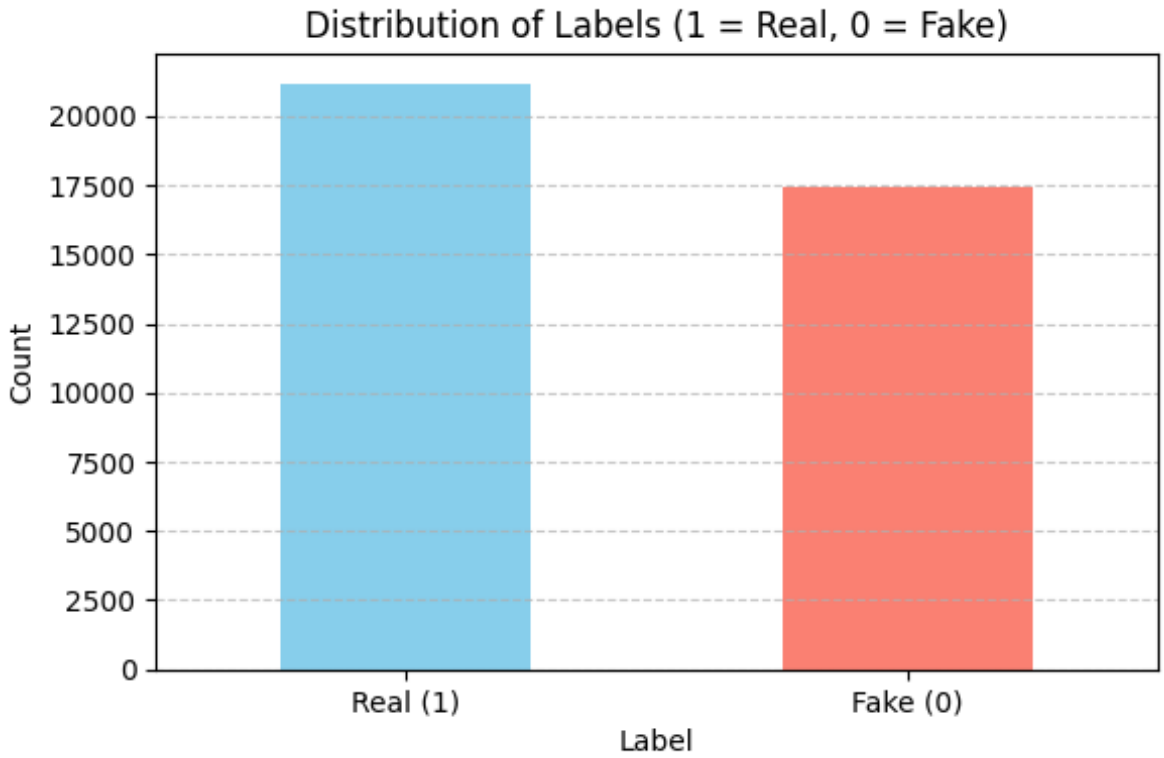

Рисунок 4.2 — Розподіл справжніх і фейкових новин в очищеному датасеті

4.2.1 Видалення стоп-слів

Одним із важливих кроків під час попередньої обробки текстових даних є видалення стоп-слів. Це слова, які дуже часто зустрічаються в мові, але самі по собі зазвичай не несуть ніякої корисної для аналізу інформації. У випадку з англійською це такі слова як, наприклад, “the”, “is”, “in”, “and”, “that”, “on”, “with” тощо. Вони допомагають будувати граматичну структуру речень, але майже не впливають на зміст.

У цьому дипломному проєкті для очищення текстів використовувався стандартний набір англійських стоп-слів з бібліотеки NLTK. Він містить кілька сотень найтиповіших слів, які автоматично фільтруються після токенізації. Видалення таких слів дозволяє зменшити кількість “зайвих” ознак у даних, зробити модель менш чутливою до шуму та пришвидшити подальші обчислення.

Функцію для очищення тексту наведено на рисунку 4.3.

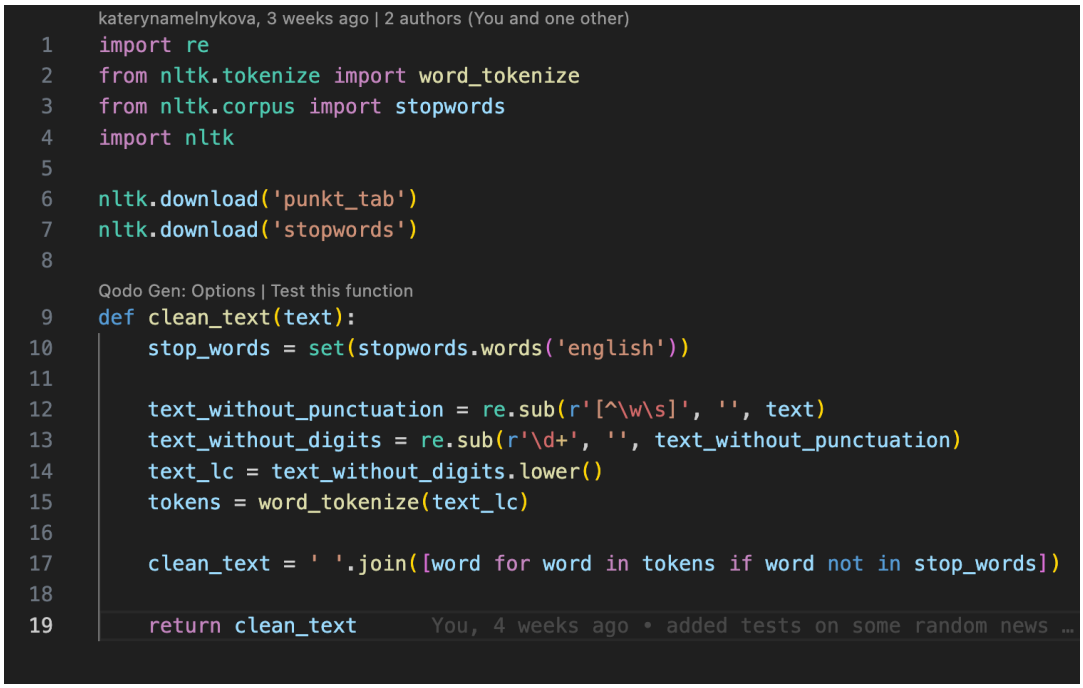

Рисунок 4.3 — Очищення тексту від стоп-слів у проєкті

Розглянемо цей процес на прикладі речення “The cat is sitting in the garden.” Стоп-словами в даному реченні є: “The”, “is” та “in”. Після очищення отримаємо вектор слів ["cat", "sitting", "garden"], що міститиме набагато менше шуму та не викликатиме таких спотворень у роботі моделей [6].

Цей крок особливо корисний у завданнях типу виявлення фейкових новин, де важливо зрозуміти саме зміст повідомлення, а не граматичні зв’язки. Наприклад, у заголовку слова на кшталт “fraud”, “government” або “conspiracy” дають набагато більше інформації для моделі, ніж формальні “the” чи “is”. Тому очищення тексту сприяє кращій якості класифікації.

4.2.2 Векоризація і токенізація

Для того щоб навчальна модель могла працювати з текстовими даними, їх потрібно попередньо перевести у числовий формат. Адже машинне навчання не розуміє тексту як такого, йому потрібні числа, вектори або матриці. Саме цим і займається векторизація. У цьому проєкті використовувався метод TF-IDF векторизації. Його основна ідея полягає в тому щоб надати словам вагу, яка

залежить від того, наскільки часто вони зустрічаються в одному документі, але водночас наскільки рідко вони трапляються в інших.

Слова, які зустрічаються в усіх текстах, наприклад, “news” або “said” отримують меншу вагу. Натомість терміни, що специфічні для окремих новин (скажімо, “scandal” чи “vaccine”) навпаки, оцінюються як більш інформативні. TF-IDF допомагає зосередитися на тому, що вирізняє один документ з-поміж інших.

TF (Term Frequency) вимірює частоту терміна в документі. Вона розраховується як відношення кількості разів, коли термін зустрічається в документі, до загальної кількості термінів у цьому документі. Мета полягає в тому, щоб підкреслити слова, які часто зустрічаються в документі.

$$TF(t,d) = \frac{\text{Кількість появ терміну } t \text{ в документі } d}{\text{Загальна кількість термінів у документі } d}, \quad (4.1)$$

IDF (Inverse Document Frequency) вимірює рідкість терміна в колекції документів. Вона обчислюється як логарифм від відношення загальної кількості документів до кількості документів, у яких зустрічається цей термін. Основна мета — зменшити вагу слів, які часто трапляються у всіх документах.

$$IDF(t,D) = \log\left(\frac{\text{Кількість документів у корпусі } D}{\text{Кількість документів, що містять } t}\right), \quad (4.2)$$

Значення TF-IDF обчислюється як добуток значень TF та IDF [7].

$$TF - IDF(t,d,D) = TF(t,d) * IDF(t,D), \quad (4.3)$$

У коді цей процес реалізовано через об'єкт TfidfVectorizer [8] з бібліотеки scikit-learn. У ньому одразу вказано stop_words='english', тобто стоп-слова будуть відфільтровані автоматично ще до розрахунку ваг. Власне, ця векторизація стала першою частиною пайплайну (Pipeline), у якому обробка тексту та навчання моделі

йдуть один за одним. Після того як кожен текст було перетворено на вектор чисел (де кожне число це вага відповідного терміну), ці вектори подаються на вхід логістичній регресії для навчання.

На виході маємо набір ознак, де кожен текст це вектор, довжина якого дорівнює кількості унікальних слів у всьому корпусі (після відсіву стоп-слів). Це дозволяє моделі не просто бачити, які слова є в документі, а й розуміти їхню важливість у контексті.

Такий підхід було застосовано для всіх класичних моделей машинного навчання.

Приклад створення Pipeline з використанням TfidfVectorizer та логістичної регресії наведено на рисунку 4.4.

```
lr = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english')),
    ('clf', LogisticRegression())
])

lr.fit(X_train, y_train)
```

Рисунок 4.4 — Приклад створення Pipeline з використанням TfidfVectorizer та логістичної регресії

Проте, для нейронних мереж було застосовано інший підхід. Спочатку було створено об’єкт Tokenizer [9] з бібліотеки keras tensorflow, з обмеженням словника num_words=max_words, щоб не “годувати” мережу рідкісними чи випадковими словами. Згодом токенізатор “пройшовся” усім корпусом (fit_on_texts) і склав власний словник: кожне унікальне слово отримало свій індекс. Так, звичайний текст на кшталт «breaking news scandal erupts» перетворюється на щось схоже до [18, 7, 351, 92]. Ці цифри вже зручні для мережі, бо вона оперує саме числами, а не рядками.

Далі метод texts_to_sequences переводить кожен документ у такий числовий список. Проте довжина речень різниться: одні короткі, інші як абзац. Щоб мережа

отримувала фіксований розмір вхідних даних, було застосовано pad_sequences. Короткі послідовності наповнюються нулями зліва (за замовчуванням), надто довгі акуратно обрізаються до max_len. У результаті масив X це рівномірна матриця samples × max_len, повністю готова для сприйняття моделлю. Тобто було збережено порядок слів, вирішено проблему із різною довжиною текстів і водночас не втрачено змісту, який потрібен мережі для навчання.

Приклад токенізації вхідних даних наведено на рисунку 4.5.

```
max_words = 20000
max_len = 200

tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(news['text'])

sequences = tokenizer.texts_to_sequences(news['text'])
X = pad_sequences(sequences, maxlen=max_len)
```

Рисунок 4.5 — Токенізація вхідних даних перед використанням нейронних мереж

4.3 Побудова і навчання моделей

4.3.1 Побудова і навчання класичних моделей

Після завершення попередньої обробки тексту та його векторизації була реалізована побудова і навчання моделей. У цьому проєкті для задачі класифікації текстових новин було обрано кілька базових, але перевірених підходів, реалізованих для більшої зручності і узгодженості через інструмент Pipeline [10] з бібліотеки scikit-learn. Pipeline дозволяє зібрати кілька кроків обробки та навчання в єдиний "конвеєр", який можна запускати однією командою. Наприклад, типова послідовність може включати векторизацію тексту (через TfidfVectorizer) і подальше навчання моделі класифікації.

Зокрема, було використано логістичну регресію, наївний байєс, метод опорних векторів та випадковий ліс. Ці моделі докладніше описано у розділі 3. Усі моделі тренувалися на одних і тих самих векторизованих даних, а після навчання зберігалися у вигляді .pkl файлів. Це дало змогу зручно повертатися до них на етапах тестування, без потреби кожного разу запускати навчання з нуля. Такий підхід економить час і дозволяє ефективно організувати роботу застосунку.

Приклад роботи Pipeline і joblib наведено на рисунку 4.6.

```
lr = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english')),
    ('clf', LogisticRegression())
])

lr.fit(X_train, y_train)

joblib.dump(lr, f'../{models_isot_path}/logistic_regression.pkl')


pipeline = joblib.load(f'../{models_isot_path}/logistic_regression.pkl')
y_pred = pipeline.predict(X_test)
```

Рисунок 4.6 — Приклад роботи Pipeline і joblib, реалізований в проєкті

4.3.2 Побудова і навчання нейронних мереж

У рамках проєкту було також реалізовано декілька моделей на основі нейронних мереж. Серед них: GRU, класична LSTM та двонапрямлена BiLSTM. Детальніше будову цих моделей описано у розділі 3. Всі моделі будувалися за допомогою Keras API, що входить до складу TensorFlow, і мали подібну структуру з деякими варіаціями.

Кожна архітектура починалася з шару Embedding, який відповідає за перетворення індексів слів на вектори фіксованої розмірності. Це дозволяє моделі краще працювати з лінгвістичними зв’язками між словами. Після цього додавався один із рекурентних шарів в залежності від типу нейронної мережі (GRU, LSTM чи Bi-LSTM). У випадку з Bi-LSTM цей шар було огорнуто в шар Bidirectional. Це

спеціальний тип шару, який дозволяє нейронній мережі враховувати контекст як зліва направо, так і справа наліво, що власне і потрібно для Bi-LSTM. Bidirectional створює дві копії шару LSTM, одна читає послідовність у прямому напрямку, а інша у зворотному. Обидві частини навчаються паралельно, а потім їхні вихідні значення об'єднуються. У результаті чого модель має доступ до повнішого контексту на кожному кроці обробки, що часто призводить до покращення точності класифікації.

На виході кожної моделі використовувався щільний (Dense) шар з одним нейроном і сигмоїдною функцією активації, що забезпечує прогноз імовірності належності тексту до одного з двох класів. Як функцію втрат було обрано binary_crossentropy, щодетальніше описана у третьому розділі. Для оптимізації використовувався відомий алгоритм Adam (Adaptive Moment Estimation) [11]. Даний алгоритм поєднує в собі переваги двох інших відомих методів: AdaGrad та RMSProp і реалізує адаптивне регулювання швидкості навчання для кожного параметра моделі окремо. Це означає, що на відміну від традиційного стохастичного градієнтного спуску з фіксованим кроком, Adam автоматично змінює швидкість оновлення ваг залежно від історії градієнтів.

Алгоритм підтримує експоненційно згладжені оцінки першого (моменту градієнта) та другого порядку (квадратів градієнтів). Це дозволяє йому гнучко коригувати розмір кроку оновлення для параметрів з різною частотою та амплітудою змін. Завдяки цьому Adam є більш стійким до шуму та швидше досягає збіжності.

Навчання моделей відбувалося протягом п’яти епох із розміром пакету 64, із відокремленням 10% тренувальних даних для валідації. Значення гіперпараметрів підбиралися індивідуально для кожної моделі і були обрані експериментально. Після завершення навчання кожна модель зберігалася у форматі .h5 для подальшого використання без необхідності повторного тренування.

4.4 Вебзастосунок

Вебзастосунок побудовано на основі FastAPI, що дозволило об'єднати функціональність REST API з базовою серверною візуалізацією HTML через Jinja2. Користувач взаємодіє з HTML-інтерфейсом, де можна ввести текст новини та вибрати одну з доступних моделей. Після надсилання даних запит обробляється JavaScript клієнтом і передається до серверної частини у вигляді JSON через POST-запит.

Інтерфейс вебзастосунку наведено на рисунку 4.7.

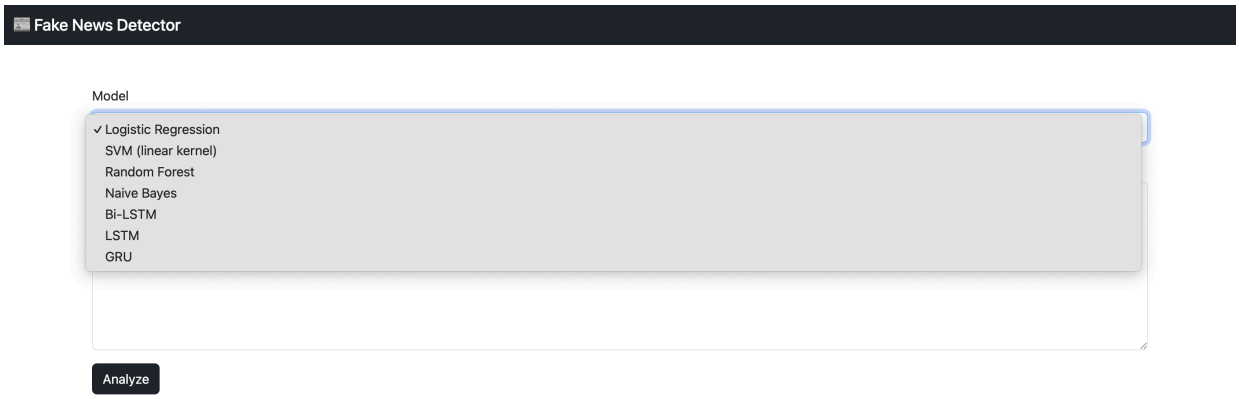

Рисунок 4.7 — Інтерфейс вебзастосунку

Серверна частина завантажує попередньо збережені моделі машинного навчання, включно як з класичними моделями (Logistic Regression, SVM, Random Forest, Naive Bayes), так і з нейронними (LSTM, GRU, Bi-LSTM). Для цього використовується спеціальний клас FakeNewsModelManager, який інкапсулює логіку завантаження моделей та обчислення прогнозу.

Після надходження запиту модель обирається відповідно до переданого параметра, текст векторизується, і результат прогнозу повертається у відповідь. Відповідь повертається у вигляді JSON-структури, що містить назву моделі, числовий результат (тільки для нейронних моделей) і саму класифікацію (FAKE або REAL).

На клієнтському рівні результат обробляється за допомогою JavaScript і динамічно відображається без перезавантаження сторінки. Перша HTML-сторінка рендериться сервером, подальша взаємодія реалізована через асинхронні API-запити. Це створює змішану архітектуру: з одного боку, надсилається серверний HTML (SSR), з іншого, подальші дії виконуються на клієнті (через JavaScript), що характерно для SPA-підходів.

Діаграма діяльності, що ілюструє описані вище процеси, представлена на кресленику ІС12.220БАК.005 Д4.

Висновки до розділу 4

У цьому розділі було детально описано структуру системи виявлення фейкових новин, її функціональну модель, архітектуру, а також способи обробки та передавання даних. Крім цього було побудовано модель варіантів використання, яка демонструє взаємодію основних акторів — користувача, розробника, вебзастосунку та моделей машинного навчання з основними компонентами системи.

Також подано структуру вхідного датасету, що базується на попередньо обробленому датасеті ISOT, у якому новини представлені у вигляді тексту з відповідною міткою істинності. Наведено специфікацію функцій, які забезпечують класифікацію новин, попередню обробку тексту та взаємодію із вебінтерфейсом.

Архітектура програмного забезпечення описана через діаграми класів, послідовності та компонентів, які ілюструють логіку роботи системи, взаємозв’язки між її модулями та їхню інтеграцію.

Крім цього, було детально описано процес підготовки вхідних даних перед обробкою їх моделями машинного навчання, детально описано процес векторизації та токенізації вхідних даних, а також структуру Pipeline при навчанні класичних моделей. Також було детально описано принцип побудови та навчання всіх класичних моделей машинного навчання а також архітектуру нейронних мереж.

5 ТЕСТУВАННЯ ТА ОГЛЯД ГОТОВОЇ СИСТЕМИ

5.1 Тестування та порівняльний аналіз моделей машинного навчання

Оцінка якості моделей є невід'ємною частиною будь-якого проєкту, пов'язаного з машинним навчанням. Побудувати модель недостатньо, потрібно також зрозуміти, як вона працює в реальних умовах, наскільки стабільні її результати, чи не відбулося перенавчання (необхідно перевірити чи справляється модель з класифікацією даних, які вона раніше не бачила). У цьому контексті тестування відіграє ключову роль. Саме тут стає зрозуміло, чи має модель практичну цінність, чи просто добре запам'ятала навчальні приклади. У наступних підрозділах буде детальніше розглянуто, як саме проводився аналіз результатів і чим відрізняються між собою всі реалізовані моделі.

У даному дослідженні було використано метод hold-out [12], даний метод передбачає однократне поділення датасету на тренувальну та тестову частини. Навчання моделі відбувається лише на тренувальній вибірці, а оцінка її здатності узагальнювати виконується на відкладених даних. Зазвичай вибірка ділиться в співвідношенні 80 до 20, хоча це не є жорстким правилом. Такий підхід особливо зручний при роботі зі значними обсягами даних або коли потрібен швидкий результат без надмірних обчислень. У порівнянні з перехресною перевіркою (cross-validation) він є менш ресурсномістким, але водночас і менш стійким до варіативності вибірки. Зважаючи на розмір датасету та складність моделей даний метод був найбільш доречним в даній ситуації.

Метод hold-out наглядно проілюстровано на рисунку 5.1.

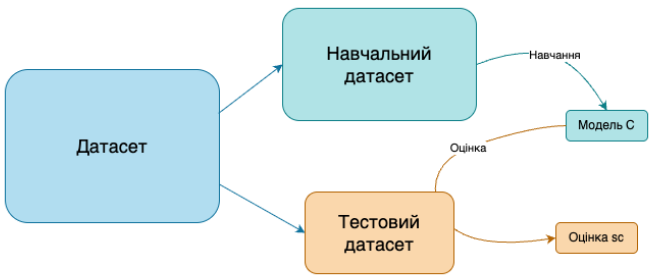

Рисунок 5.1 — Hold-out

5.1.1 Огляд метрик оцінки моделей машинного навчання

Матриця невідповідностей (Confusion matrix) [13]. Оцінка класифікаційної моделі часто не обмежується лише загальною точністю або середніми метриками. Щоб детальніше побачити, де саме модель помиляється, використовують матрицю невідповідностей. Це проста таблиця, яка показує, скільки разів модель передбачила той чи інший клас і скільки з цих передбачень були правильними. Як правило, по одному з вимірів йдуть фактичні значення (ground truth), а по іншому передбачені. У літературі можна зустріти обидва варіанти: або фактичні по рядках, або по стовпцях, суть від цього не змінюється.

У випадку бінарної класифікації така матриця має чотири комірки. Вони відповідають кількості випадків, коли модель класифікувала приклад правильно або помилково. Наприклад, true positive - це коли зразок позитивного класу було правильно класифіковано. False positive — коли модель помилково віднесла негативний приклад до позитивного класу. Аналогічно визначаються false negative та true negative. Сукупно ці чотири значення описують усі рішення моделі на тестовій вибірці.

Для багатокласової класифікації структура розширюється, але принцип лишається тим самим. Кожен рядок відповідає реальному класу, кожен стовпець передбаченому. Ідеальною ситуацією є домінування значень на діагоналі, що означає, що більшість передбачень збігаються з істинними класами. Позакласові помилки добре видно поза діагоналлю — це випадки, коли модель переплутала одні класи з іншими. Такий формат представлення дозволяє швидко побачити, де саме є систематичні збої.

Матрицю невідповідностей наведено на рисунку 5.2.

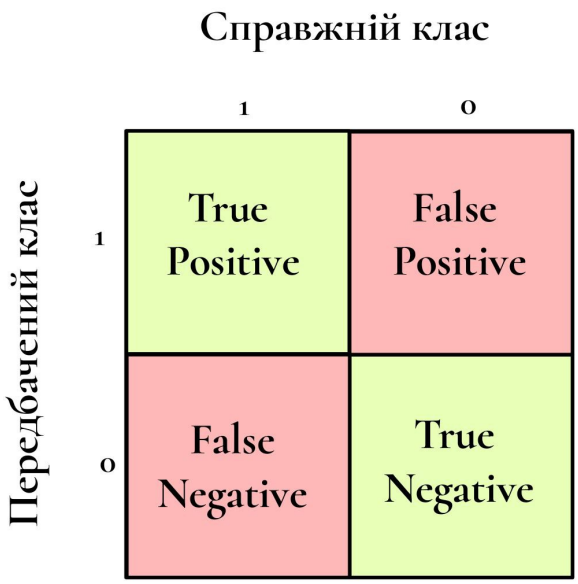

Рисунок 5.2 — Матриця невідповідностей

Точність (Accuracy) [14]. Точність — це одне з найважливіших метрик, на яку звертають увагу при оцінці моделі. Вона дає загальне уявлення про те, наскільки багато передбачень були правильними. При обрахунку точності кількість правильно передбачених класів (True Positive та True Negative) ділиться на загальну кількість прикладів. Такий підхід є досить правильним і доречним, але лише у випадку коли дані гармонійно збалансовані. Тобто, у випадку задачі бінарної класифікації, коли розподіл об’єктів обох класів становить приблизно 50/50.

Коли ж один клас починає домінувати, точність перестає бути надійним орієнтиром. Якщо модель вгадує здебільшого лише більший за кількістю клас, то загальна точність може залишатись високою, навіть якщо вона постійно помиляється на меншому класі. Наприклад, якщо в наборі 90 % новин справжні, а 10 % фейкові, то модель може просто завжди передбачати «справжня» і отримає при цьому точність 90 %. Але якщо насправді потрібно саме виявляти фейкові тексти, то така модель нічого корисного не робить. Вона ігнорує меншість, яка якраз і є найбільшою проблемою. Тому точність варто використовувати обережно й обов'язково дивитися на інші метрики, які враховують характер помилок.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5.1)$$

Влучність (precision). Влучність — це ще одна метрика, що допомагає краще зрозуміти, як поводиться модель, особливо коли йдеться про роботу з важливими чи рідкісними класами. Вона показує те скільки з усіх передбачень позитивного класу були насправді правильними. Інакше кажучи, наскільки модель «обережна» у своїх позитивних рішеннях.

Наприклад, якщо модель визначила сто новин як справжні (1 — позитивний клас), а насправді 20 з них були фейками (0 — негативний клас), то кількість TP значень дорівнювала 80, а FP — 20, precision буде дорівнювати 0.8. Це означає, що двадцять відсотків випадків — це помилки, де фейкові тексти було помилково визнано справжніми. У таких задачах, як виявлення фейків, precision особливо важливий, так як високі значення цієї метрики свідчать про здатність моделі утримувати хибно позитивні передбачення на мінімальному рівні.

$$Precision = \frac{TP}{TP + FP} \quad (5.2)$$

Повнота (recall). Дана метрика відображає здатність моделі виявляти всі об'єкти позитивного класу. У контексті виявлення фейкових новин це показує те наскільки добре модель знаходить усі справжні новини серед усього масиву текстів, які насправді є правдивими. Наприклад, якщо у тестовій вибірці було 100 справжніх новин, а модель правильно розпізнала лише 80 з них, то recall становитиме 0.8 (TP = 80, а FN = 20). У цьому випадку 20 справжніх новин було хибно віднесено до фейкових, що може призводити до втрати важливої інформації. Тому, якщо перед нами стоїть мета зменшити кількість втрачених позитивних зразків, recall стає значимою метрикою.

$$Recall = \frac{TP}{TP + FN} \quad (5.3)$$

F1-score. F1-score — це ще одна метрика, що узагальнює дві важливі характеристики класифікатора: precision і recall. У багатьох випадках ці метрики перебувають у певному протиріччі. Коли модель намагається охопити якомога більше справжніх позитивних (TP) випадків, тобто підвищити recall, вона нерідко починає захоплювати й хибнопозитивні (FP) приклади, що знижує precision. І навпаки. F1-score намагається знайти баланс між цими двома метриками, обчислюючи гармонічне середнє між precision та recall, і в такий спосіб дає більш комплексну оцінку якості класифікації.

Ця метрика особливо корисна на незбалансованих наборах даних, де одна з категорій трапляється значно рідше. У таких умовах простий показник точності (accuracy) може бути оманливим. Наприклад, якщо модель передбачає рідкісне явище на кшталт хвороби або фейкової новини, вона може досягати високого precision, просто не передбачаючи позитивний клас узагалі, але recall у такому випадку буде нульовим. Інша крайність — якщо модель усе вважає позитивним класом, recall буде високим, але precision стане дуже низьким. У цьому випадку F1 покаже, що модель працює погано, навіть якщо окремі метрики виглядають добре. Саме тому F1 часто використовують у дослідженнях, де важлива збалансована оцінка.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.4}$$

Під час оцінювання якості класифікації, особливо в багатокласових або незбалансованих задачах, окремі метрики для кожного класу самі по собі не дають повної картини. Тому вводяться усереднені значення, такі як macro avg і weighted avg, що дозволяють отримати більш узагальнену оцінку якості моделі. Вони відрізняються способом обчислення й тим, наскільки враховують розподіл класів.

Macro average [15]. Macro average — це просте середнє арифметичне значення метрики (наприклад, precision або recall) для кожного класу. Кожен клас у цьому випадку має однакову вагу, незалежно від того, наскільки часто він трапляється у вибірці. Це робить macro average корисним тоді, коли важливо оцінити, як модель

поводиться щодо кожного класу окремо, не зважаючи на дисбаланс. Але водночас macro average не показує реальний вплив рідкісних класів на загальну продуктивність, тому він може виглядати добре навіть тоді, коли модель дуже погано працює з найчастішим класом.

$$M_{macro} = \frac{1}{N}\sum_{i=1}^{N} M_i\,, \tag{5.5}$$

де N — кількість класів, M — метрика середнє значення якої обраховується, M_i — значення метрики для кожного класу.

Weighted average [15]. Дане середнє значення часто використовують як узагальнюючу оцінку, коли важливо враховувати нерівномірний розподіл класів у даних. На відміну від macro average, яке просто обчислює середнє значення метрик без огляду на те, скільки прикладів було у кожного класу, weighted підходить до цього більш зважено. Дане значення підтягує кожну метрику відповідно до того, скільки було зразків (значення support), тобто чим більше прикладів класу, тим більший вплив його точності, повноти чи F1-score на підсумкову оцінку. У звітах класифікації ця метрика часто показує результат, який найближчий до загального сприйняття «точності» моделі, особливо коли один клас домінує.

З іншого боку, такий підхід містить і свої недоліки, так як невеликі класи можуть практично не впливати на підсумкову оцінку. Якщо модель працює на них погано, то це буде непомітно. Наприклад, якщо більшість новин у датасеті справжні, то модель, яка просто вгадує «справжня» майже завжди, отримає високий weighted F1, навіть якщо з фейковими новинами вона не справляється зовсім. Тому інтерпретувати цю метрику слід з урахуванням контексту.

$$M_{weighted} = \sum_{i=1}^{N} w_i * M_i\,, \tag{5.6}$$

де N — кількість класів, M — метрика середнє зважене значення якої обраховується, M_i — значення метрики для кожного класу, w_i — значення ваг для кожного класу.

$$w_i = \frac{support_i}{\sum_{j=1}^{N} support_j}, \tag{5.7}$$

де N — кількість класів, $support_i$ — кількість зразків у класі *i*.

5.1.2 Порівняння моделей машинного навчання

Naive Bayes. Модель Naive Bayes показала найнижчі результати серед усіх протестованих алгоритмів. Загальна точність класифікації склала 92.27%, але дисбаланс у показниках для окремих класів чітко помітний. Для класу 1 (тобто правдивих новин) модель досягла високого значення recall — 0.98687, тобто майже всі правдиві новини були правильно ідентифіковані. Проте значення precision при цьому було помітно нижчим — 0.88611, що вказує на велику кількість хибнопозитивних (FP) випадків: модель часто помилково відносила фейкові новини до правдивих.

Для класу 0 (фейкових новин) ситуація протилежна. Значення precision було високим — 0.98116, але recall становив лише 0.84355. Це означає, що майже всі передбачення фейкових новин справді були фейковими, однак значна частина фейків не була виявлена, модель пропустила їх, помилково вважаючи за правду.

Такі результати добре узгоджуються з основною ідеєю наївного баєсівського класифікатора: модель надзвичайно чутлива до ймовірностей, отриманих із частот ознак у кожному класі. Якщо правдиві новини статистично домінують у навчальних даних або мають більш “однорідні” шаблони, Naive Bayes може переоцінювати ймовірність того, що новина є правдивою. Це особливо помітно у високому значенні recall для класу 1.

Можна також припустити, що Naive Bayes погано впорався із взаємозалежними ознаками. Оскільки модель передбачає незалежність між усіма ознаками, вона не здатна захопити складніші залежності між словами, які могли б бути важливими для виявлення фейків. Це могло призвести до численних помилок саме в тих випадках, де фейкові новини мають словесні структури, подібні до правдивих.

Результати класифікації Naïve Bayes наведено на рисунку 5.3.

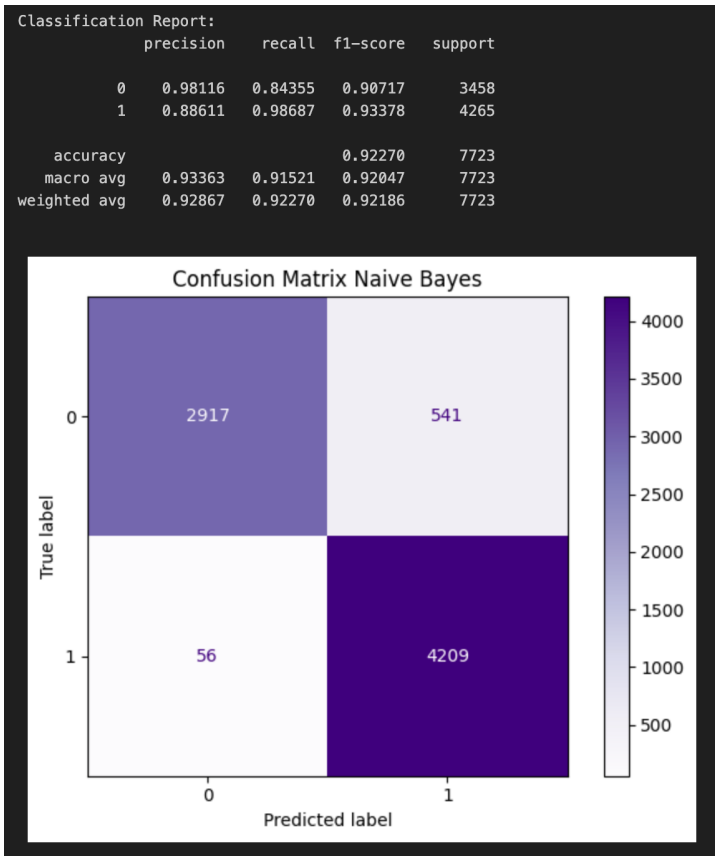

Рисунок 5.3 — Результати класифікації Naïve Bayes

Логістична регресія. Логістична регресія показала досить високі результати класифікації в порівнянні з Naïve Bayes, досягнувши загальної точності 98.65% на тестовій вибірці. Це один з найвищих показників серед усіх розглянутих моделей.

Розглядаючи метрики детальніше, для класу 0 (фейкові новини) precision становить 0.99006, а recall — 0.97976. Це вказує на те, що модель майже не помиляється, коли класифікує новини як фейкові, і водночас виявляє переважну

більшість дійсно фейкових прикладів. Для класу 1 (правдиві новини) precision становить 0.98372, а recall — 0.99203, що означає ще кращу здатність моделі виявляти правдиву інформацію з мінімальною кількістю помилкових негативних передбачень.

Такі результати, ймовірно, зумовлені тим, що логістична регресія добре масштабується на великі розріджені ознаки, зокрема в задачах обробки тексту. На відміну від попереднього класифікатора, вона не передбачає незалежність ознак і може враховувати складніші взаємозв'язки між словами в тексті.

Результати класифікації логістичної регресії наведено на рисунку 5.4.

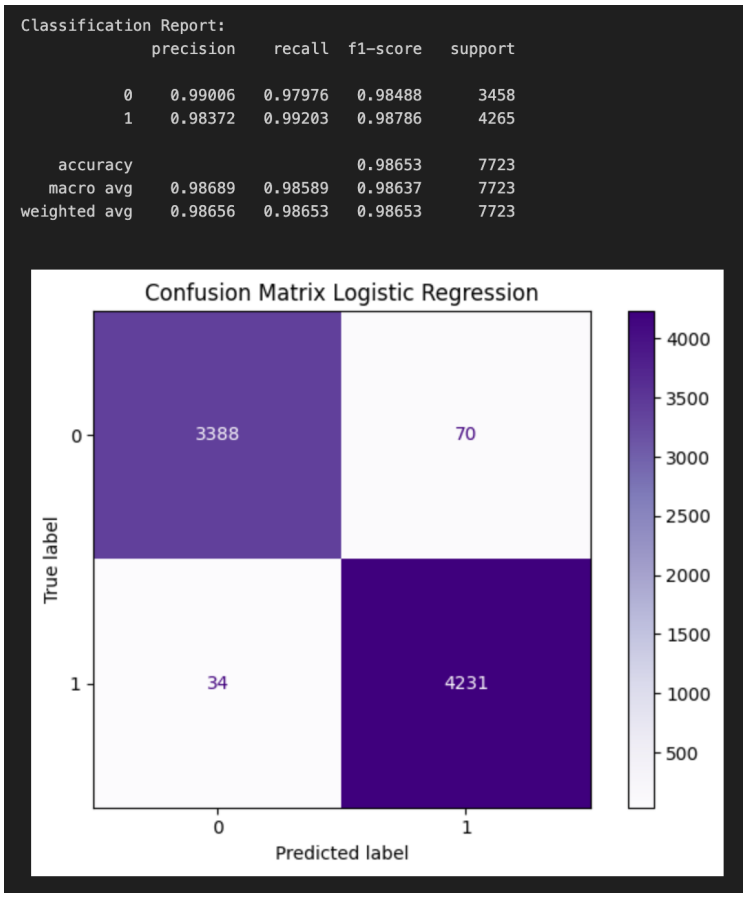

Рисунок 5.4 — Результати класифікації логістичної регресії

Random Forest. Random Forest показав досить високі результати, хоча й трохи поступився логістичній регресії та SVM, що буде розглянуто пізніше. Загальна точність склала 98.28%, що виглядає досить непогано, однак детальніша оцінка

класифікації по класах відкриває цікаві нюанси. Для фейкових новин (клас 0) модель досягла precision 0.99288, тобто практично всі новини, які вона позначила як фейкові, справді були такими. Проте recall тут нижчий — 0.96848. Це означає, що певна кількість фейкових новин залишилася непоміченою.

У випадку з класом 1, тобто правдивими новинами, все виглядає навпаки. Recall сягає 0.99437 — майже повна ідентифікація правди. Precision же трохи нижчий (0.97494), що натякає на можливу тенденцію моделі класифікувати частину фейкових новин як правдиві.

Причини таких характеристик можна шукати в самій природі Random Forest. Це ансамблева модель, яка будує багато дерев рішень і "голосує", щоб дійти до класифікації. Вона добре працює з табличними даними, здатна вловлювати складні взаємозв’язки, але при цьому може бути трохи консервативною, особливо в умовах класового дисбалансу. І так як в датасеті було трохи більше прикладів правдивих новин, це могло вплинути на схильність моделі класифікувати фейки.

Результати класифікації випадкових лісів наведено на рисунку 5.5.

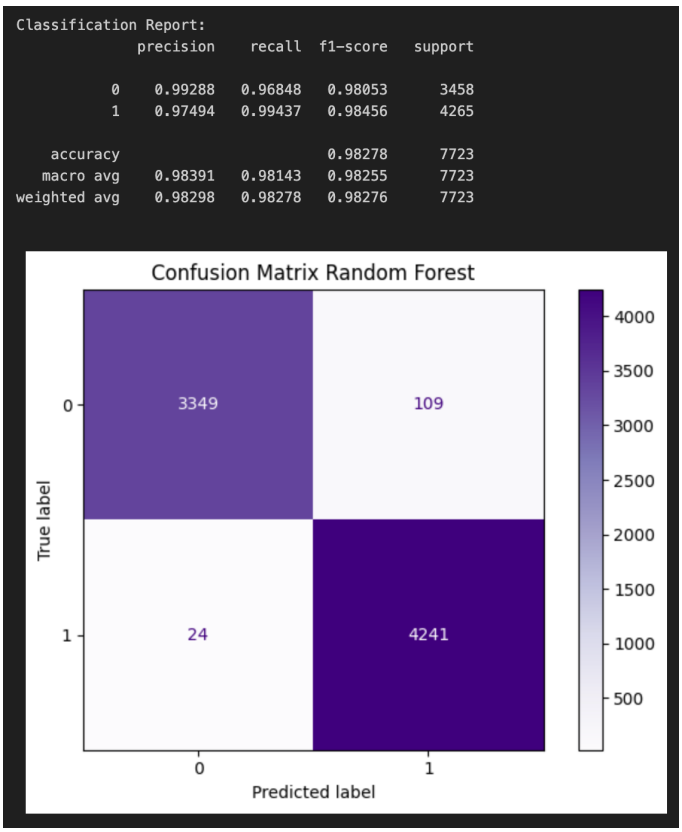

Рисунок 5.5 — Результати класифікації випадкових лісів

Support Vector Machine. Під час тестування модель Support Vector Machine показала найвищу точність серед усіх алгоритмів — 99.29%. Це означає, що загалом вона майже не помилялася при класифікації новин. Якщо подивитися детальніше, стає зрозуміло, що цей результат ґрунтується не лише на точності, а й на дуже збалансованій поведінці моделі щодо обох класів. У випадку фейкових новин precision досяг 0.99592, а recall — 0.98814. Іншими словами, модель не тільки рідко помиляється, коли стверджує, що новина фейкова, але й майже завжди знаходить усі фейки.

Для правдивих новин показники ще вищі: recall становить 0.99672, а precision — 0.99045. Це говорить про майже повну впевненість моделі в ідентифікації правди і про здатність робити це з дуже невеликим числом хибнопозитивних (FP) рішень.

Ймовірним поясненням таких результатів є те, що SVM добре підходить для задач із чітко відокремлюваними класами, особливо у багатовимірному просторі ознак. Завдяки тому, що текстові векторизовані дані являють собою багатовимірні вектори, моделі вдалося легко віднайти гіперплощину, яка з максимальною точністю розділяє фейкові та правдиві новини застосовуючи просту лінійну ядрову функцію (kernel function). Крім цього, SVM також часто демонструє стійкість до перенавчання, що важливо при роботі з текстовими даними, де легко втратити баланс між узагальненням і запам’ятовуванням.

Результати класифікації SVM наведено на рисунку 5.6.

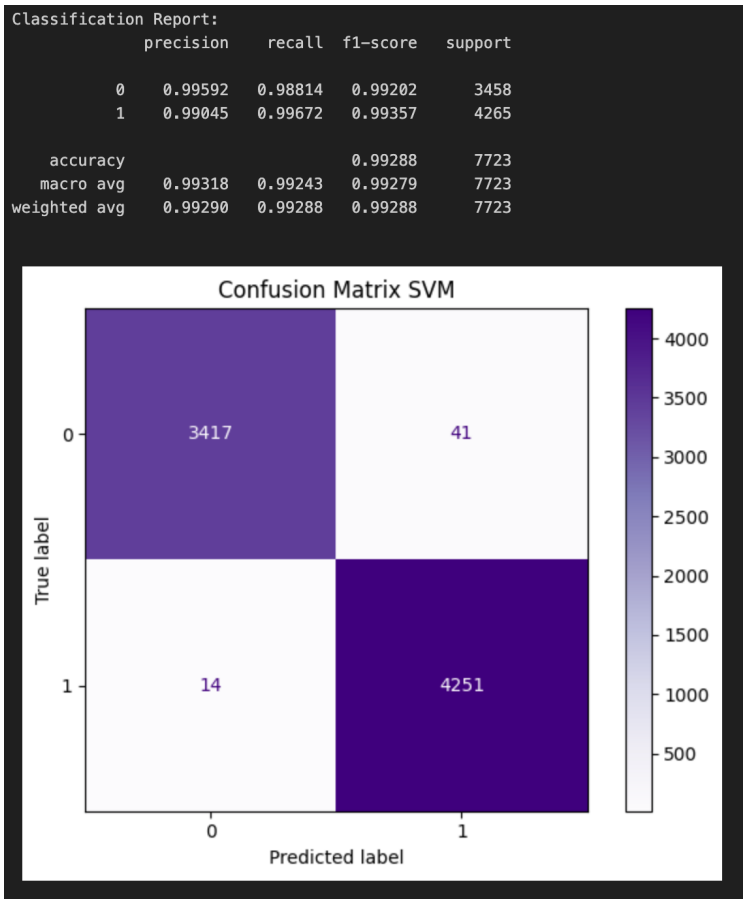

Рисунок 5.6 — Результати класифікації SVM

GRU. GRU показала досить непогані результати, особливо у розпізнаванні правдивих новин. Recall класу 1 досяг 0.99484, це означає те що майже жодна правдива новина не залишилася непоміченою. Precision трохи нижчий, але все одно на високому рівні, 0.98354, отже помилок у вигляді хибнопозитивних фейків було небагато. Щодо класу 0 ситуація теж доволі стабільна, хоча видно невелике просідання в значеннях recall до 0.97947. Це може свідчити про те, що модель дещо схильна плутати фейки з правдою, але не критично. Загальна точність 98.8% є цілком конкурентним результатом, особливо з огляду на те, що GRU є простішою альтернативою LSTM, яка швидше навчається і потребує менше ресурсів. Схоже, для цього типу задачі її здатності вловлювати послідовні зв’язки в тексті цілком достатньо. Ймовірно, структура новин дозволяє моделі розпізнавати певні патерни або стилістичні елементи, які типові саме для правдивих новин, або певну роль зіграла кількісна перевага даного класу.

Результати класифікації GRU наведено на рисунку 5.7.

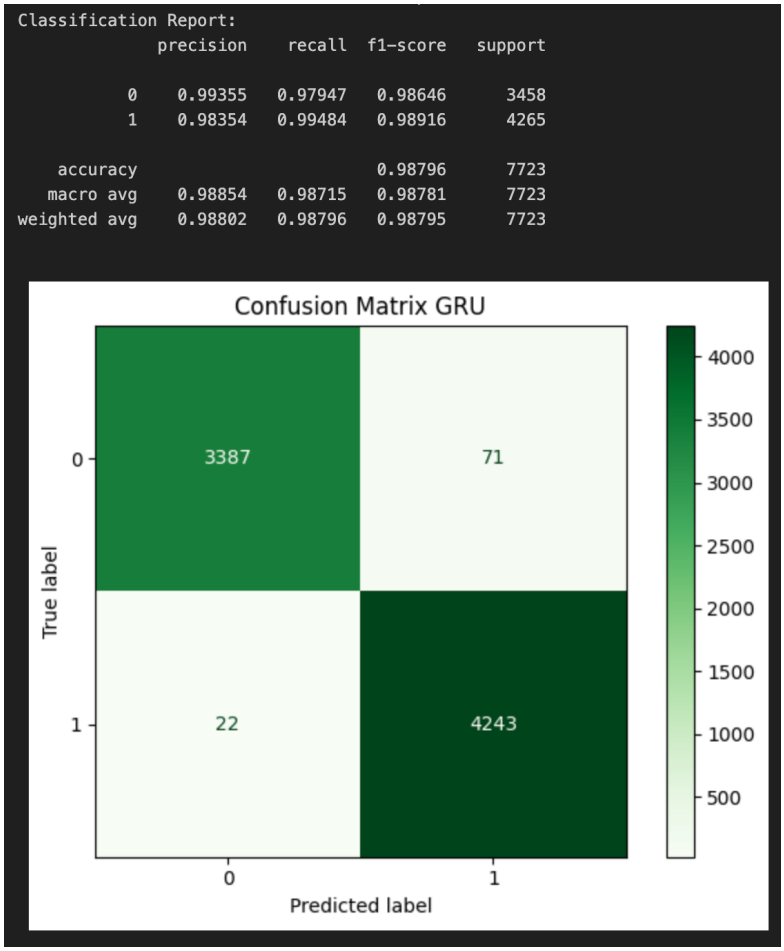

Рисунок 5.7 — Результати класифікації GRU

LSTM. LSTM показала впевнені результати, хоча й трохи гірші порівняно з GRU, що трохи неочікувано. Модель чітко і стабільно класифікує обидва класи: recall для фейкових новин становить 0.98236, для правдивих — 0.98265, тобто значення є практично однаковими. Це може свідчити про те, що LSTM вдалося вловити загальну структуру текстів, але без надмірної фіксації на якихось специфічних рисах одного з класів. Precision теж є досить високим, особливо для правдивих новин, де значення сягнуло 0.98565. Тобто модель не надто схильна класифікувати справжні новини як фейкові. І все ж, попри складнішу архітектуру, LSTM не перевершила GRU. Ймовірно, через більшу кількість параметрів і відповідну потребу в тоншій оптимізації. Загалом, у LSTM є потенціал, але тут вона виглядає трохи надто обережно. Проте її стабільність і симетрія між класами є

помітною перевагою, модель працює передбачувано, симетрично на всіх класах, і значення загальної accuracy в 98.25% підтверджує це.

Результати класифікації LSTM наведено на рисунку 5.8.

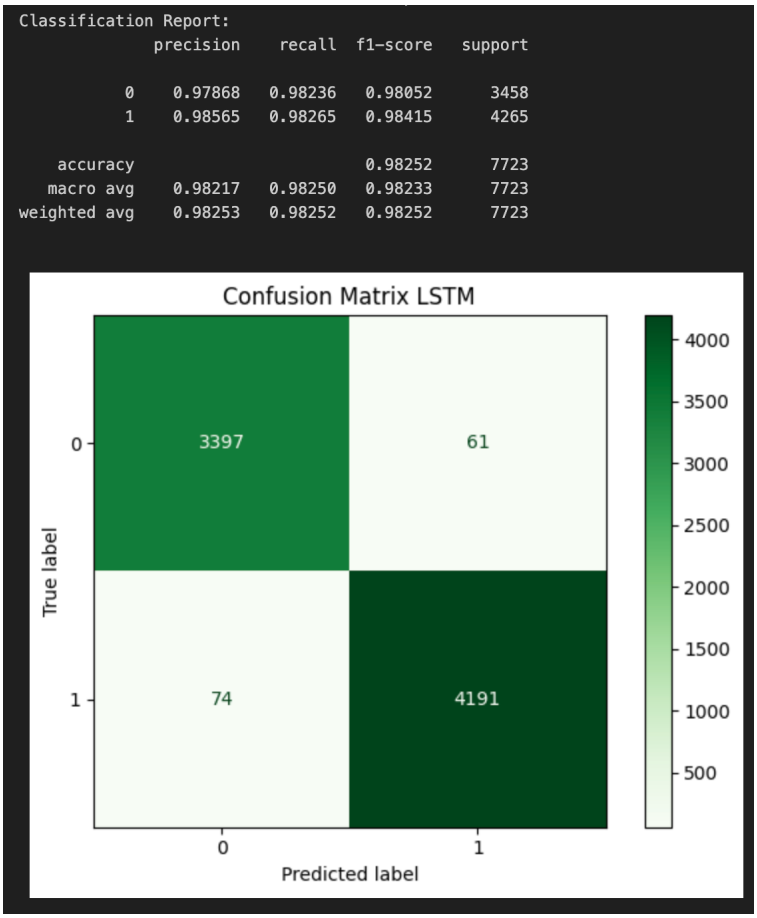

Рисунок 5.8 — Результати класифікації LSTM

Bi-LSTM. Bi-LSTM демонструє результат, що балансує між агресивністю і обережністю. Модель добре розуміє контекст із двох боків і зліва, і справа, що особливо корисно для природної мови, де важливими можуть виявитись як попередні, так і наступні слова. Вона точно ідентифікує фейки: recall для класу 0 склав 0.98988, тобто майже ідеальне виявлення, precision трохи нижчий — 0.97244, що означає невелику кількість хибнопозитивних результатів. Водночас для правдивих новин precision сягнув 0.99167, але recall дещо нижчий — 0.97726. Загальна точність склала 98.29%, що цілком на рівні з LSTM, хоча видно невелику асиметрію між класами. Двонаправлена архітектура вловлює більше нюансів, але,

схоже, не завжди розставляє правильні акценти. Ймовірно, деякі правдиві новини мають формальні або стилістичні ознаки, які модель інтерпретує як підозрілі, особливо якщо в лівому і правому контексті трапляються патерни, типові для фейків. Попри це, Bi-LSTM поводиться цілком узгоджено й демонструє сильні класифікаційні властивості, особливо в питанні виявлення фейкових новин.

Результати класифікації Bi-LSTM наведено на рисунку 5.9.

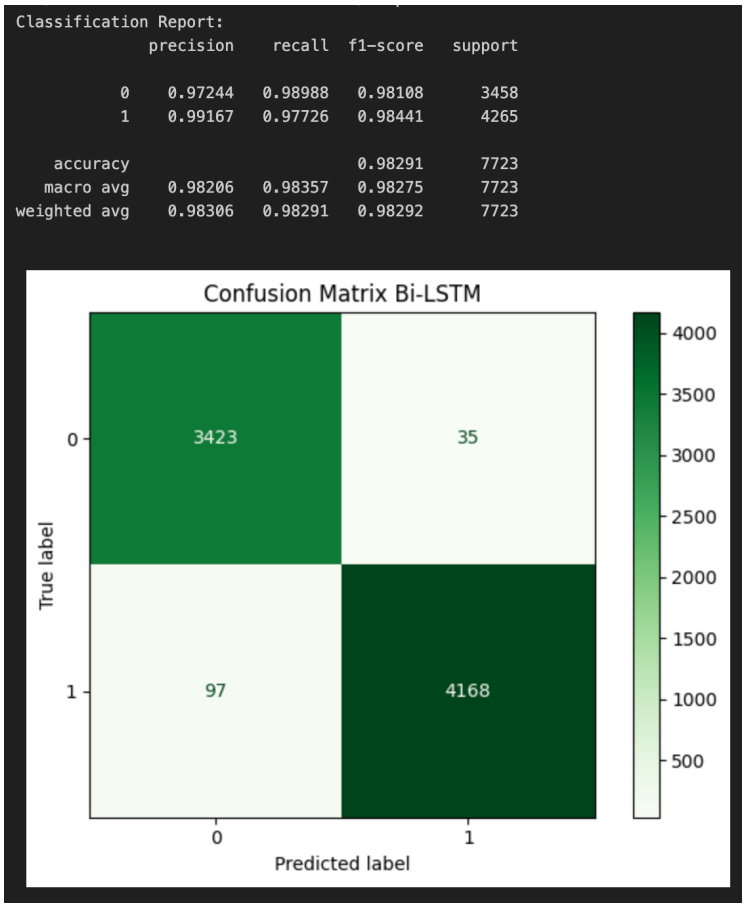

Рисунок 5.9 — Результати класифікації Bi-LSTM

5.2 Розгортання та огляд готового додатку

Для розгортання застосунку було використано Docker, що дозволило створити ізольоване середовище з усіма необхідними залежностями та стабільною конфігурацією. Середовище було визначено у Dockerfile, де окреслюється базовий образ, додаються системні компоненти, копіюється вміст проєкту та виконується

інсталяція залежностей. Усі Python-бібліотеки чітко задекларовані у файлі requirements.txt, що забезпечує прозорість і відтворюваність. Сам процес керується за допомогою Makefile, з його допомогою все збирається, встановлюється й запускається за одну команду. Це особливо зручно під час тестування, коли хочеться мінімізувати ручну роботу.

Готовий застосунок реалізовано у вигляді вебінтерфейсу, що дозволяє користувачеві вставити текст новини та отримати прогноз про її достовірність. Після натискання кнопки результат з'являється одразу, без перезавантаження сторінки, що забезпечує зручність використання. Інтерфейс мінімалістичний, без зайвих елементів, що дозволяє сконцентруватися лише на головній задачі застосунку - оцінці новинного повідомлення. Завдяки інтеграції попередньо збережених моделей машинного навчання застосунок працює в режимі реального часу й демонструє стабільну продуктивність навіть за умов обмежених ресурсів.

Перевіримо роботу застосунку згенерувавши фейкову новину за допомогою ChatGPT а також візьмемо справжню новину з достовірного джерела і спробуємо класифікувати їх, взявши для прикладу модель SVM.

Приклад класифікації фейкової новини наведено на рисунку 5.10.

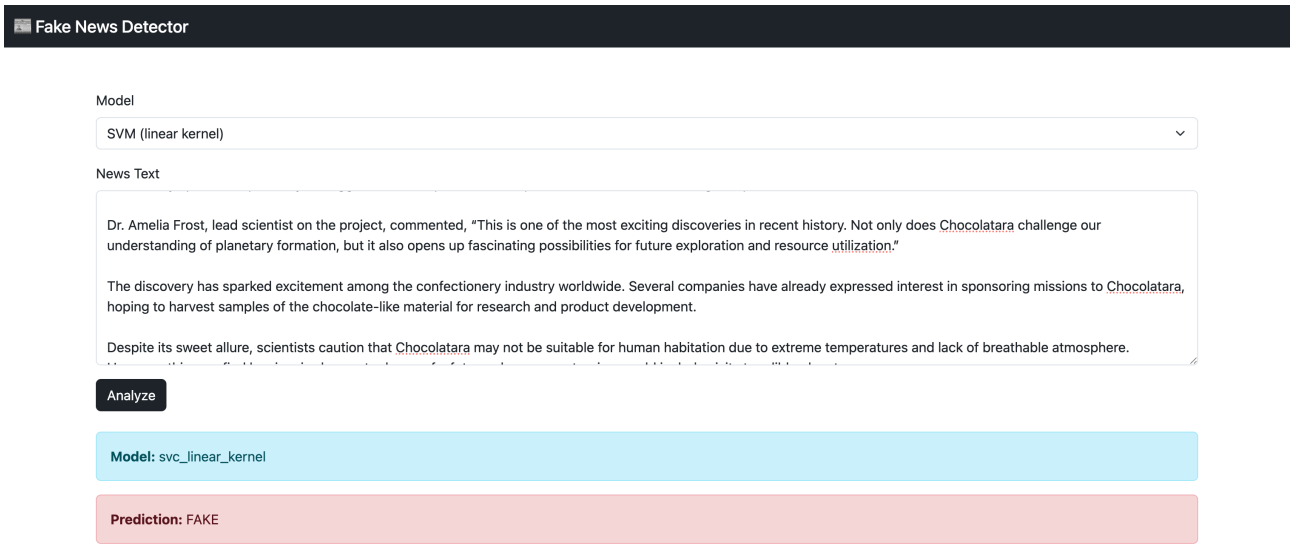

Рисунок 5.10 — Приклад класифікації фейкової новини

Також користувач може обрати іншу модель зі спадного меню. Для прикладу оберемо LSTM і спробуємо класифікувати справжню новину.

Приклад класифікації справжньої новини наведено на рисунку 5.11.

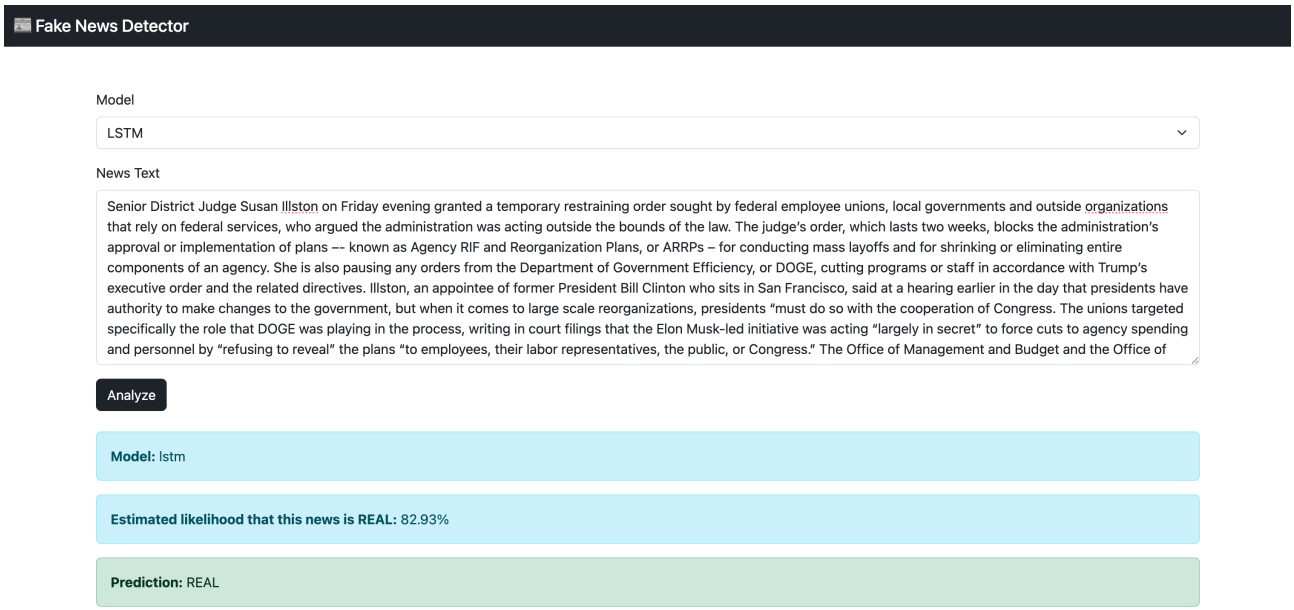

Рисунок 5.11 — Приклад класифікації справжньої новини

Висновки до розділу 5

У цьому розділі було проведено комплексне оцінювання побудованої системи виявлення фейкових новин, що охоплює одразу кілька важливих аспектів: як функціональне розгортання, так і систематичне тестування обраних моделей на базі метрик класифікації. Для оцінювання точності роботи моделей застосовано hold-out підхід, який дозволив чітко відокремити дані для навчання від даних для тестування і уникнути витоку інформації з майбутнього. Незважаючи на свою простоту, цей метод продемонстрував ефективність і був достатнім у межах обсягу наявної вибірки.

Для кожної моделі було обчислено precision, recall, f1-score, а також загальну accuracy, що дало змогу глибше зрозуміти, як саме алгоритми справляються з класифікацією фейкових та правдивих новин. Порівняння результатів показало, що

класичні моделі, такі як Logistic Regression та SVM, забезпечують дуже високий рівень точності при низьких витратах ресурсів. Натомість глибокі нейронні архітектури на кшталт GRU та LSTM, хоч і незначно, але поступаються за стабільністю результатів, при цьому вимагаючи значно більше ресурсів для навчання. SVM, зокрема, досягла найвищих значень метрик серед усіх протестованих підходів, що, ймовірно, свідчить про добре віддільні кластери у багатовимірному просторі векторизованих ознак.

Окрему увагу було приділено розгортанню системи. Створено повноцінний вебдодаток із використанням FastAPI, який працює всередині контейнера Docker. Завдяки використанню Makefile процес налаштування та запуску зведено до кількох простих команд, що значно спрощує супровід і дає змогу швидко масштабувати застосунок або переносити його між середовищами. Такий підхід забезпечує не лише відтворюваність експериментів, а й створює технічну базу для подальшого розвитку проєкту, зокрема у напрямку інтеграції в більші інформаційні системи.

ВИСНОВКИ

У межах даної бакалаврської роботи було реалізовано повноцінну систему для автоматичного виявлення фейкових новин на основі методів машинного навчання. Цей проєкт охопив увесь цикл розробки: від збору та попередньої обробки даних до навчання моделей, оцінювання результатів та розгортання вебдодатку. Основною метою було дослідити, наскільки ефективно різні алгоритми машинного навчання здатні вирішувати задачу бінарної класифікації новин, а також побудувати практичний застосунок, що здатен автоматично аналізувати введений текст і класифікувати його як фейковий або правдивий.

Аналіз літератури та попередніх досліджень у цій галузі показав, що проблема поширення дезінформації, зокрема в цифровому просторі, є критично актуальною. З розвитком соціальних мереж і платформ для миттєвого розповсюдження контенту, фейкові новини стали не лише медійною, а й соціальною загрозою, що здатна впливати на політичні процеси, безпеку та громадську думку. У цьому контексті автоматизовані засоби ідентифікації фейкової інформації набувають все більшого значення.

У процесі роботи було реалізовано і порівняно кілька популярних моделей: Naive Bayes, Logistic Regression, Random Forest, SVM, а також рекурентні нейронні мережі GRU, LSTM і Bi-LSTM. Для кожної з них проведено оцінювання на основі метрик точності, повноти, F1-міри та загальної правильності класифікації. Результати показали, що найкращу якість розпізнавання демонструють моделі SVM і Logistic Regression, що забезпечили точність понад 99% та стабільну роботу як із правдивими, так і з фейковими повідомленнями. Крім цього, SVM досягла найвищого f1-score серед усіх підходів, що свідчить про збалансованість роботи даної моделі на обох класах та ефективну побудову гіперплощини для розділення класів навіть у складному багатовимірному просторі ознак. Це особливо важливо у контексті текстових даних, де межі між класами можуть бути розмитими й непередбачуваними.

На відміну від класичних моделей, нейронні архітектури GRU, LSTM і Bi-LSTM продемонстрували хороші результати, але не показали істотного покращення в порівнянні з простішими підходами. Попри те, що вони були здатні уловлювати послідовні зв’язки в тексті, у межах доступного набору даних ця перевага виявилась недостатньо вираженою. Ймовірно, їх повний потенціал можна реалізувати на значно більших корпусах або у випадках, коли обробка довгострокових залежностей має вирішальне значення.

Для уникнення витоку інформації між навчальним і тестовим етапами було використано hold-out метод, який дозволив об’єктивно оцінити узагальнюючу здатність моделей. Поділ даних на навчальну і тестову вибірки у співвідношенні 80/20 забезпечив належний баланс між достатньою кількістю прикладів для навчання та достовірністю перевірки. Такий підхід є простим, проте ефективним, особливо у разі, коли обсяг даних є достатнім для адекватного представлення обох класів у кожній підмножині.

З технічної точки зору було створено повнофункціональний вебдодаток на базі FastAPI, що дозволяє користувачеві взаємодіяти з системою через інтерфейс HTTP-запитів.

Підсумовуючи, можна зробити висновок, що побудована система показала себе як ефективний і технічно надійний інструмент для автоматичного виявлення фейкових новин. Обрані моделі машинного навчання виявилися здатними точно класифікувати текстові повідомлення з різними стилістичними та семантичними особливостями. Реалізація вебінтерфейсу та контейнеризація дозволяють розглядати цей проєкт як основу для впровадження у реальні інформаційні системи. Надалі система може бути розширена підтримкою багатомовних даних, додаванням активного навчання або інтеграцією з потоковими джерелами новин, що зробить її ще більш актуальною та корисною в умовах сучасного інформаційного середовища.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fake news detection using python and machine learning / J. Jouhar et al. Procedia computer science. 2024. Vol. 233. P. 763–771. URL: https://doi.org/10.1016/j.procs.2024.03.265 (дата звернення: 20.04.2025).
2. Aldwairi M., Alwahedi A. Detecting fake news in social media networks. Procedia computer science. 2018. Т. 141. С. 215–222. URL: https://doi.org/10.1016/j.procs.2018.10.171 (дата звернення: 20.04.2025).
3. Dutta R., Majumder M. ABiLSTM with BERT embedding for classification of imbalanced COVID-19 rumors. Current applied science and technology. 2024. С. e0259284. URL: https://doi.org/10.55003/cast.2024.259284 (дата звернення: 21.04.2025).
4. Disinformation, fakes and propaganda identifying methods in online messages based on NLP and machine learning methods / V. Vysotska та ін. International journal of computer network and information security. 2024. Т. 16, № 5. С. 57–85. URL: https://doi.org/10.5815/ijcnis.2024.05.06 (дата звернення: 21.04.2025).
5. Wanda P., Diqi M. DeepNews: enhancing fake news detection using generative round network (GRN). International Journal of Information Technology. 2024. URL: https://doi.org/10.1007/s41870-024-02017-3 (дата звернення: 21.04.2025).
6. Mousavi E. NLP series: day 4 – stopword removal and normalization. Medium. URL: https://medium.com/@ebimsv/nlp-series-day-4-stopword-removal-and-normalization-418e0ec0016b (дата звернення: 15.05.2025).
7. Jain A. TF-IDF in NLP (term frequency inverse document frequency). Medium. URL: https://medium.com/@abhishekjainindore24/tf-idf-in-nlp-term-frequency-inverse-document-frequency-e05b65932f1d (дата звернення: 17.05.2025).
8. TfidfVectorizer. scikit-learn. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html (дата звернення: 20.05.2025).

9. Tf.keras.preprocessing.text.Tokenizer | tensorflow v2.16.1. TensorFlow. URL: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer (дата звернення: 25.05.2025).
10. Pipeline. scikit-learn. URL: https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html (дата звернення: 27.05.2025).
11. Wei D. Demystifying the Adam Optimizer in Machine Learning. *Medium*. URL: https://medium.com/@weidagang/demystifying-the-adam-optimizer-in-machine-learning-4401d162cb9e (дата звернення: 28.05.2025).
12. Allibhai J. Holdout vs. Cross-validation in Machine Learning. *Medium*. URL: https://medium.com/@jaz1/holdout-vs-cross-validation-in-machine-learning-7637112d3f8f (дата звернення: 28.05.2025).
13. Murel J. Ph.D, Kavlakoglu E. What is a confusion matrix? | IBM. *IBM - United States*. URL: https://www.ibm.com/think/topics/confusion-matrix (дата звернення: 30.05.2025).
14. GeeksforGeeks. Evaluation Metrics in Machine Learning - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/metrics-for-machine-learning-model/ (дата звернення: 30.05.2025).
15. Sefidian A. M. Understanding Micro, Macro, and Weighted Averages for Scikit-Learn metrics in multi-class classification with example. *Amir Masoud Sefidian - Sefidian Academy*. URL: https://iamirmasoud.com/2022/06/19/understanding-micro-macro-and-weighted-averages-for-scikit-learn-metrics-in-multi-class-classification-with-example/ (дата звернення: 30.05.2025).