Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут
комп'ютерних наук та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему:«Чат-бот розпізнавання дезінформації у соціальних мережах засобами NLP

Виконав: студент 6 курсу групи КН-61м
спеціальності
122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Псюк Д. Ю.
(прізвище та ініціали)

Керівник Процах Н. П.
(прізвище та ініціали)

Рецензент Сторожук О. Л.
(прізвище та ініціали)

Львів – 2025

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій

Кафедра комп'ютерних наук

Рівень вищої освіти другий (магістреський)

Спеціальність 122 "Комп'ютерні науки"

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувачка кафедри КН

_____ Борецька І. Б.

"10" грудня 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Псюку Дмитру Юрійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи Чат-бот розпізнавання дезінформації у соціальних мережах засобами NLP

керівник роботи *Процах Наталія Петрівна, д.ф.-м.н., проф*,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "29" квітня 2025 року № С-288

2. Термін подання студентом роботи 10 грудня 2025 р.

3. Вихідні дані до роботи Розробити систему розпізнавання дезінформації в соціальних мережах із використанням методів обробки природної мови (NLP). Реалізувати програмний застосунок у вигляді бота, що забезпечує приймання текстових повідомлень, їхню попередню обробку, переклад, класифікацію за допомогою навченої моделі та повернення результатів користувачу.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Стан проблемної області

Інформаційне забезпечення

Математичне забезпечення

Програмне забезпечення

Розроблення стартапу-проєкту

Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Підготовка матеріалів до доповіді

6. Дата видачі завдання 1 травня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	02.05.25 – 17.05.25	Виконано
2	Формування та налаштування датасету	18.05.25 – 02.06.25	Виконано
3	Розширення датасету та підготовка до навчання моделі	03.06.25 – 19.06.25	Виконано
4	Навчання та оптимізація моделі NLP	20.06.25 – 15.07.25	Виконано
5	Розробка серверної частини системи	15.07.25 – 20.08.25	Виконано
6	Розробка клієнтської частини	21.08.25 – 18.09.25	Виконано
7	Тестування системи та виправлення помилок	19.09.25 – 29.10.25	Виконано
8	Оформлення пояснювальної записки	30.10.25 – 25.11.25	Виконано

Студент _____ Псюк Д.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Процах Н. П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Магістерська робота містить 75 сторінок пояснювальної записки, 21 рисунок, 1 додаток, 20 джерел.

У магістерській роботі наведено розробку системи для автоматичного розпізнавання дезінформації у текстах соціальних мереж із використанням методів NLP та машинного навчання. Метою проєкту було створення сервісу, що аналізує текстові повідомлення та визначає ймовірність їх фейковості. У роботі сформовано власний датасет, виконано попередню обробку текстів, побудовано модель класифікації на основі TF-IDF і логістичної регресії та проведено оптимізацію порогу прийняття рішень. Серверна частина реалізована на Python (Flask), а клієнтська — у вигляді бота, який забезпечує взаємодію з користувачем та адміністрування системи. Розроблена система демонструє високу точність і може бути використана як інструмент швидкої оцінки достовірності інформації.

Ключові слова: NLP, машинне навчання, фейкові новини, класифікація текстів, бот, TF-IDF.

ABSTRACT

The thesis contains 75 pages of explanatory note, 21 figures, 1 appendix, 20 sources.

This thesis presents the development of a system for automatic detection of misinformation in social-media texts using NLP and machine learning methods. The goal was to create a service that analyzes messages and determines the probability of them being fake.A custom dataset was created, text preprocessing was applied, and a TF-IDF–based logistic regression model was built with threshold optimization. The server side was implemented in Python (Flask), while the client interface was developed as a bot.The system demonstrates high accuracy and can be used as a tool for rapid credibility assessment of online information.

Keywords: NLP, machine learning, misinformation detection, text classification, bot, Flask, TF-IDF

ТЕХНІЧНЕ ЗАВДАННЯ

Необхідно розробити інтелектуальну систему для автоматичного визначення фейкових новин на основі аналізу текстових повідомлень із використанням методів обробки природної мови (NLP) та машинного навчання, а саме:

1. розробити структуру та сформувати збалансований датасет текстів правдивих і фейкових новин;
2. реалізувати модуль попередньої обробки тексту (нормалізація, очищення, токенізація, видалення шуму) та його векторизації на основі TF-IDF;
3. розробити та навчити модель бінарної класифікації текстів, виконати калібрацію ймовірностей та оптимізацію порога прийняття рішень, оцінити якість за метриками Accuracy, Precision, Recall, F1-score, ROC-AUC;
4. створити серверний модуль на основі Flask для обробки запитів, виконання класифікації, роботи з датасетом та перенавчання моделі;
5. розробити бота для взаємодії з користувачем, який забезпечує надсилання текстів на аналіз, отримання результатів та доступ адміністратора до керування датасетом і запуску перенавчання;
6. реалізувати механізм захисту від дублікатів повідомлень і валідації вхідних даних (мінімальна кількість слів, обмеження довжини тексту);
7. провести експериментальні дослідження якості роботи системи, проаналізувати результати та сформулювати рекомендації щодо подальшого розвитку й практичного використання розробленого рішення.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ 8
ВСТУП 9
РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ 11
1.1 Проблема дезінформації у сучасному інформаційному просторі 11
1.2 Поведінкові та соціальні чинники поширення фейків 13
1.3 Організовані інформаційні атаки та технічні труднощі їх виявлення 14
1.4 Особливості українського інформаційного середовища 14
1.5 Обмеження сучасних систем та необхідність гнучких рішень 15
Висновки до розділу 16
РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ 17
2.1 Загальна характеристика інформаційного забезпечення 17
2.2 Вхідні та вихідні дані системи 19
2.3 Використані програмні інструменти, фреймворки та технології 22
2.4 Джерела та структура текстових даних 25
2.5 Інформаційні процеси, обробки та взаємодії компонентів системи 27
Висновки до розділу 29
РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ 31
3.1 Теоретичні основи класифікації текстової інформації 31
3.2 Математичні моделі подання тексту 33
3.3 Побудова простору ознак для класифікації текстових повідомлень 35
3.4 Математичний апарат логістичної регресії 36
3.5 Балансування класів та нормалізація ваг у задачі класифікації 37
3.6 Оцінювання якості математичної моделі класифікації 39
3.7 Математична інтерпретація навчання та перенавчання моделі 41
Висновки до розділу 41
РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ 43
4.1 Архітектура та загальна структура програмної системи 43
4.2 Реалізація серверного застосунку 44

4.3 Модуль машинного навчання та робота з моделлю 48
4.4 Адміністративний функціонал, робота з датасетом та обробка помилок 52
Висновки до розділу 56
РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ 57
5.1 Опис концепції та ціннісної пропозиції продукту 57
5.2 Аналіз цільової аудиторії та сценаріїв застосування 58
5.3 Аналіз ринку та конкурентів 59
5.4 Бізнес-модель та план розвитку продукту 61
Висновки до розділу 63
ВИСНОВОК 64
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 65
ДОДАТКИ 67

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

AI — Artificial Intelligence (штучний інтелект);

API — Application Programming Interface (програмний інтерфейс взаємодії);

БД — база даних;

Bot — програмний агент для автоматизованої взаємодії з користувачем;

CSV — Comma-Separated Values (формат збереження табличних даних);

ML — Machine Learning (машинне навчання);

NLP — Natural Language Processing (обробка природної мови);

TF-IDF — Term Frequency–Inverse Document Frequency (метод векторизації текстів);

LR — Logistic Regression (логістична регресія — модель класифікації);

ROC-AUC — Receiver Operating Characteristic — Area Under Curve (метрика якості моделі);

F1-score — гармонійне середнє між точністю (Precision) та повнотою (Recall);

HTTP — HyperText Transfer Protocol (протокол передачі даних у вебі);

JSON — JavaScript Object Notation (формат передачі структурованих даних);

Flask — фреймворк Python для створення веб-сервісів та API;

Tokenizer — модуль розбиття тексту на токени;

Vectorizer — векторизатор текстів (перетворення тексту на числові ознаки);

Dataset — набір даних для навчання та тестування моделі;

Admin-mode — режим керування даними та перенавчанням для адміністратора.

ВСТУП

У сучасному інформаційному просторі значну частину контенту становлять новини та повідомлення, що поширюються у соціальних мережах та онлайн-платформах. Разом із зростанням швидкості комунікації стрімко збільшується і кількість фейкових новин, які формують хибні уявлення, впливають на суспільну думку та можуть спричиняти небезпечні наслідки. Традиційні методи ручної перевірки інформації вже не забезпечують необхідної швидкості та масштабованості, тому виникає потреба у використанні автоматизованих систем визначення достовірності текстових повідомлень. Методи обробки природної мови (NLP) у поєднанні з алгоритмами машинного навчання дають змогу ефективно аналізувати зміст повідомлення, визначати характерні ознаки фейковості та здійснювати класифікацію текстів у режимі реального часу. Такі системи є важливими інструментами для боротьби з дезінформацією та можуть застосовуватися у журналістиці, освітніх проєктах, державних структурах і повсякденному користуванні.

У межах цієї магістерської роботи розроблено інтелектуальну систему автоматичного визначення фейкових новин. Вона поєднує методи NLP, машинного навчання, серверний модуль на Flask та зручний інтерфейс у вигляді бота. Система дозволяє користувачу надсилати текст для перевірки та отримувати оцінку достовірності з прогнозованою ймовірністю. Для адміністратора доступні функції управління датасетом, додавання нових прикладів та запуск перенавчання моделі. Модель машинного навчання побудована на основі TF-IDF векторизації та логістичної регресії, що забезпечує хороший баланс між точністю, швидкістю роботи та можливістю регулярного оновлення. Особливу увагу приділено попередній обробці текстів, оскільки коректне очищення та нормалізація даних значно підвищують якість класифікації. Крім того, у системі реалізовано механізм антидублікатів та валідації вхідних даних, що забезпечує стабільність у роботі моделі.

Практична цінність роботи полягає у створенні інструменту, який може використовуватися для швидкої перевірки достовірності текстів та як елемент захисту

інформаційного простору. Система легко масштабується, підтримує оновлення датасету та адаптується до нових типів фейкових повідомлень.

Метою роботи є розробка системи визначення фейкових новин на основі аналізу тексту з використанням NLP, машинного навчання та інтерактивного бот-інтерфейсу. Для досягнення поставленої мети виконано такі завдання:

- проаналізовано проблемну область та існуючі підходи до автоматичного визначення фейкових новин;
- сформовано та підготовлено датасет правдивих і фейкових текстів;
- реалізовано модуль попередньої обробки та векторизації даних;
- розроблено та навчено модель класифікації;
- створено серверний модуль для взаємодії з моделлю;
- розроблено інтерфейс бота для доступу до функціоналу системи;
- впроваджено механізм адміністрування та перенавчання моделі;
- здійснено оцінювання точності та ефективності системи.

Розроблена система є актуальним та корисним інструментом у протидії дезінформації й сприяє підвищенню якості споживання інформації у цифровому середовищі. Її використання дозволяє користувачам оперативно перевіряти підозрілі повідомлення та зменшувати вплив недостовірного контенту. Завдяки можливості розширення датасету та повторного навчання модель здатна адаптуватися до нових типів фейкових новин і залишатися ефективною у динамічному інформаційному просторі.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Проблема дезінформації у сучасному інформаційному просторі

У сучасному цифровому середовищі інформація поширюється надзвичайно швидко, а соціальні мережі та онлайн-платформи стали основними каналами комунікації та споживання новин. Разом зі зростанням обсягів інформації значно збільшилась і кількість недостовірних матеріалів, фейкових новин та маніпуляцій, що створюють суттєві загрози для суспільної безпеки, політичної стабільності та інформаційної культури громадян. Дезінформація стала глобальною проблемою, яка впливає на всі сфери — від економіки та медицини до політики та міжнародних відносин.

Фейкові новини поширюються швидше та активніше, ніж достовірні повідомлення, оскільки вони часто мають емоційно насичений характер, створюються з метою привернення уваги та маніпулювання громадською думкою. Психологічні механізми довіри до знайомих людей у соціальних мережах, поширення сенсаційних заголовків, відсутність критичного мислення у значної частини користувачів та небажання перевіряти джерела інформації сприяють масовому розповсюдженню фейків. Крім того, автоматизовані боти, скоординовані мережі акаунтів та алгоритми рекомендацій у соцмережах посилюють ефект залучення та забезпечують ще швидше розповсюдження маніпулятивного контенту.

Традиційний ручний фактчекінг, який здійснюється журналістами або аналітичними центрами, залишається важливою частиною боротьби з дезінформацією, проте його масштабованість обмежена. Людина не здатна опрацьовувати настільки великі обсяги інформації в реальному часі, що робить ручні методи недостатніми для сучасних викликів. Ситуація ускладнюється тим, що кількість фейкових повідомлень постійно зростає, а їх різноманітність і складність збільшується.

У зв’язку з цим актуальними стали автоматизовані системи визначення фейковості текстів, що використовують методи обробки природної мови та машинного навчання [15]. Сучасні алгоритми NLP дозволяють здійснювати очищення, нормалізацію та

структуризацію тексту, визначати статистичні закономірності, семантичні особливості та характерні ознаки дезінформації. Методи машинного навчання забезпечують можливість навчати моделі на великих масивах текстів, узагальнювати виявлені закономірності та прогнозувати, чи є новина правдивою або фейковою [4].

У світовій практиці для виявлення фейкових новин застосовуються такі підходи, як статистичні моделі, логістична регресія, дерева рішень, градієнтний бустинг, SVM, а також глибокі нейромережі, включаючи трансформерні моделі. Проте значна частина цих систем орієнтована на англомовні тексти, не має адаптації до української мови або не є відкритою для інтеграції в зовнішні проєкти. Багато платформ зосереджуються на ручному аналізі, не надають можливості автоматичного оновлення навчальної вибірки та не підтримують інтерактивну взаємодію з кінцевими користувачами [16].

У сучасних умовах ефективна система визначення фейкових новин має відповідати низці вимог. По-перше, вона повинна забезпечувати високу швидкість аналізу та точність класифікації, адже перевірка новин часто потребує оперативності. По-друге, система повинна бути адаптивною — підтримувати оновлення датасету, перенавчання моделі та врахування нових видів дезінформації. Важливо також забезпечити простий та доступний інтерфейс, який дозволяє пересічним користувачам швидко отримувати результат аналізу тексту. Крім того, система повинна бути здатна працювати з українською мовою, враховуючи її морфологічні та синтаксичні особливості.

Автоматизовані інструменти визначення фейкових новин все частіше інтегруються у модулі фактчекінгу, системи моніторингу соціальних мереж, новинні агрегатори та освітні проєкти. Вони дозволяють зменшити навантаження на аналітиків, підвищують оперативність реагування та точність оцінки достовірності контенту. Водночас важливою залишається проблема недостатнього обсягу якісних даних для навчання моделей українською мовою, що ускладнює створення високоточних систем саме для локального інформаційного простору.

У рамках цього дослідження розглянуто сучасний стан проблеми дезінформації, проаналізовано існуючі методи її виявлення та визначено обмеження традиційних та

автоматизованих підходів. Розглянуті інструменти дозволили сформувати вимоги до розроблюваної системи: вона має бути швидкою, точною, гнучкою, підтримувати оновлення навчальних даних, працювати з українською мовою та мати зручний спосіб взаємодії з користувачем.

Таким чином, стан проблемної області підтверджує актуальність створення системи автоматичного виявлення фейкових новин, яка поєднує методи NLP, машинного навчання та інтерактивні інструменти для динамічної роботи з текстами. Подальша розробка такої системи дозволяє підвищити рівень інформаційної безпеки, сприяти медіаграмотності користувачів та забезпечувати більш якісний контроль над інформаційними потоками у цифровому середовищі.

Подальший аналіз проблеми дезінформації дозволяє зрозуміти, що її вплив не обмежується лише поширенням неправдивих повідомлень. Наслідки фейків мають складний, часто прихований характер, що проявляється у зміні поведінкових моделей користувачів, розвитку недовіри до офіційних джерел, формуванні емоційних реакцій та розмиванні здатності критично оцінювати інформацію. Дезінформація не лише вводить в оману, але й створює довготривалі когнітивні викривлення, які можуть залишатися навіть після спростування фейку, оскільки користувачі схильні запам’ятовувати емоційний контекст, а не фактологічну точність.

1.2 Поведінкові та соціальні чинники поширення фейків

Причиною такої вразливості є особливості психологічного сприйняття інформації. Люди часто формують свою думку на основі перших вражень, емоційних оцінок або повідомлень, що відповідають їхнім переконанням. Більшість користувачів не перевіряє інформацію на достовірність, оскільки втома, перенасичення новинами та високий темп життя не залишають часу для ретельного аналізу. У результаті фейкові новини поширюються серед населення природно й практично без бар’єрів. Саме тому інструменти для автоматичного виявлення недостовірних повідомлень мають бути максимально простими у використанні, швидкими й доступними.

Важливим аспектом сучасного інформаційного середовища є те, що значна частина фейків не виникає випадково, а є частиною цілеспрямованих інформаційних кампаній. Такі кампанії можуть включати масове створення фейкових акаунтів,

застосування бот-мереж, поширення однотипних повідомлень у різних соціальних каналах, використання таргетованих оголошень або навіть провокативних мемів. Усе це створює ілюзію достовірності, оскільки користувачі починають сприймати повторювану інформацію як правдиву. Такі скоординовані дії часто спрямовані на політичний вплив, дискредитацію державних структур, штучне загострення суспільних конфліктів або формування панічних настроїв. Автоматизовані засоби аналізу текстів є одним із найефективніших інструментів протидії таким інформаційним операціям.

1.3 Організовані інформаційні атаки та технічні труднощі їх виявлення

Разом із тим виявлення фейкових новин має низку технічних складнощів. Дезінформація може бути як повністю вигаданою, так і частково правдивою, вирваною з контексту або поданою у маніпулятивній формі. Існують тексти, які стилістично виглядають правдоподібно, містять формально «офіційні» формулювання або цитати, що ускладнює їх автоматичну класифікацію. Особливо важко виявляти короткі фейкові повідомлення, так звані «мікрофейки», які складаються лише з кількох слів чи фраз, але здатні мати сильний емоційний вплив і швидко поширюватися серед користувачів. У таких випадках для моделі важливу роль відіграє якісна підготовка текстів, очищення від шуму та коректне векторне подання.

1.4 Особливості українського інформаційного середовища

Український сегмент Інтернету також має свою специфіку, що створює додаткові труднощі. Значна частина текстів містить суржик, запозичення, зменшувальні форми або локальні діалекти, які впливають на лексичну структуру й ускладнюють автоматичний аналіз. Часто новини та повідомлення створюються у розмовному стилі, з великою кількістю емоційних конструкцій, що унеможливлює застосування простих шаблонних методів. Крім того, українська мова має складну морфологічну структуру, що вимагає спеціальних підходів до нормалізації та токенізації текстів. Також важливим фактором є недостатній обсяг великих відкритих датасетів українською мовою, що змушує розробників створювати власні набори даних, адаптовані до реальних умов.

1.5 Обмеження сучасних систем та необхідність гнучких рішень

Сучасні автоматизовані системи виявлення дезінформації за кордоном часто не придатні до використання в українському інформаційному середовищі. Значна частина таких систем орієнтована на англомовні тексти, має обмежені можливості інтеграції, не враховує локальні особливості та не дозволяє швидко оновлювати модель при появі нових типів фейків. Крім того, більшість платформ працює як веб-сайти або аналітичні сервіси, що робить їх менш зручними для повсякденного користувача. У зв'язку з цим інтерактивні засоби, такі як боти, стають кращим рішенням, оскільки забезпечують просту взаємодію, миттєву відповідь і мінімальні вимоги до підготовки користувача.

Для побудови ефективної системи виявлення фейкових новин необхідно враховувати не лише лінгвістичні й технічні аспекти, а й поведінкові моделі користувачів. Інструмент має бути достатньо гнучким, щоб адаптуватися до нового контенту, підтримувати оновлення навчального набору та забезпечувати можливість перенавчання без порушення основної роботи системи. Також важливо, щоб механізми аналізу були інтерпретованими: користувач повинен розуміти, чому система визнала певне повідомлення фейковим, а адміністратор — мати можливість контролювати роботу моделі та її навчання.

Враховуючи всі зазначені аспекти, можна зробити висновок, що створення інтелектуальної системи для автоматичного визначення фейкових новин є актуальним завданням, яке потребує комплексного підходу. Використання методів NLP та машинного навчання дозволяє вирішити частину проблем, пов'язаних із масштабованістю та точністю, тоді як інтерактивні інтерфейси забезпечують зручність для користувача. З огляду на те, що структура та форма фейкових новин постійно змінюються, система повинна бути здатною до самооновлення, підтримувати розширення датасету та коригування моделей. У поєднанні ці підходи формують основу сучасного інструменту для підвищення інформаційної безпеки та стійкості суспільства до дезінформаційних впливів [5].

Висновки до розділу

У результаті аналізу стану проблемної області встановлено, що дезінформація є складним та багатогранним явищем, яке активно поширюється у цифровому середовищі завдяки соціальним мережам, інформаційним платформам та організованим маніпулятивним кампаніям. Психологічні особливості сприйняття інформації, висока швидкість її розповсюдження та низький рівень критичного мислення користувачів створюють сприятливі умови для масштабного впливу фейкових новин. Проаналізовані сучасні методи їх виявлення демонструють обмеження традиційних підходів та підтверджують необхідність застосування технологій обробки природної мови й машинного навчання, здатних забезпечити швидку, адаптивну та точну оцінку достовірності текстів. Особливої актуальності набуває створення інтелектуальних систем, орієнтованих на українську мову, які можуть динамічно оновлювати дані, враховувати місцевий контекст та забезпечувати зручну взаємодію з користувачем через інтерфейс бота. Таким чином, проведений аналіз підтверджує актуальність, необхідність і практичну значущість розробки системи автоматичного визначення фейкових новин, що послужить основою для подальших етапів дослідження та реалізації програмного рішення.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Загальна характеристика інформаційного забезпечення

Інформаційне забезпечення є одним із ключових складників створюваної системи автоматичного визначення фейкових новин, оскільки саме правильна організація даних, потоків інформації та інфраструктури обробки визначає точність, стабільність і надійність функціонування програмного комплексу. У межах розробленої системи інформаційне забезпечення охоплює як текстові дані, що аналізуються моделлю, так і технічні ресурси, необхідні для їх обробки, зберігання, передачі та оновлення. Основною особливістю є те, що система взаємодіє з користувачем у режимі реального часу за допомогою бота, тому будь-які процеси — від попередньої обробки тексту до формування прогнозу — повинні відбуватися швидко, безперебійно та з мінімальною затримкою.

Інформаційне забезпечення охоплює декілька взаємопов’язаних компонентів. Основна частина даних представлена текстовими повідомленнями, які користувачі надсилають для перевірки. Такі повідомлення характеризуються неоднорідністю, різноманітністю стилів, наявністю сленгу, орфографічних або пунктуаційних помилок, що вимагає спеціальної системи попереднього очищення. Окреме місце посідає навчальна вибірка, на якій тренується модель, оскільки саме її якість визначає точність класифікації. Крім текстових даних, інформаційне забезпечення включає параметри моделі, засоби векторизації, службові файли, лог-файли, додаткові конфігураційні структури та допоміжні дані.

Система використовує підхід, у якому всі інформаційні компоненти інтегровані у єдиний інформаційний комплекс. Це дозволяє забезпечити повноцінну безперервну взаємодію між модулями, а також реалізувати можливість автоматичного оновлення даних, повторного тренування моделі та розширення обсягу навчальної вибірки. Такий підхід робить систему гнучкою, масштабованою та здатною до адаптації в умовах швидко змінюваного інформаційного середовища.

Таким чином, інформаційне забезпечення створюваної системи формує основу її функціонування, забезпечуючи повний цикл роботи: від отримання сирих текстових даних до їх аналізу, класифікації, передачі результату та подальшого збагачення моделі новими прикладами. Саме тому детальне опрацювання цього розділу є необхідною умовою створення надійного програмного рішення.

Важливою складовою інформаційного забезпечення також є організація внутрішньої логіки зберігання даних. Оскільки система працює з текстовими повідомленнями, модельними файлами та службовими структурами, необхідно забезпечити чіткий поділ між динамічними та статичними елементами. Динамічні дані, такі як нові приклади фейкових або правдивих текстів, регулярно доповнюють навчальний набір і можуть впливати на результати подальшого навчання моделі. Статичні дані — це конфігураційні файли, параметри моделі та векторизатор, які зберігаються окремо й використовуються сервером як опорні елементи під час кожного запиту. Такий підхід дає змогу досягти структурованості й уникнути змішування даних, що може спричинити логічні помилки в роботі системи.

Не менш важливою є організація надійних механізмів логування, які записують ключові події: надходження запиту, результат класифікації, виникнення помилок, а також дії адміністратора, пов’язані з оновленням даних або запуском повторного навчання моделі. Логи дають змогу відслідковувати поведінку системи, аналізувати помилки та своєчасно виявляти можливі вразливості. Крім того, у випадку надання доступу до декількох адміністраторів лог-файли стають засобом контролю дій, що впливають на стан інформаційної інфраструктури.

Окрему роль у формуванні інформаційного забезпечення відіграє вимога щодо актуальності даних. Оскільки інформаційний простір змінюється динамічно, особливо в частині дезінформації, статичний датасет через певний час втрачає ефективність. Тому система повинна бути здатною періодично оновлювати навчальну вибірку через адміністративний інтерфейс, що дозволяє додавати нові тексти для тренування моделі. Цей механізм сталого оновлення забезпечує адаптивність системи та підвищує її здатність розпізнавати сучасні тенденції у створенні фейкових повідомлень [20].

Не менш важливим елементом інформаційного забезпечення виступає структура взаємодії між користувачем і сервером, яка реалізована через простий, але ефективний інтерфейс бота. Такий підхід дозволяє мінімізувати бар'єри у використанні системи: користувачу не потрібно встановлювати додаткові програми, переходити на веб-сайти чи працювати зі складними формами. Автоматизований механізм прийому текстів та відправки відповіді формує потік даних, який легко контролювати та масштабувати. Застосування бота як основного каналу комунікації також знижує навантаження на сервер, оскільки всі взаємодії відбуваються у вигляді коротких стандартних запитів.

У структурі інформаційного забезпечення важливо виділити також компоненти, що забезпечують збереження моделі та суміжних артефактів. Після кожного циклу повторного тренування оновлюється файл моделі, який містить оптимізовані вагові коефіцієнти, а також векторизатор, що відповідає за формування числових представлень текстів. Наявність таких окремих модулів дає можливість швидко змінювати або вдосконалювати модель без необхідності перебудовувати всю інфраструктуру. Збереження надійних артефактів також забезпечує відтворюваність результатів, що є критично важливим у контексті машинного навчання.

Оскільки система орієнтована на роботу в режимі реального часу, важливим аспектом є забезпечення мінімальної затримки під час обробки даних. Інформаційне забезпечення в цьому випадку охоплює оптимізацію внутрішніх процесів, ефективне кешування повторюваних операцій, а також застосування алгоритмів із низькими обчислювальними витратами. Використання компактних моделей машинного навчання дозволяє гарантувати швидку обробку тексту навіть за умов значного навантаження або роботи на обмежених серверних ресурсах.

2.2 Вхідні та вихідні дані системи

Вхідною інформацією для системи є текстові повідомлення, що надходять від користувачів через інтерфейс бота. Текст може бути довільним: він не обмежується тематикою, стилем чи формою подання. Це створює додаткові вимоги до внутрішньої логіки системи, оскільки вона повинна бути здатною коректно працювати як із формальними новинними повідомленнями, так і з емоційними висловлюваннями,

заголовками або короткими фразами. Оскільки користувачі допускають граматичні помилки, використовують розмовну мову або сленг, система повинна бути стійкою до таких форм спотворення.

Перед потраплянням у модель вхідний текст проходить етапи очищення, включаючи нормалізацію регістру, видалення зайвих символів, усунення повторів пробілів та інших елементів, що не несуть змістового навантаження. Це дозволяє покращити якість представлення тексту у вигляді числових векторів та підвищити точність класифікації.

Вихідними даними є прогноз моделі щодо належності тексту до категорії «фейк» або «не фейк». Результат подається у вигляді оцінки ймовірності, що дозволяє сформувати не лише бінарну відповідь, але й рівень впевненості системи. У випадку невисокої впевненості модель може повідомити, що текст викликає сумніви, залишаючи користувачеві можливість самостійно оцінити достовірність.

Система також генерує службові вихідні дані, необхідні для адміністрування: лог-файли, помилки, службові звіти, параметри успішних чи невдалих проходжень навчання моделі. Ці дані не надсилаються користувачу, але використовуються адміністратором для підтримки роботи сервера та планування оновлень.

Вхідні та вихідні дані є основою функціонування інтелектуальної системи класифікації текстів і визначають логіку обробки інформації на всіх етапах її проходження. У межах даного проєкту вхідними даними виступають текстові повідомлення, що надходять від користувачів через інтерфейс бота, а також службові дані адміністратора, які забезпечують можливість доповнення навчальної вибірки та коригування моделі. Текстові повідомлення не мають жорстко визначеної структури, що робить їх типовими об’єктами обробки природної мови (NLP). Це означає, що система повинна бути здатною працювати з різноманітними формулюваннями, лінгвістичними конструкціями, пунктуаційними особливостями та індивідуальним стилем користувача, що додатково ускладнює процес аналізу [7].

Особливістю вхідних даних є їхня непередбачуваність: користувач може надсилати як короткі новинні твердження, так і повноцінні абзаци тексту, які містять складні описи подій. Через це важливим елементом інформаційного забезпечення

стає попередня обробка тексту, спрямована на очищення від зайвих символів, нормалізацію регістру та приведення фраз до уніфікованого формату. Така попередня підготовка дозволяє зменшити різноманітність вхідної інформації та підвищити точність роботи векторизатора і моделі класифікації.

Окремо варто розглянути службові дані, що надходять від адміністратора системи. Серед них найважливішими є тексти, які класифікуються вручну як «правдиві» або «фейкові» й використовуються для розширення навчальної вибірки. Ці дані мають особливу цінність, адже забезпечують адаптацію моделі до нових тенденцій у поширенні дезінформації. У цьому контексті адміністративні вхідні дані виконують роль механізму самонавчання: вони не лише впливають на роботу моделі, а й визначають темп її розвитку та здатність справлятися з новими викликами інформаційного простору.

Вихідні дані системи мають чітко регламентований формат, що забезпечує їх зручне сприйняття користувачами. Основним результатом роботи системи є класифікація тексту як «правдивого» або «фейкового», а також прогнозна ймовірність належності до фейкових новин. Цей індикатор дає змогу користувачу самостійно оцінити ступінь ризику та прийняти рішення щодо довіри до інформації. Формат відповіді повинен бути лаконічним, адже система інтегрована з ботом, що передбачає швидку подачу результатів без надмірної деталізації. Попри це, система генерує розширені службові відомості, включно з векторним поданням тексту, внутрішніми прогнозами моделі та допоміжними параметрами, але ці дані не відображаються кінцевому користувачу й використовуються лише в тестуванні та налагодженні.

Сама природа вихідних даних вимагає високої точності, адже вони впливають на сприйняття інформації користувачем та можуть формувати його ставлення до конкретного повідомлення. У зв’язку з цим модель класифікації має працювати стабільно та забезпечувати повторюваність результатів. Саме тому система використовує окремі файли моделі та векторизатора, що завантажуються під час старту сервера та не змінюються до моменту чергового перенавчання.

Одним із важливих аспектів формування вихідних даних є те, що система має працювати в умовах невизначеності, коли фейкові та правдиві новини можуть мати

схожі лексичні ознаки. Тому прогнозна ймовірність відіграє ключову роль, дозволяючи користувачу оцінити рівень упевненості моделі. Таким чином, вихідні дані виконують не лише інформативну функцію, а й функцію підтримки прийняття рішень.

Ngrok використовується як інструмент тунелювання, що дозволяє створювати захищене публічне URL-посилання для локального вебсервера. Це забезпечує можливість тестування та інтеграції бота під час розробки без необхідності розгортання інфраструктури на віддаленому сервері. Завдяки ngrok процес налагодження webhook-запитів значно спрощується: кожного разу, коли запускається локальний Flask-сервер, ngrok автоматично створює тимчасовий HTTPS-тунель, через який Bot API може надсилати повідомлення. Такий підхід прискорює розробку, дозволяє оперативно тестувати зміни та забезпечує гнучкість під час експериментів із функціональністю системи [14].

Загалом, підсистема вхідних і вихідних даних забезпечує основу для роботи класифікаційної моделі, обробки текстів, самонавчання системи та взаємодії з користувачем і адміністратором. Вона формує інформаційне середовище, у якому здійснюється аналіз текстів, здійснюється розширення навчальних даних та забезпечується зворотний зв’язок для користувача, що дозволяє системі залишатися ефективною, адаптивною та зрозумілою в практичному використанні.

2.3 Використані програмні інструменти, фреймворки та технології

Інформаційне забезпечення програмного комплексу значною мірою визначається тими інструментами та технологіями, що були використані під час його розробки. У межах створення системи автоматичного визначення фейкових новин було обрано низку фреймворків, бібліотек та інструментів, які забезпечують надійну, стабільну та масштабовану роботу системи. Їх добір здійснювався з урахуванням вимог до обробки текстів, ефективного виконання обчислень, взаємодії із зовнішнім інтерфейсом бота та забезпечення можливості подальшого тренування моделі.

Однією з ключових технологій, що визначає структуру серверної частини системи, є Python — високорівнева мова програмування, яка поєднує зрозумілий синтаксис із широкими можливостями для роботи з машинним навчанням. Вибір саме цього

інструменту є логічним, оскільки Python має одну з найпотужніших екосистем для роботи з текстами та побудови моделей класифікації. Гнучкість цієї мови дозволяє створювати модулі різного рівня складності — від попередньої обробки тексту і векторизації до запуску логістичної регресії та генерації прогнозів у режимі реального часу. Розвинена інфраструктура Python забезпечила можливість легко інтегрувати модель у веб-сервер та реалізувати ефективну взаємодію з ботом, що обробляє запити користувачів [1].

Центральним елементом серверної логіки є фреймворк Flask, який використовується як основний інструмент для побудови веб-сервера. Flask забезпечує простий механізм прийому HTTP-запитів та повернення відповідей у форматі JSON. Це дозволило реалізувати компактну, але продуктивну серверну частину, яка приймає текстові дані, передає їх у модуль обробки, формує результати класифікації та повертає їх ботові. Однією з особливостей Flask є те, що він не накладає надмірних обмежень на структуру проєкту, що дало можливість розробити гнучку та легку архітектуру, адаптовану під задачі машинного навчання, без потреби у складних конфігураціях або багатоетапних налаштуваннях [2].

Для роботи із зовнішнім інтерфейсом комунікації використано API бота. Він виступає єдиною точкою взаємодії між системою та користувачем, забезпечуючи передачу текстових повідомлень у сервер та отримання відповідей. Сам бот не містить власної логіки аналізу, а лише виконує функцію посередника, що робить його легким та швидким у роботі. Такий підхід дозволяє розвантажити користувацьку частину та зосередити всі обчислення на сервері, який володіє доступом до моделі та навчальних даних. Бот виступає логічною складовою інформаційного забезпечення, адже він визначає формат, обсяг і спосіб надходження вхідних даних, а також те, як користувач сприймає результати класифікації.

Важливою технологією, що забезпечує інтелектуальну складову системи, є бібліотека scikit-learn. Саме вона була використана для побудови класифікаційної моделі, яка визначає приналежність тексту до категорії «фейковий» або «правдивий» [11]. Scikit-learn має широкий набір алгоритмів, проте для даної задачі було обрано логістичну регресію як оптимальний баланс між простотою, швидкістю обчислень та

здатністю інтерпретувати результати. Однією з ключових переваг scikit-learn є можливість зберігати модель у вигляді окремого файлу, що значно спрощує її подальше використання: сервер може завантажити модель при старті та виконувати класифікацію без необхідності повторного тренування [3].

Серед інструментів, що забезпечують підготовку текстів до класифікації, важливе місце займає модуль векторизації, реалізований засобами TfidfVectorizer. Цей інструмент виконує перетворення текстів у числові вектори, що є основою для навчання моделі машинного навчання. Векторизація враховує частотність слів, їх значущість у конкретному тексті та у всій колекції документів, що дозволяє виділити найбільш інформативні ознаки. Використання TF-IDF у межах цієї системи є цілком обґрунтованим, оскільки воно формує компактне, але достатньо інформативне представлення тексту, яке не потребує великих обчислювальних ресурсів і легко масштабується при розширенні навчальної вибірки [8].

Серед інших інструментів, що забезпечують коректну роботу системи, слід виділити бібліотеки Pandas та NumPy, які використовуються для обробки табличних даних і виконання низки проміжних операцій. Pandas відіграє центральну роль у роботі з навчальним датасетом, дозволяючи виконувати фільтрацію, структурування, об’єднання та збереження даних у зручному форматі. NumPy, у свою чергу, забезпечує ефективні операції з масивами, що є критично важливим у процесі векторизації та виконання математичних обчислень під час прогнозування [6].

Для серіалізації та збереження моделі використовується бібліотека Joblib. Зберігання моделі та векторизатора у вигляді файлових об’єктів дозволяє уникнути повторних розрахунків і значно прискорює процес обробки текстів. Це особливо важливо, оскільки система працює у режимі реального часу, і кожен додатковий мілісекундний затримка може вплинути на сприйняття швидкодії користувачем. Завдяки Joblib створено стабільний механізм перенесення моделей між сесіями серверної роботи, що забезпечує високу надійність та відтворюваність результатів [18].

Окрему роль у системі відіграє можливість роботи із зовнішніми запитами, яка реалізована за допомогою бібліотеки Requests. Вона використовується для взаємодії

бота та Flask-сервера, забезпечуючи передачу даних у стандартизованому форматі. Використання Requests дозволило побудувати зрозумілий та надійний канал комунікації, який забезпечує правильність обміну даними та стабільність роботи навіть у випадку зростання кількості користувачів [12].

Ще однією технологією є модуль регулярних виразів, що використовується під час попередньої обробки текстів. Регулярні вирази дозволяють видаляти зайві символи, нормалізувати написання, знаходити та коригувати фрагменти, що можуть негативно вплинути на результат обробки. Саме цей інструмент забезпечує початковий рівень «очищення» тексту, яке є необхідним для формування коректних векторів ознак.

Сукупність обраних фреймворків та бібліотек формує цілісну інфраструктуру, у межах якої система здатна ефективно обробляти тексти, прогнозувати їх приналежність до певної категорії та забезпечувати безперебійну взаємодію з користувачем через інтерфейс бота. Кожний інструмент виконує свою специфічну роль, а їх взаємодія створює збалансовану архітектуру, яка може легко масштабуватися, адаптуватися та розширюватися. Така побудова інформаційного забезпечення гарантує надійність системи, її стійкість до збоїв та здатність працювати в умовах постійної зміни інформаційного середовища.

2.4 Джерела та структура текстових даних

Датасет, на якому ґрунтується робота системи, складається з текстових новин та коротких повідомлень, що поділені на категорії «фейк» та «правда». Джерела даних включають як реальні інформаційні повідомлення, так і штучно створені приклади, що відображають найтиповіші патерни дезінформації. Структура датасету передбачає наявність двох стовпців — тексту та мітки класу. Важливою характеристикою є збалансованість даних, оскільки нерівномірне співвідношення класів може спричинити зміщення моделі та зниження її здатності коректно класифікувати рідкісні патерни фейків.

Під час формування датасету враховано фактори різноманітності: повідомлення охоплюють політичні, соціальні, економічні, наукові, медичні та побутові теми. Це дозволяє моделі краще узагальнювати інформацію та робити точніші прогнози в

реальних умовах, коли система стикається з повідомленнями, що не були присутні в навчальній вибірці [17].

У межах розробленої системи структуру текстових даних слід розглядати як центральний елемент інформаційного забезпечення, що безпосередньо впливає на якість роботи класифікаційної моделі. Всі текстові записи, які використовуються для навчання та тестування моделі, містять короткі новинні твердження або інформаційні повідомлення, що відображають різні типи контенту: політичні новини, суспільні події, наукові твердження, побутові фрази, вигадані твердження та елементи відвертої дезінформації. Це поєднання різних тематичних напрямів створює збалансоване навчальне середовище, що сприяє підвищенню здатності моделі розрізняти правдиву та неправдиву інформацію незалежно від змісту чи стилістичних особливостей тексту [10].

Джерелами формування бази даних виступають як власноруч відібрані тексти, так і спеціально згенеровані приклади, які моделюють структуру справжніх новин або типових фейкових повідомлень. Завдяки такому підходу забезпечується різноманітність навчального набору, що є важливим чинником у задачах класифікації текстів. З одного боку, правдиві приклади включають твердження зі сфери політики, економіки, освіти та соціального життя, що відтворюють характерні особливості інформаційних матеріалів. З іншого боку, фейкові тексти містять неправдоподібні або вигадані твердження, характерні для маніпулятивного або відверто сенсаційного контенту. Такий контраст дозволяє моделі формувати чіткі кордони між класами та краще розпізнавати характерні мовні маркери дезінформації.

Важливим аспектом структури текстових даних є їхня уніфікація. Незважаючи на основну схожість формату — коротке текстове повідомлення — усі дані мають бути приведені до єдиного стилю подання. Це забезпечує коректність роботи векторизатора та підвищує якість ознак, що подаються на вхід моделі. Попередня обробка включає нормалізацію регістру, видалення зайвих символів, а також очищення від повторюваних фрагментів. Такий підхід дозволяє суттєво зменшити шум у даних, який часто негативно впливає на результати моделювання.

Окрім навчальної вибірки, до структури текстових даних входять і ті повідомлення, які надходять від користувачів під час реальної роботи системи. Це створює потенціал динамічного розширення датасету, що дозволяє системі залишатися актуальною незалежно від змін у інформаційному середовищі. Адміністратор може вручну визначити клас, до якого належить отриманий текст, і додати його до навчального набору. Завдяки цьому формуються умови для поступового масштабування бази даних та покращення якості моделі без необхідності глобальних змін у її структурі.

Таким чином, джерела та структура текстових даних у системі формують основу для роботи моделі машинного навчання, впливаючи на її здатність розпізнавати закономірності, робити точні прогнози та адаптуватися до нових умов. Ретельно сформована та постійно підтримувана текстова база виступає ключовим компонентом, що забезпечує ефективність та достовірність роботи всієї системи.

2.5 Інформаційні процеси, обробки та взаємодії компонентів системи

Робота системи визначення фейкових новин ґрунтується на комплексі внутрішніх інформаційних процесів, які забезпечують повний цикл обробки тексту — від моменту його надходження до формування кінцевого висновку. Ці процеси об’єднують у собі механізми попередньої обробки даних, векторизацію тексту, застосування моделі машинного навчання, генерацію результатів та їх передачу користувачеві через інтерфейс бота. Система побудована таким чином, щоб забезпечити мінімальну затримку, високу точність та стабільність роботи навіть за умов значного навантаження.

Першим етапом є отримання повідомлення через інтерфейс бота, який виконує роль шлюзу між користувачем і сервером. Текст передається до серверної частини у вигляді HTTP-запиту, що містить лише необхідні параметри: ідентифікатор користувача та сам текст повідомлення. Такий підхід дозволяє забезпечити компактність переданих даних та уникнути надмірної передачі структурованої інформації, що підвищує ефективність взаємодії. На цьому етапі система виконує перевірку довжини тексту, кількості слів та коректності формування запиту, що є важливим для уникнення помилкових класифікацій.

Після надходження тексту система переходить до процесу попередньої обробки. Це ключова частина інформаційної логіки, що дозволяє мінімізувати невідповідності в даних. Попередня обробка включає очистку від зайвих символів, зведення тексту до нижнього регістру, видалення або заміну спеціальних знаків, нормалізацію пробілів та структурування тексту. Хоча ці операції видаються простими, вони виконують критичну роль у формуванні якісних ознак для моделі машинного навчання. Будь-яка невідповідність на цьому рівні може негативно вплинути на підсумковий прогноз, тому система приділяє особливу увагу уніфікації вхідних даних.

Після очищення текст потрапляє у модуль векторизації, який переводить його у числове подання. Векторизація є одним із найважливіших етапів, оскільки саме від неї залежить, які ознаки машинне навчання зможе розпізнати в тексті. В основі роботи векторизатора лежить TF-IDF-представлення, яке дозволяє виділити інформативні слова, зменшити вплив загальних термінів і сформувати числові вектори, що містять суттєві ознаки текстового повідомлення. Такий підхід забезпечує оптимальний баланс між обсягом інформації та швидкістю її подальшої обробки, що особливо важливо в умовах роботи у режимі реального часу [19].

На наступному етапі отримані вектори передаються до моделі машинного навчання. В основі системи використана логістична регресія — алгоритм, який добре зарекомендував себе у задачах бінарної класифікації. Модель аналізує ознаки тексту, визначає його схожість із відомими прикладами у навчальній вибірці та формує прогноз у вигляді ймовірності належності до класу «фейк». Отриманий результат є не лише індикатором кінцевої класифікації, а й містить інформацію про ступінь упевненості моделі у своєму висновку. Завдяки цьому система може надавати користувачеві як однозначний висновок, так і додаткове числове значення, що допомагає оцінити ризик недостовірності інформації.

Після формування прогнозу система переходить до етапу генерації відповіді для користувача. Тут важливо забезпечити доступність та зрозумілість результату, оскільки користувач очікує швидкої та лаконічної відповіді. Система формує текстове повідомлення, яке містить статус повідомлення та рівень імовірності. Для уникнення непорозумінь вихідні дані подаються у стандартизованому форматі, що є однаковим

для всіх користувачів. Це підвищує довіру й забезпечує однакове трактування результатів незалежно від суб’єктивного сприйняття.

Одним із найважливіших процесів системи є динамічне оновлення навчальних даних. Розроблений механізм дозволяє адміністратору додавати нові тексти, визначати їхній клас і запускати повторне навчання моделі. Це створює основу для самонавчання системи, оскільки нові приклади фейкових або правдивих текстів дозволяють моделі адаптуватися до сучасних тенденцій дезінформації. Процес повторного тренування реалізований таким чином, що не перериває роботу системи та не впливає на взаємодію з користувачами. Оновлені файли моделі і векторизатора завантажуються автоматично після завершення тренування, що забезпечує стабільність і безперервність роботи.

Не менш важливою є загальна архітектура інформаційних потоків. Система побудована на принципі чіткої взаємодії між сервером, моделлю та інтерфейсом бота. Потоки даних рухаються в одному напрямку під час класифікації текстів і в зворотному під час оновлення навчальної вибірки. Така організація дозволяє уникнути надмірних затримок і забезпечує мінімальну залежність між компонентами. Кожен елемент системи відповідає за конкретний етап обробки, а їх узгоджена робота формує цілісний інформаційний процес.

Таким чином, інформаційні процеси, логіка функціонування та взаємодія компонентів системи становлять єдину структуру, яка забезпечує ефективну, стабільну та адаптивну роботу програмного комплексу. Чітка організація потоків даних, підготовка текстів, ефективна векторизація та використання моделі машинного навчання створюють надійну основу для точного визначення фейкової інформації та оперативного інформування користувачів.

Висновки до розділу

У межах другого розділу було розглянуто структуру інформаційного забезпечення системи, визначено особливості організації даних, описано використовувані інструменти та методи обробки текстової інформації. Проведений аналіз дозволив встановити, що ефективність роботи системи безпосередньо залежить від узгодженої дії всіх програмних компонентів — від попередньої обробки текстів до побудови

прогнозу і передачі результату користувачеві. Використання сучасних бібліотек та фреймворків створює надійну основу для швидкої роботи моделі, тоді як можливість оновлення навчальних даних забезпечує адаптацію до змін у інформаційному середовищі. Сформована структура інформаційних потоків гарантує стабільність системи, її масштабованість та здатність функціонувати в режимі реального часу, що є ключовими вимогами для програмних рішень у сфері протидії дезінформації

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Теоретичні основи класифікації текстової інформації

Класифікація текстової інформації є одним із ключових напрямів сучасного машинного навчання та обробки природної мови. У найширшому розумінні класифікація передбачає віднесення певного текстового об’єкта до однієї з наперед визначених категорій на основі його змістових чи статистичних характеристик. У контексті даного проєкту категорії зводяться до двох класів — «правдиві новини» та «фейкові новини», що формує задачу бінарної класифікації. Теоретичні основи цього процесу поєднують у собі елементи лінгвістичного аналізу, математичного моделювання та алгоритмічних підходів, що забезпечують можливість формалізувати природну мову та перетворити її у форму, придатну для машинної інтерпретації [9].

Особливістю текстової інформації є те, що вона не має чіткої структури, а її зміст формується з набору слів, контекстуальних зв’язків, семантичних значень і стилістичних особливостей. Тому першим кроком у математичному трактуванні текстів є їх перетворення у формальний об’єкт — набір ознак, які відображають внутрішній зміст та структуру документа. Такий процес називають репрезентацією або векторизацією. Репрезентація є критично важливою, оскільки саме вона визначає якість подальших прогнозів: від того, наскільки вдало текст перетворено на числову форму, залежить здатність моделі розпізнавати закономірності та виконувати точну класифікацію.

У загальному вигляді класифікація текстів спирається на припущення, що тексти, які належать до одного класу, мають спільні статистичні або структурні характеристики. Наприклад, фейкові новини часто містять певні типові ознаки: перебільшення, емоційно забарвлену лексику, неправдоподібні твердження або маніпулятивні формулювання. Натомість правдиві новини зазвичай характеризуються нейтральним стилем, точнішими фактами та збалансованою подачею інформації. Математичні моделі виявляють подібні закономірності шляхом аналізу частотності окремих термінів та їх співвідношень у різних текстах.

Побудова класифікаційної моделі передбачає наявність навчальної вибірки, яка містить приклади текстів із відповідними мітками класів. Навчання моделі полягає у пошуку співвідношень між ознаками тексту та його класом. Формально цей процес можна трактувати як знаходження функції, що зіставляє кожному вектору ознак певну ймовірність належності до конкретного класу. У випадку бінарної класифікації обчислюється значення між 0 і 1, яке відображає впевненість моделі в тому, що текст є фейковим. Такий підхід забезпечує не лише класифікацію, а й відносну оцінку ризику, що є важливим для задач, де повне виключення помилок є недосяжним.

Сутність класифікації текстових даних також полягає у тому, що моделі мають працювати в умовах невизначеності. Тексти можуть бути неоднозначними, містити жартівливі елементи, іронію або містити інформацію, яку складно перевірити. Математичні алгоритми не мають доступу до зовнішнього контексту та аналізують лише сам текст, спираючись на статистичні залежності. Це створює специфічну природу задачі, де точність прогнозу безпосередньо залежить від обсягу та якості навчальних даних.

Однією з ключових теоретичних засад є ідея лінійної відокремлюваності класів у просторі ознак. Якщо певним чином перетворити текст у багатовимірний простір, то тексти одного класу повинні розташовуватися переважно в одному регіоні цього простору. Тоді модель може провести між цими областями поверхню розділення, яка дозволить класифікувати нові тексти, що не входили у початковий набір навчання. Для задачі визначення фейкових новин цей підхід працює особливо добре, оскільки фейкові та правдиві тексти часто дійсно мають помітні лінгвістичні відмінності.

Теоретична складність задачі також пов’язана з тим, що текстові дані мають високу розмірність. Навіть короткі фрагменти можуть містити десятки унікальних слів, що у масштабах великої вибірки перетворюється на тисячі ознак. Одна з фундаментальних проблем машинного навчання — так звана «прокляття розмірності», — виникає тоді, коли число ознак суттєво перевищує кількість прикладів для навчання. Тому математичні моделі мають вміти працювати у таких умовах, уникаючи перенавчання та зберігаючи узагальнюючу здатність.

У контексті математичного забезпечення важливу роль відіграє і питання нормалізації. Незважаючи на те, що текст складається зі слів, кожне з цих слів може мати різні форми, бути написаним у різному стилі або містити варіації. Математичні алгоритми працюють із числовими значеннями, тому всі текстові елементи мають бути приведені до єдиного формату, що дозволяє зменшити розкид значень та уникнути хибних кореляцій. Саме цей процес лежить в основі коректного формування простору ознак.

Таким чином, теоретичні основи класифікації текстів охоплюють широкий спектр концепцій — від формальних математичних моделей та статистичних закономірностей до алгоритмічних підходів, спрямованих на аналіз природної мови. Застосування цих принципів у системі визначення фейкових новин забезпечує основу для побудови точних прогнозів, що дозволяють автоматично й надійно визначати недостовірну інформацію в умовах динамічного інформаційного середовища.

3.2 Математичні моделі подання тексту

Математичне подання текстових даних є фундаментальним етапом у побудові системи машинного навчання, оскільки саме від вибору способу репрезентації залежить якість подальшої класифікації. Текст природною мовою не має чіткої формальної структури, а тому повинен бути перетворений у числову форму, придатну для математичного аналізу. У межах створеної системи визначення фейкових новин застосовується модель TF-IDF, яка дозволяє відобразити текст як вектор ознак, що враховує частотність термінів та їхню інформативність у межах колекції документів.

Класична модель «мішка слів» лежить в основі багатьох методів подання тексту, оскільки дозволяє представити документ як множину унікальних термінів без урахування синтаксичних залежностей чи порядку слів. Проте сама по собі частотність не завжди коректно відображає значущість терміну: надто поширені слова втрачають аналітичну цінність, тоді як рідкісні терміни можуть містити ключову інформацію, яка визначає зміст документа. Саме тому використання розширеної моделі TF-IDF забезпечує більш збалансоване подання текстових даних.

Сутність TF-IDF полягає у визначенні ваги слова як добутку частотності в документі та оберненої частотності появи цього слова у всій вибірці. Такий підхід

дозволяє приглушувати вплив загальних термінів і водночас виділяти значущі слова, характерні для певних типів повідомлень. У результаті кожен документ перетворюється на багатовимірний вектор, де кожне значення відповідає вагомості певного терміну. Приклад візуалізації принципу TF-IDF (Рисунок 3.1), який демонструє різницю між термінами за значенням їхньої інформативності.

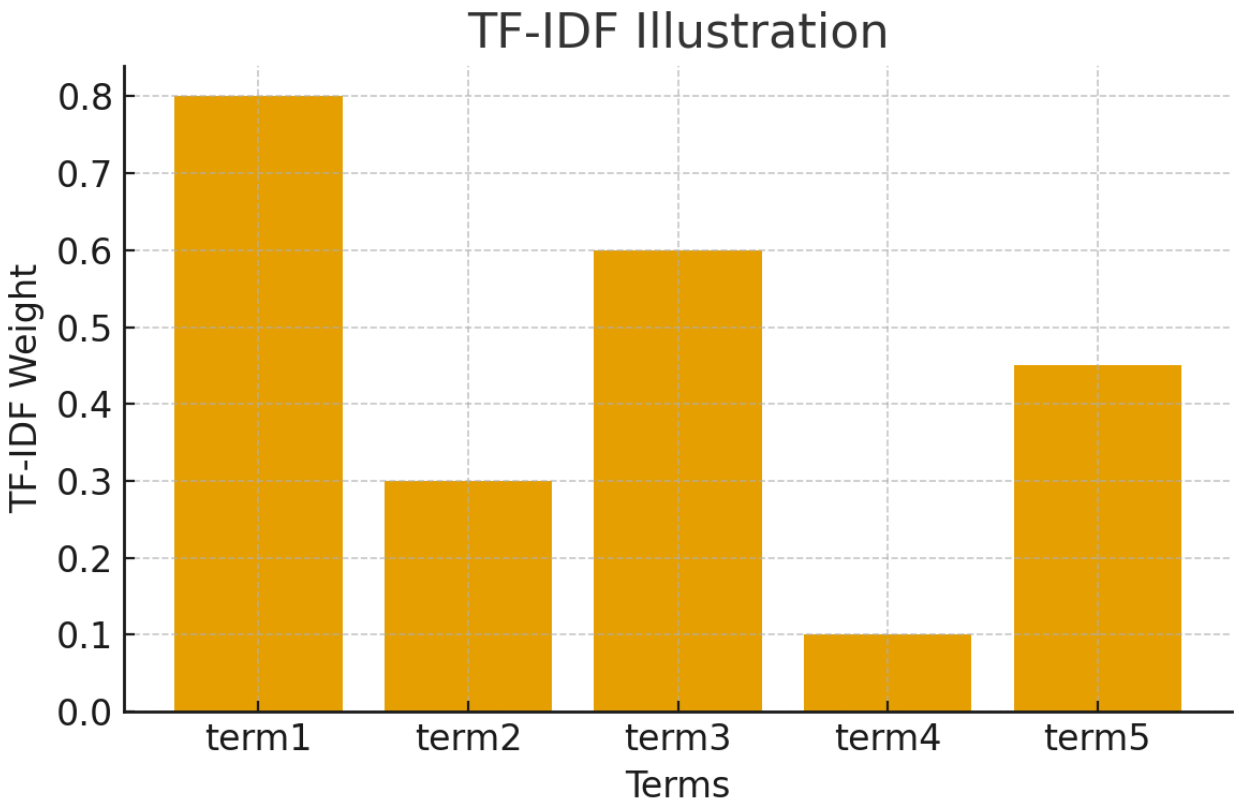

Рисунок 3.1 – Візуалізація ваг термінів у моделі TF-IDF

Цей підхід є зручним у застосуванні до задачі визначення фейкових новин, оскільки дозволяє моделі логістичної регресії ефективно відокремлювати тексти за мовними індикаторами, властивими для певного класу повідомлень. Використання TF-IDF забезпечує стабільність обчислень, інтерпретованість результатів та достатньо низьку обчислювальну складність, що особливо важливо для систем, які функціонують у режимі реального часу та регулярно доповнюють навчальний датасет новими текстами.

Таким чином, модель TF-IDF формує надійну математичну основу для подання текстових даних, забезпечуючи можливість точного перетворення складної природної мови у структуру, доступну для аналізу алгоритмами машинного навчання.

3.3 Побудова простору ознак для класифікації текстових повідомлень

Побудова простору ознак є ключовим етапом у створенні математичної моделі класифікації текстів, оскільки саме на рівні ознак формується структурне уявлення про документи. У межах системи визначення фейкових новин простір ознак ґрунтується на TF-IDF-векторизації, яка дозволяє представити текст у числовій формі та зробити його придатним для подальнього аналізу логістичною регресією. Перед побудовою векторів виконується попередня обробка: очищення тексту від сторонніх символів, перетворення до нижнього регістру та нормалізація пробілів. Це забезпечує однорідність корпусу та коректне формування зважених ознак.

Структура простору ознак для текстових даних є високовимірною, оскільки кількість ознак відповідає розміру словника, сформованого з навчальної вибірки. Такий підхід дозволяє враховувати широкий спектр мовних елементів, однак потребує контролю за надлишковістю. Для цього використовуються параметри min_df та max_df, які відсікають рідкісні та надто поширені терміни. У результаті простір ознак формується з найбільш інформативних слів, що відображають зміст документів та забезпечують модель можливістю відокремлювати фейкові тексти від правдивих.

Кожен документ у сформованому просторі подається у вигляді вектора ваг, що дає змогу логістичній регресії шукати гіперплощину, яка розділяє два класи. Якість класифікації значною мірою залежить від того, наскільки збалансованим є простір ознак: надто велика його розмірність може призводити до перенавчання, тоді як недостатня — до втрати важливої інформації. У даній системі використання TF-IDF забезпечує оптимальний баланс між деталізацією та узагальненістю, що дозволяє моделі працювати стабільно та ефективно.

Таким чином, побудова простору ознак виступає фундаментом математичної частини системи. Саме завдяки TF-IDF та попередній обробці текстів досягається якісне формування векторного представлення документів, на основі якого модель машинного навчання здійснює класифікацію новинних повідомлень.

3.4 Математичний апарат логістичної регресії

Логістична регресія є базовою математичною моделлю, яка широко застосовується у задачах бінарної класифікації, включно з визначенням фейкових новин. Її популярність зумовлена поєднанням простоти, інтерпретованості та здатності отримувати ймовірнісні оцінки для кожного класу. На відміну від лінійної регресії, що прогнозує числові значення, логістична регресія моделює імовірність належності об'єкта до певного класу, що робить її придатною для задач, де результат має дискретний характер.

Основою моделі є лінійна комбінація ознак, зважених коефіцієнтами. Для текстової класифікації такими ознаками виступають ваги термінів, сформовані векторизатором TF-IDF. Якщо позначити вектор ознак як x, а вектор параметрів моделі як w, то лінійний вираз w·x визначає ступінь приналежності документа до класу. Проте сам по собі цей вираз не може бути інтерпретований як ймовірність, тому до нього застосовується логістична (сигмоїдна) функція (Рисунок 3.2).

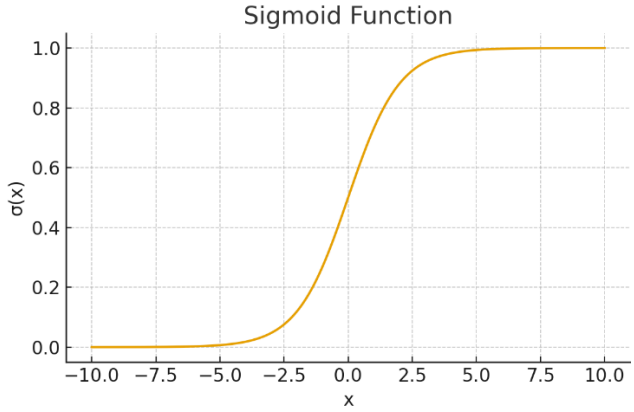

Рисунок 3.2 – Сигмоїдна функція логістичної регресії

Сигмоїдна функція:

$$\sigma(z) = \frac{1}{1+e^{-z}}, \tag{3.1}$$

де z — вхідне значення моделі (лінійна комбінація ознак після застосування векторизатора), e — основа натурального логарифма. Перетворює довільне числове значення на інтервал від 0 до 1, завдяки чому модель отримує можливість інтерпретувати результати як ймовірність належності документа до класу «фейк». Якщо значення σ(z) перевищує визначений поріг (зазвичай 0.5), текст класифікується як фейковий, інакше — як правдивий. У нашій системі додатково може регулюватися

порогове значення, що дозволяє оптимізувати роботу моделі залежно від структури датасету.

Процес навчання логістичної регресії полягає у знаходженні оптимальних параметрів w, які мінімізують функцію втрат. Для цієї моделі використовується log-loss (бінарна кросентропія), яка штрафує впевнені, але неправильні прогнози сильніше, ніж невпевнені. Оптимізація параметрів виконується методом градієнтного спуску або його модифікаціями, які забезпечують поступове покращення моделі шляхом руху у напрямку, протилежному градієнту функції втрат.

У задачі визначення фейкових новин логістична регресія має низку переваг. Вона є чутливою до значущих ознак, що дозволяє виокремлювати характерні патерни мови у текстах. Крім того, модель швидко навчається, здатна працювати з великими розмірностями простору ознак та не вимагає надмірних обчислювальних ресурсів. Її пояснюваність дозволяє аналізувати внесок окремих слів у фінальний результат, що є важливим для систем протидії дезінформації.

Завдяки поєднанню математичної строгості, ймовірнісного характеру передбачень та ефективності при роботі з текстовими векторами TF-IDF логістична регресія виступає оптимальним вибором для реалізації класифікаційного компонента у даному проєкті. Вона забезпечує баланс між точністю моделі та швидкодією, що робить її придатною для використання у режимі реального часу.

3.5 Балансування класів та нормалізація ваг у задачі класифікації

Однією з ключових математичних проблем у класифікації текстових даних є дисбаланс класів, який виникає тоді, коли кількість прикладів однієї категорії суттєво переважає кількість прикладів іншої. У задачі визначення фейкових новин така ситуація зустрічається доволі часто, адже у реальних умовах правдивих повідомлень зазвичай значно більше, ніж фейкових. Це призводить до того, що модель машинного навчання, орієнтуючись на мінімізацію помилок, може схилятися до передбачення саме домінуючого класу. У результаті система демонструє високу «формальну» точність, але втрачає здатність виявляти менш представлений клас, тобто фейкові новини.

Для ілюстрації цього ефекту часто використовують матрицю неточностей, яка показує розподіл передбачень моделі по чотирьох категоріях: істинні позитиви, істинні негативи, хибні позитиви та хибні негативи (Рисунок 3.3). Якщо більшість передбачень припадає на домінуючий клас, то модель демонструє тенденцію ігнорувати менш поширену категорію, що є небажаним результатом для задачі виявлення фейкових повідомлень.

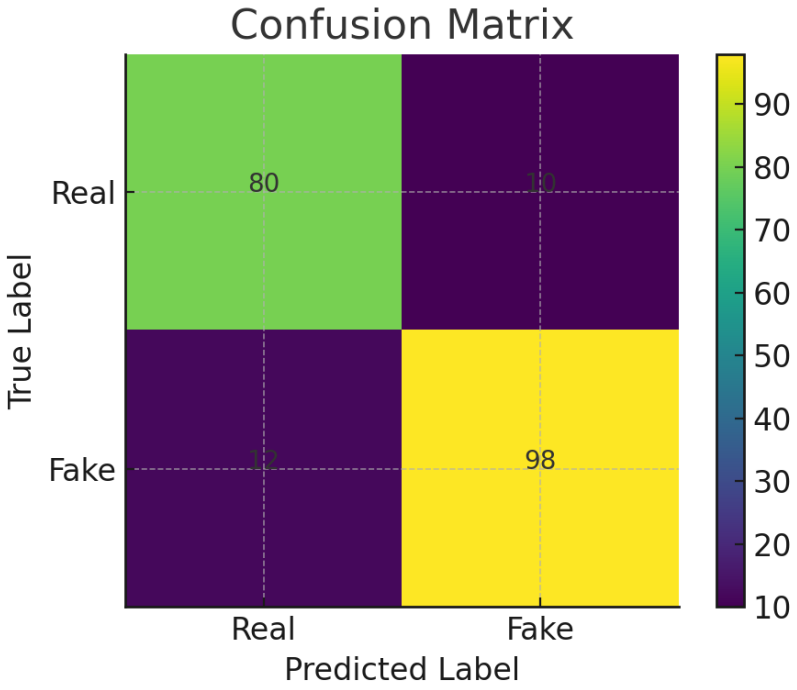

Рисунок 3.3 – Матриця неточностей класифікаційної моделі

Щоб запобігти подібному ефекту, у логістичній регресії застосовується механізм нормалізації ваг класів. Він полягає у зміні внеску кожного класу в загальну функцію втрат: рідкісні класи отримують більшу вагу, що стимулює модель приділяти їм більше уваги. Наприклад, якщо кількість фейкових новин у навчальному датасеті значно менша, ніж правдивих, то при встановленні параметра class_weight="balanced" відбувається автоматичне збільшення їхнього впливу на процес навчання. Таким чином, модель прагне мінімізувати не лише загальну кількість помилок, але й зменшити кількість пропущених фейкових повідомлень.

Балансування класів має критично важливе значення у контексті систем протидії дезінформації, оскільки хибні негативи (коли фейк класифікується як правда) є більш небезпечними, ніж хибні позитиви. Якщо система не здатна розпізнати фейкову

інформацію, це ставить під загрозу ефективність її використання. Саме тому процес вибору ваг класів є не лише математичною операцією, але й концептуальним рішенням, спрямованим на підвищення надійності моделі в умовах реального інформаційного середовища.

Регуляризація також відіграє важливу роль у стабілізації моделі під час роботи з дисбалансом. Вона запобігає надмірному збільшенню ваг окремих ознак, які можуть випадково збігатися із прикладами менш представленого класу. Таким чином, модель уникає перенавчання та зберігає здатність узагальнювати дані.

Комплексне застосування нормалізації ваг, регуляризації та контрольованої векторизації тексту формує надійну теоретичну основу для роботи моделі у нерівномірних умовах. Завдяки цьому система зберігає баланс між чутливістю до фейкових повідомлень та стабільністю прогнозів, що є необхідним для практичного використання у сфері інформаційної безпеки.

3.6 Оцінювання якості математичної моделі класифікації

Оцінювання якості моделі є одним із найбільш важливих етапів побудови класифікаційної системи, оскільки дає можливість визначити, наскільки ефективно алгоритм виконує поставлене завдання. У межах даного проєкту аналіз якості моделі класифікації текстових повідомлень дозволяє оцінити її здатність коректно розмежовувати фейкові та правдиві новини, визначати стійкість до дисбалансу класів і перевіряти загальну узгодженість прогнозів.

Першим та базовим індикатором виступає точність, що визначає відношення правильно класифікованих прикладів до загальної кількості елементів. Проте, як показує практика у задачах зі зміщеним розподілом класів, висока точність не завжди відображає реальну ефективність моделі. Саме тому більш показовими є такі метрики, як precision та recall, які дозволяють визначити якість роботи моделі для кожного окремого класу. Precision демонструє частку коректно визначених фейкових повідомлень серед усіх передбачених як фейк, тоді як recall показує, яку частку реальних фейкових новин модель змогла виявити. Компроміс між цими двома метриками відображається у F1-мірі, яка є їх гармонійним середнім.

Однак навіть цих індикаторів може бути недостатньо для глибшого розуміння поведінки моделі, особливо у випадках, коли важливо оцінити її роботу при зміні порогу класифікації. З цією метою використовується ROC-крива, яка графічно відображає залежність між частотою хибнопозитивних спрацьовувань (FPR) та частотою істинних позитивів (TPR). На рисунку 3.4 наведено приклад такої кривої, що дозволяє оцінити загальну дискримінативну здатність моделі. Чим ближче ROC-крива наближається до верхнього лівого кута графіка, тим вищою є якість класифікації, оскільки модель демонструє здатність правильно розрізняти обидва класи навіть при варіації порогу.

Показником, який узагальнює інформацію ROC-кривої, є площа під нею — AUC. Значення AUC, близьке до 1, свідчить про високу дискримінаційну здатність моделі, тоді як значення, близьке до 0.5, означає випадковий характер прогнозів. В умовах задачі визначення фейкових новин високий показник AUC є важливим, оскільки забезпечує можливість адаптації порогу чутливості моделі залежно від рівня ризику поширення дезінформації.

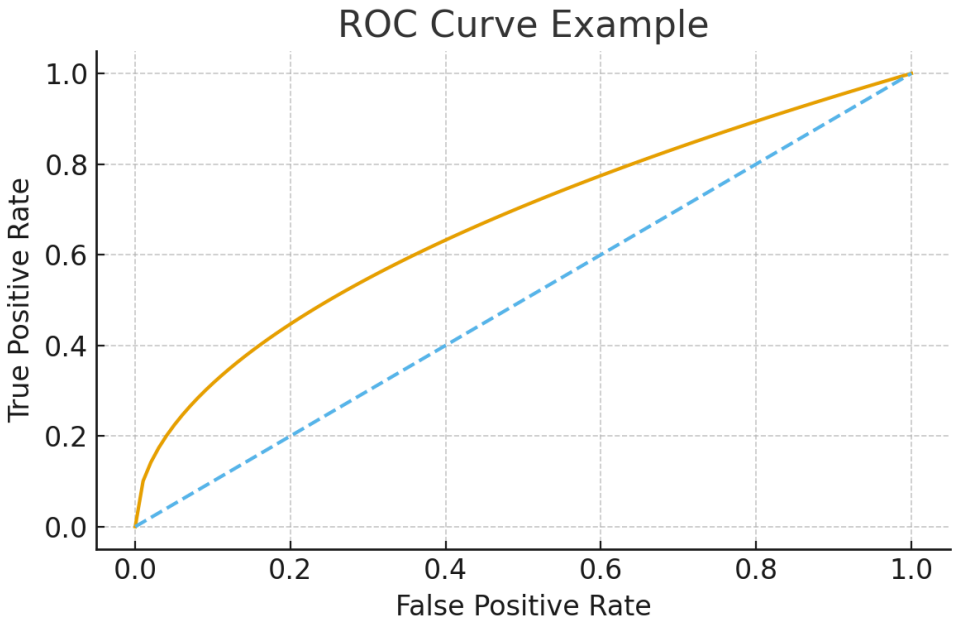

Рисунок 3.4 – ROC-крива класифікаційної моделі

Комбіноване використання ROC-кривої, матриці неточностей, precision, recall та F1-міри дозволяє отримати всебічну характеристику роботи моделі та забезпечити її

подальше вдосконалення. Такий підхід робить оцінювання не лише технічним, а й концептуальним інструментом, спрямованим на покращення ефективності системи виявлення фейкових новин.

3.7 Математична інтерпретація навчання та перенавчання моделі

Процес навчання моделі логістичної регресії у задачі класифікації текстових повідомлень базується на поступовому коригуванні параметрів, що визначають роздільну гіперплощину між класами. У рамках навчання датасет поділяється на тренувальну та тестову вибірки, що дає можливість одночасно здійснювати оптимізацію параметрів і контролювати узагальнювальну здатність моделі. Застосування стратифікації під час поділу вибірки гарантує, що пропорція класів зберігається в обох підмножинах, що є критично важливим у контексті дисбалансу даних.

Навчання моделі ґрунтується на мінімізації бінарної кросентропії, яка виступає функцією втрат. Метод градієнтного спуску або його модифікації поступово визначають напрямок зміни параметрів, що призводить до зменшення помилок класифікації. У процесі навчання ваги моделі адаптуються до структури простору ознак, що дозволяє алгоритму ефективно розрізняти фейкові та правдиві новини.

Перенавчання моделі здійснюється щоразу, коли до датасету додаються нові приклади, надіслані адміністратором через інтегрований інтерфейс системи. В такому випадку модель проходить повний цикл навчання на оновленому наборі даних, що дозволяє їй адаптуватися до нових лінгвістичних патернів та актуальних форм дезінформації. Такий підхід забезпечує динамічне оновлення математичної основи системи, дозволяючи моделі підтримувати високу ефективність в умовах змінного інформаційного середовища.

Висновки до розділу

У даному розділі було сформовано математичну основу системи класифікації фейкових новин, яка включає побудову простору ознак, формалізацію тексту за допомогою моделі TF-IDF та застосування логістичної регресії як базового алгоритму машинного навчання. Розгляд булих аспектів — від векторизації та формування ознак до балансування класів та оцінювання якості — дозволив показати, що вибраний

підхід забезпечує оптимальне співвідношення точності, інтерпретованості та обчислювальної ефективності.

Система ґрунтується на надійних математичних методах, здатних адаптуватися до нових даних і реагувати на зміну патернів у текстовій інформації, що є критично важливим для задачі виявлення дезінформації. Комплексний розгляд математичних компонентів підтверджує, що модель є достатньо гнучкою, стійкою та придатною для використання у реальних умовах, забезпечуючи високий рівень якості класифікації та можливість подальшого розширення функціональності.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Архітектура та загальна структура програмної системи

Архітектура розробленої системи визначення фейкових новин побудована на модульному принципі та включає взаємодію кількох незалежних компонентів, кожен з яких виконує чітко визначену функцію. Такий підхід забезпечує гнучкість, зрозумілу структуру, простоту супроводу та можливість подальшого масштабування програмного комплексу. Система складається з чотирьох основних частин: користувацького інтерфейсу у вигляді бота, серверного застосунку, модуля машинного навчання та підсистеми керування даними. Взаємодія між цими частинами відбувається через HTTP-запити, що забезпечує стандартизований механізм комунікації та мінімізує залежність компонентів один від одного.

Логіка роботи системи починається з отримання текстового повідомлення від користувача через бот-інтерфейс. Бот надсилає це повідомлення на серверний застосунок через webhook-запит, де відбувається первинна обробка, перевірка коректності та нормалізація введеного тексту. Далі сервер передає очищений текст до модуля класифікації, який використовує завантажені артефакти моделі машинного навчання — TF-IDF векторизатор та логістичну регресію. Після перетворення тексту у вектор та обчислення ймовірності належності повідомлення до класу «фейк» сервер формує структуровану відповідь, яку пересилає назад до користувача.

Архітектурно система розмежовує три незалежні рівні:

1. рівень взаємодії з користувачем (бот),
2. рівень прикладної логіки (Flask-сервер),
3. рівень машинного навчання і даних (модель + датасет).

Це дозволяє не лише розширювати функціональність кожного з рівнів окремо, але й спрощує тестування, оновлення та адаптацію системи під нові сценарії використання.

Додатковим елементом архітектури є адміністративний модуль, який дозволяє уповноваженому користувачу поповнювати датасет новими прикладами, ініціювати

перенавчання моделі та контролювати якість класифікації. Така можливість робить систему динамічною та самонавчальною, оскільки модель регулярно адаптується до нових типів фейкових повідомлень, що з’являються в інформаційному просторі.

Загальна структурна схема системи наведена на рисунку 4.1, де подано основні модулі та напрямки інформаційних потоків між ними.

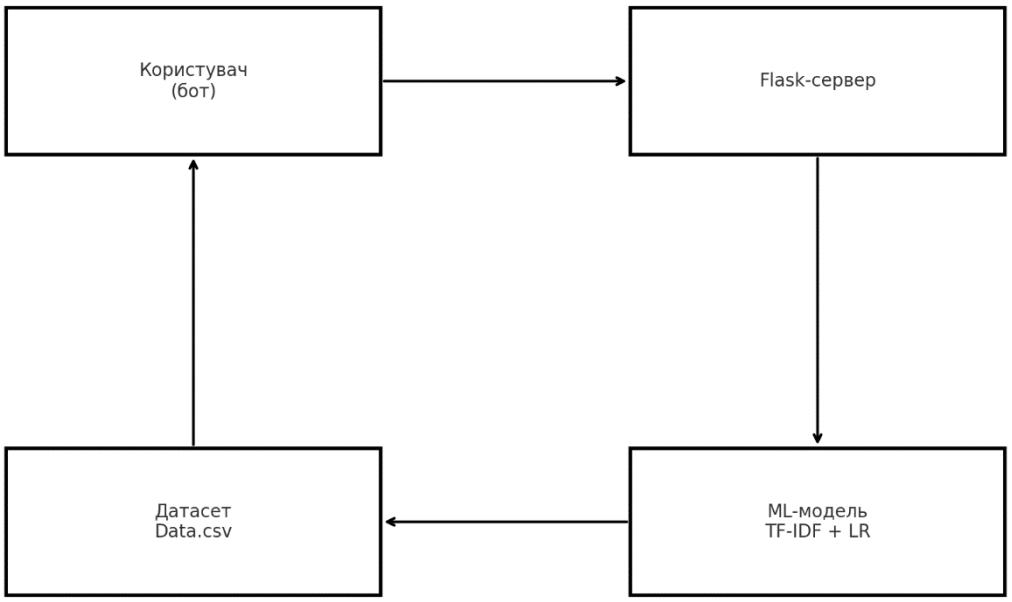

Рисунок 4.1 – Загальна архітектура системи класифікації фейкових новин

4.2 Реалізація серверного застосунку

Серверна частина розробленої системи реалізована з використанням мікрофреймворку Flask, який забезпечує обробку HTTP-запитів від бота, виклик модуля машинного навчання та формування відповіді користувачу. Серверний застосунок виступає центральною ланкою між інтерфейсом взаємодії та математичною моделлю, тому саме в ньому зосереджена основна прикладна логіка [13].

На початку файлу відбувається ініціалізація Flask-додатку, задання базових параметрів і глобальних змінних, що визначають шляхи до файлів даних та моделей, а також параметри взаємодії з ботом (Рисунок 4.2). Зокрема, задається токен доступу,

ідентифікатор адміністратора, URL-адреса API, а також константи DATA_PATH, MODEL_PATH та VECT_PATH, які вказують на розташування датасету, файлу моделі та векторизатора відповідно. Цей фрагмент коду формує основу конфігурації застосунку і забезпечує зв'язність між усіма компонентами системи.. Взаємодія між цими частинами відбувається через HTTP-запити, що забезпечує стандартизований механізм комунікації та мінімізує залежність компонентів один від одного.

```
8   app = Flask(__name__)
9
10  # ==== НАЛАШТУВАННЯ ====
11  TOKEN = "5233433991:AAGVeUz3b9MCpeREg5zZqyXiHufd0UUdIYk"
12  ADMIN_ID = 787431135
13
14
15  DATA_PATH   = "Data.csv"
16  MODEL_PATH  = "fake_news_model.pkl"
17  VECT_PATH   = "vectorizer.pkl"
18
```

Рисунок 4.2 – Фрагмент ініціалізації Flask-застосунку та базових налаштувань

Основним елементом серверної логіки є маршрут /webhook, який реалізований як обробник POST-запитів. Саме через нього бот передає на сервер вхідне повідомлення користувача у форматі JSON. Усередині функції webhook() здійснюється послідовна обробка отриманих даних: зчитування об'єкта message, визначення chat_id, отримання тексту, ідентифікація того, чи є користувач адміністратором, а також аналіз виду вхідного запиту (команда, текст, службове повідомлення) (Рисунок 4.3). На цьому етапі також обробляються callback-запити та ігноруються некоректні або порожні повідомлення. Логіка маршруту побудована таким чином, щоб одразу відсіювати неправильні запити й не навантажувати модель зайвими викликами.

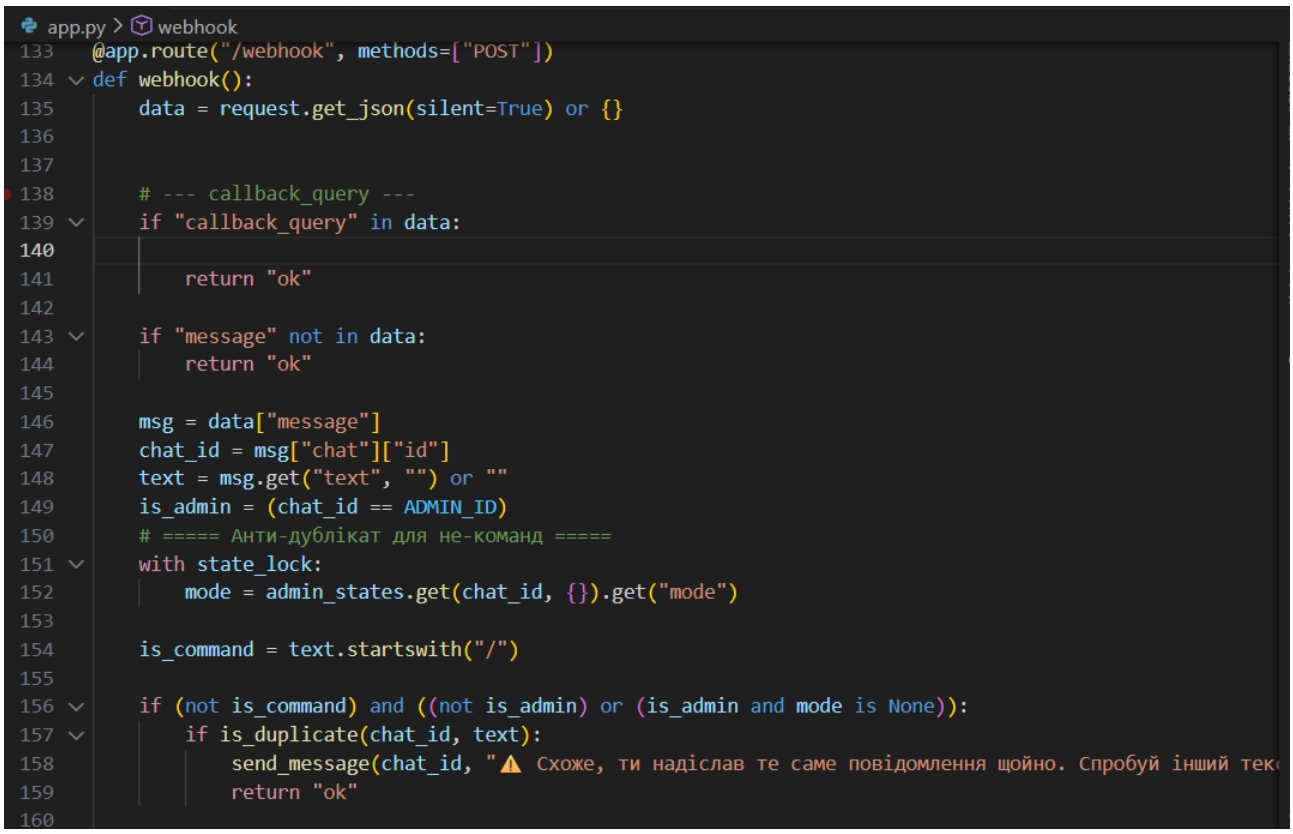

Рисунок 4.3 – Обробка HTTP-запиту у функції webhook()

У межах цього ж обробника реалізовано розгалуження логіки залежно від типу вхідного повідомлення. Якщо повідомлення є командою адміністратора (/admin, виклик режимів додавання правдивих чи фейкових новин, запуск перенавчання моделі), то система переводить чат у відповідний стан і очікує подальші дії. Для звичайного користувача, який надсилає текст для перевірки, система виконує валідацію мінімальної довжини та максимальної кількості символів, перевіряє текст на дублювання, а після цього переходить безпосередньо до етапу класифікації. Такий підхід забезпечує чітке розмежування прав доступу та зручність керування системою через адміністративні команди.

Ключовим програмним фрагментом є блок класифікації тексту, який використовує попередньо завантажені об’єкти моделі та векторизатора. Отриманий від користувача текст спочатку передається у функцію попередньої обробки preprocess, де нормалізується регістр, видаляються зайві символи та приводиться до уніфікованого вигляду. Далі очищений текст перетворюється у вектор ознак за допомогою vectorizer.transform([processed]). Отриманий вектор подається на вхід моделі

логістичної регресії, яка повертає або клас (predict), або ймовірність належності до класу «фейк» (predict_proba) (Рисунок 4.4). На основі цього формується текст відповіді для користувача: повідомлення позначається як фейкова або правдива новина, за необхідності із зазначення.

```
    processed = preprocess(text)
    X = vectorizer.transform([processed])

    if hasattr(model, "predict_proba"):
        proba_fake = float(model.predict_proba(X)[0, 1])  # ймовірність фейку

        base_thr = 0.55  # рекомендований поріг з train.py

        if proba_fake >= base_thr + 0.25:       # >= 0.80
            verdict = "❗ Ймовірно фейк (висока ймовірність)."
        elif proba_fake >= base_thr:            # 0.55–0.80
            verdict = "⚠️ Є ознаки фейку."
        elif proba_fake >= base_thr - 0.20:     # 0.35–0.55
            verdict = "🤔 Недостатньо даних. Перевір у декількох незалежних джерелах."
        elif proba_fake >= 0.20:                # 0.20–0.35
            verdict = "ℹ️ Більше ознак правдивості, але виникають сумніви."
        else:                                   # < 0.20
            verdict = "✅ Ймовірно правдива новина."

        msg = f"{verdict}\nЙмовірність фейку: {proba_fake:.1%}"
    else:
```

Рисунок 4.4 – Фрагмент коду класифікації текстового повідомлення

Окремий важливий компонент серверного застосунку — функція надсилання відповіді send_message(). Вона інкапсулює логіку формування запиту до API бота: створюється JSON-повідомлення із параметрами chat_id, текстом відповіді, режимом форматування та, за потреби, клавіатурою чи іншими елементами інтерфейсу. Для відправлення використовується HTTP-запит методом POST до відповідного endpoint-у, а у випадку неуспішної відповіді логуються код статусу та текст помилки. Така інкапсуляція дозволяє повторно використовувати функцію в різних місцях коду та полегшує супровід — у разі змін формату запитів правка здійснюється лише в одному місці (Рисунок 4.5).

```python
def send_message(chat_id, text, reply_markup=None, parse_mode="Markdown"):
    payload = {"chat_id": chat_id, "text": text, "parse_mode": parse_mode}
    if reply_markup: payload["reply_markup"] = reply_markup
    r = requests.post(f"{TELEGRAM_API_URL}/sendMessage", json=payload)
    if r.status_code != 200:
        print("[send_message ERR]", r.status_code, r.text)
```

Рисунок 4.5 – Функція надсилання відповіді користувачу через API

У поєднанні всі наведені фрагменти формують логічно завершений серверний застосунок, який виконує роль ядра системи: приймає запити від бота, обробляє їх, викликає модель машинного навчання та формує зрозумілу відповідь для користувача. Завдяки чіткому розділенню відповідальності між окремими функціями та використанню стандартних підходів до обробки HTTP-запитів система залишається розширюваною, зрозумілою для супроводу та придатною до подальшої модернізації.

4.3 Модуль машинного навчання та робота з моделлю

Модуль машинного навчання є центральною складовою системи, оскільки саме він забезпечує побудову класифікаційної моделі, її збереження, завантаження та інтеграцію в основний робочий процес. Його структура побудована таким чином, щоб модель могла працювати в режимі реального часу, а також регулярно оновлюватися на основі доповнених даних. Завдяки цьому система стає самонавчальною — її точність зростає у міру надходження нових прикладів фейкових та правдивих повідомлень.

Першим етапом взаємодії є завантаження збережених артефактів — векторизатора та моделі. Для цього застосовується функція, що перевіряє наявність відповідних файлів та зчитує їх у пам’ять (Рисунок 4.6). Такий підхід дозволяє серверу розпочинати роботу одразу після запуску, без потреби повторного тренування. Якщо хоча б один із артефактів відсутній, система переходить у захисний режим, блокуючи класифікацію та запобігаючи помилкам роботи.

```python
def load_artifacts():
    model = joblib.load(MODEL_PATH) if os.path.exists(MODEL_PATH) else None
    vect  = joblib.load(VECT_PATH)  if os.path.exists(VECT_PATH)  else None
    return model, vect

model, vectorizer = load_artifacts()
```

Рисунок 4.6 – Функція load_artifacts() для завантаження моделі та векторизатора

Коли адміністратор ініціює процес оновлення моделі, виконується механізм повторного навчання. Він запускає окремий процес, який здійснює повний цикл опрацювання вхідних даних: перевірку структури датасету, очищення текстів, векторизацію та навчання оновленої моделі (Рисунок 4.7). У випадку помилки система повертає повідомлення з кодом збою, що дозволяє швидко діагностувати проблему за логами. Такий метод забезпечує ізоляцію процесу тренування та гарантує безперервність роботи сервера.

```python
def retrain_and_reload():
    """Запускає train.py та перевантажує артефакти."""
    with retrain_lock:
        res = subprocess.run(
            ["python", "train.py"],
            capture_output=True,
            text=True,
        )
        print("[TRAIN OUT]\n", res.stdout)
        print("[TRAIN ERR]\n", res.stderr)

        if res.returncode != 0:
            # якщо тренування впало — не підміняємо модель
            return False, f"❌ Помилка тренування (код {res.returncode}). Дивись логи сервера."

        m, v = load_artifacts()
        if m is None or v is None:
            return False, "⚠️ Не вдалося підвантажити модель/векторизатор після тренування."

        global model, vectorizer
        model, vectorizer = m, v
        return True, "🤖 Модель оновлена і готова до роботи."
```

Рисунок 4.7 – Механізм повторного тренування моделі через retrain_and_reload()

Основна частина навчального циклу зосереджена у функції main() модуля, яка формує весь pipeline: від завантаження датасету до отримання метрик точності. Вона виконує валідацію структури вхідного файлу, фільтрацію некоректних рядків та приведення тексту до єдиного типу (Рисунок 4.8). Це дає змогу уникнути проблем під

час векторизації та зменшити кількість шумових даних, що можуть впливати на стабільність та якість моделі.

```
def main():
    # ===== 1. Завантаження даних =====
    if not DATA_PATH.exists():
        raise FileNotFoundError(f"Файл {DATA_PATH} не знайдено")

    df = pd.read_csv(DATA_PATH, on_bad_lines="skip")  # криві рядки пропускаємо

    if not {"text", "label"}.issubset(df.columns):
        raise ValueError("Data.csv повинен мати колонки: text,label")

    df["text"] = df["text"].astype(str)
```

Рисунок 4.8 – Фрагмент функції main() для підготовки даних

Для побудови ознак використовуються два TF-IDF векторизатори — словний та символьний. Перший генерує уні-, бі- та триграми слів, другий — символів, що дозволяє краще вловлювати помилки, стилістичні особливості текстів та структурні закономірності (Рисунок 4.9). Об’єднання двох типів ознак підвищує стійкість моделі до різних форматів подачі тексту, зокрема коротких фраз, неправильно сформованих речень чи повідомлень з орфографічними відхиленнями.

```
# ===== 2. Векторизатори =====
word_vect = TfidfVectorizer(
    analyzer="word",
    ngram_range=(1, 3),  # уні/бі/тріграми слів
    min_df=2,
    max_df=0.95,
    sublinear_tf=True,
    stop_words=UK_STOPWORDS,
)

char_vect = TfidfVectorizer(
    analyzer="char",
    ngram_range=(3, 5),  # символи — добре ловлять орфографію/мостику
    min_df=2,
    sublinear_tf=True,
)
```

Рисунок 4.9 – Налаштування TF-IDF векторизаторів (словного та символьного)

Як класифікатор використовується модель LinearSVC, додатково обгорнута у CalibratedClassifierCV для отримання коректних ймовірнісних прогнозів, необхідних

для відображення оцінки “імовірності фейку” в інтерфейсі користувача (Рисунок 4.10). Крос-валідація дозволяє покращити узагальнюючу здатність моделі, а калібрування — отримувати стабільні прогнози на основі сигмоїдної функції.

```
# ===== 3. Модель: LinearSVC + калібрування =====
base_clf = LinearSVC(
    class_weight="balanced",
    random_state=42,
)

clf = CalibratedClassifierCV(
    estimator=base_clf,
    cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42),
    method="sigmoid",
)

clf.fit(X_train, y_train)
```

Рисунок 4.9 – Побудова та калібрування моделі LinearSVC у CalibratedClassifierCV

Після завершення навчання обчислюються основні метрики — Accuracy, F1 та ROC-AUC, які зберігаються у логах і повертаються адміністратору як підтвердження успішності тренування (Рисунок 4.11). Це дозволяє оперативно відстежувати ефективність моделі та своєчасно реагувати на можливе погіршення якості класифікації.

```
report = classification_report(y_test, y_pred, digits=3)

print("Accuracy:", round(acc, 4))
print("F1:", round(f1, 4))
if auc is not None:
    print("ROC-AUC:", round(auc, 4))
print(report)
```

Рисунок 4.11 – Обчислення та виведення метрик моделі

У сукупності модуль машинного навчання забезпечує гнучку, масштабовану та надійну інфраструктуру для виконання автоматичного аналізу текстової інформації. Його архітектура дозволяє системі адаптуватися до нових даних, підтримувати

високу точність і стабільність роботи в умовах постійного зростання обсягів інформації.

4.4 Адміністративний функціонал, робота з датасетом та обробка помилок

Адміністративний модуль системи виконує критично важливу роль, оскільки саме через нього здійснюється керування датасетом, ініціалізація повторного навчання моделі та контроль якості системи в цілому. Цей функціонал реалізовано безпосередньо у боті, що дає змогу адміністратору інтерактивно взаємодіяти з усіма ключовими механізмами системи в реальному часі, не звертаючись до сервера вручну. Такий підхід значно спрощує процес збору нових даних, а також забезпечує стабільний розвиток моделі без необхідності глибоких технічних знань.

Після введення команди /admin система відображає адміністративну панель, яка включає кілька режимів:

- «Додати правдиву новину»,
- «Додати фейкову новину»,
- «Перенавчити модель»,
- «Скасувати».

У цьому режимі бот відстежує стан адміністратора і переводить діалог у відповідний контекст, що дозволяє уникнути двозначних ситуацій при введенні тексту. Інтерфейс адмін-панелі (Рисунок 4.12), де відображено початковий стан взаємодії адміністратора з системою.

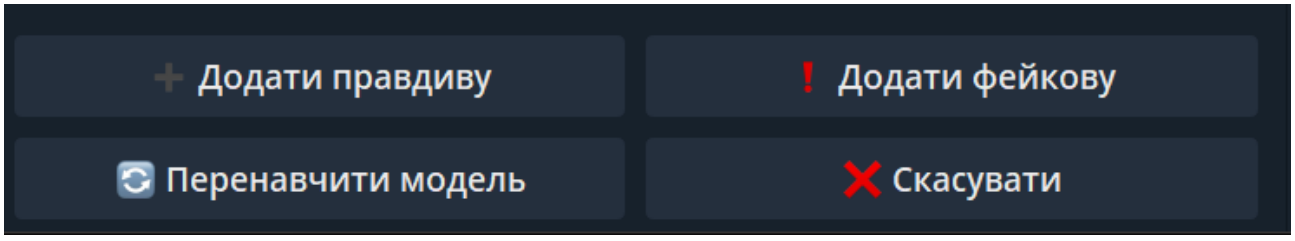

Рисунок 4.12 – Кнопки адміністративної панелі для роботи з датасетом

У межах цього функціоналу реалізовано механізм додавання нових записів у датасет. Після вибору режиму додавання правдивої або фейкової новини система надсилає інструкцію щодо обмежень за довжиною (мінімальна кількість слів та максимальна кількість символів). Якщо введений текст не відповідає вимогам, адміністратор отримує відповідне повідомлення про помилку (Рисунок 4.13). Це

дозволяє підтримувати якість даних, уникати надто коротких або некоректних повідомлень, що могли б знизити точність моделі.

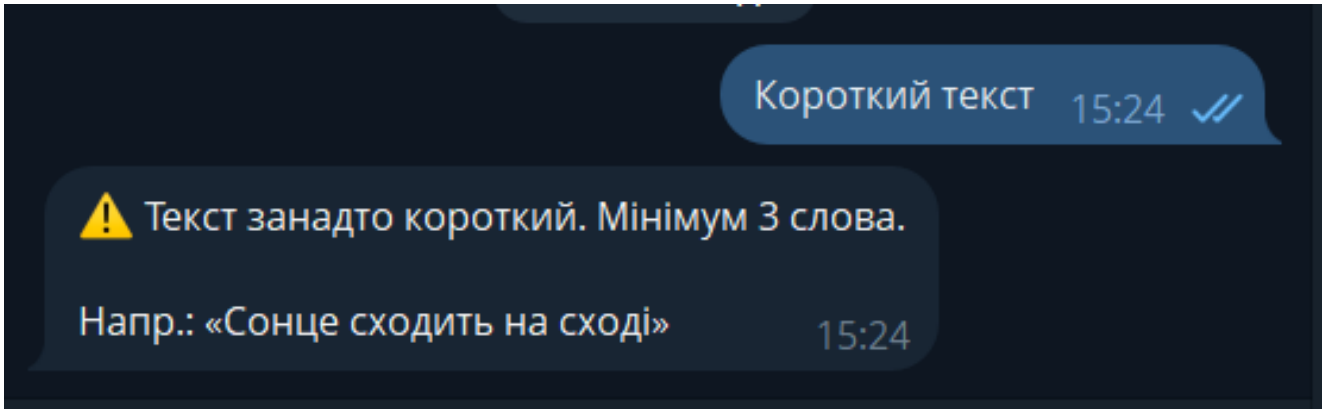

Рисунок 4.13 – Повідомлення системи про недостатню кількість слів у введеному тексті

Додані адміністратором повідомлення проходять нормалізацію, очищення від зайвих символів та записуються до файлу Data.csv. На відміну від звичайного користувача, адміністратор має можливість розширювати датасет у будь-який момент, що робить систему гнучкою та адаптивною (Рисунок 4.14).

```
if is_admin and text == "➕ Додати правдиву":
    with state_lock:
        admin_states[chat_id] = {"mode": "await_truth"}
    send_message(chat_id, f"Надішли текст правдивої новини (мінімум {MIN_WORDS} слова, максимум {MAX_CHARS} символів):")
    return "ok"
```

Рисунок 4.14 – Обробка натискання кнопки “Додати правдиву” у коді сервера

Ще одним важливим елементом є антидублікат, який запобігає повторному надсиланню однакових повідомлень протягом короткого проміжку часу. Для цього застосовується SHA-1 хешування нормалізованого тексту, а хеші зберігаються у словнику за ChatID. Якщо користувач або адміністратор повторно надсилає однаковий текст, система попереджає його про це (Рисунок 4.15). Такий підхід зменшує кількість зайвих запитів до моделі та запобігає неконтрольованому наповненню датасету.

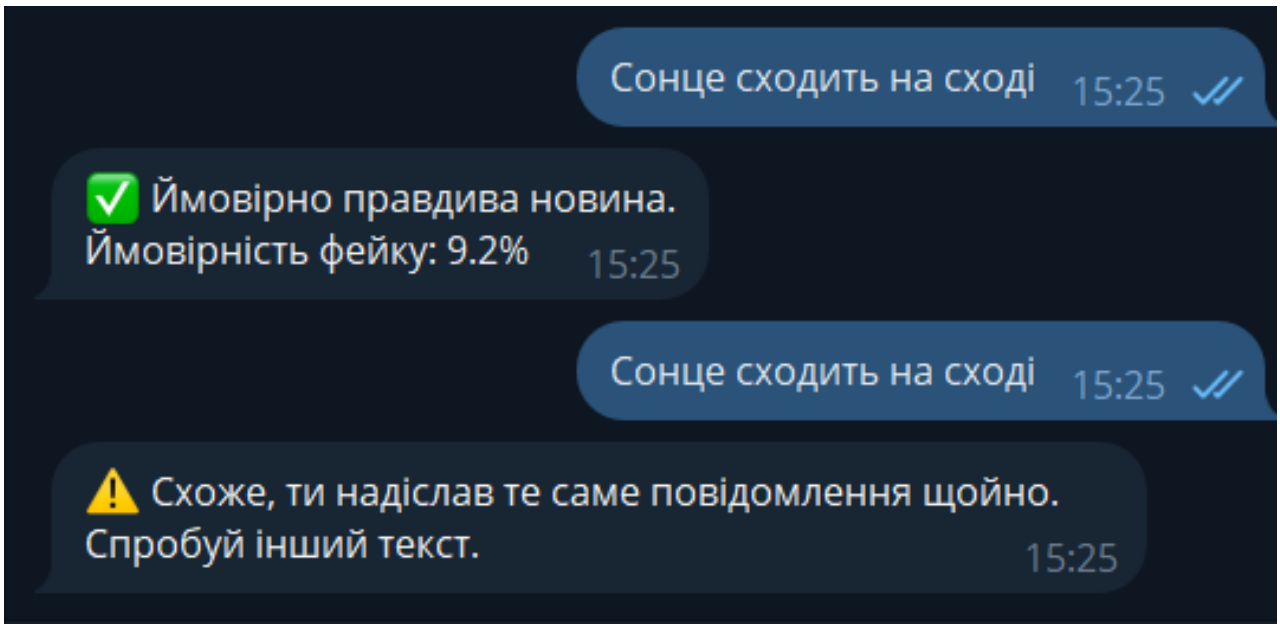

Рисунок 4.15 – Механізм виявлення дубльованих повідомлень від користувача

Особливу увагу приділено системі обробки помилок. Вона включає кілька рівнів захисту:

- перевірку структури JSON, отриманого від бота;
- перевірку наявності текстового поля message;
- відстеження помилок надсилання відповіді через API бота;
- обробку винятків, що можуть виникнути під час повторного навчання моделі.

Усі такі випадки фіксуються у логах, що дозволяє швидко ідентифікувати потенційні проблеми.

Після успішного додавання новин адміністратор може ініціювати повторне навчання моделі. Система повідомляє про початок процесу та повертає результат — успішний або з повідомленням про помилку (Рисунок 4.16). Це забезпечує прозорість роботи механізму оновлення та дозволяє контролювати якість побудованої моделі.

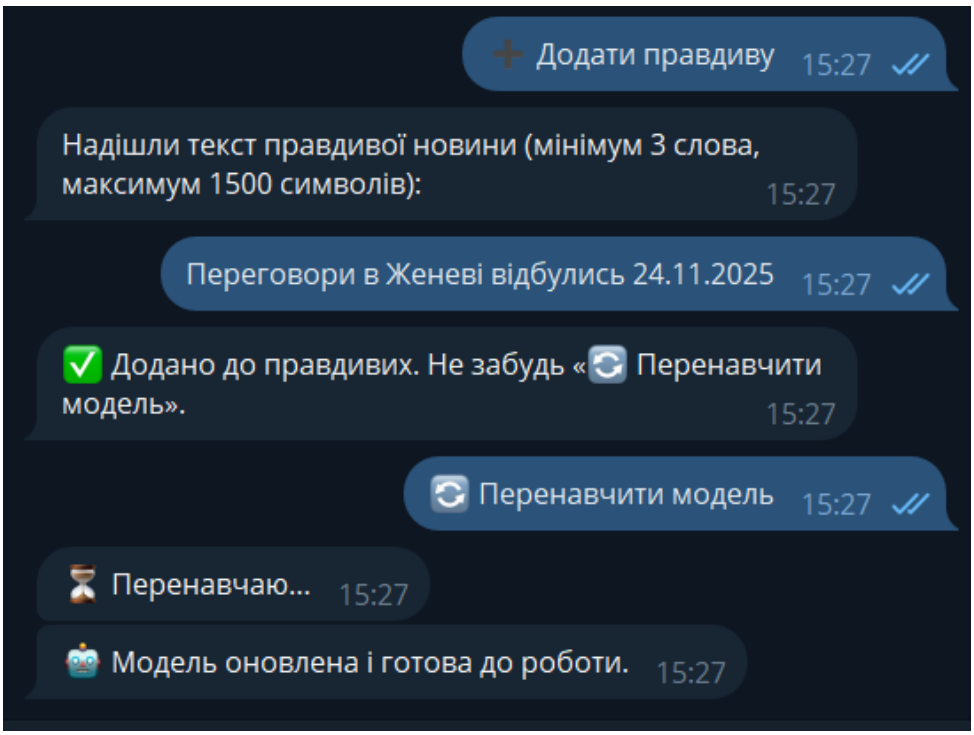

Рисунок 4.16 – Взаємодія адміністратора з системою: додавання нового прикладу та повторне навчання моделі

Для звичайного користувача реалізовано окремий механізм валідації введення. Повідомлення з надто малою кількістю слів, надто довгим текстом або таким, що містить лише спеціальні символи, автоматично блокуються системою, а користувач отримує підказку щодо коректного формату (Рисунок 4.17).

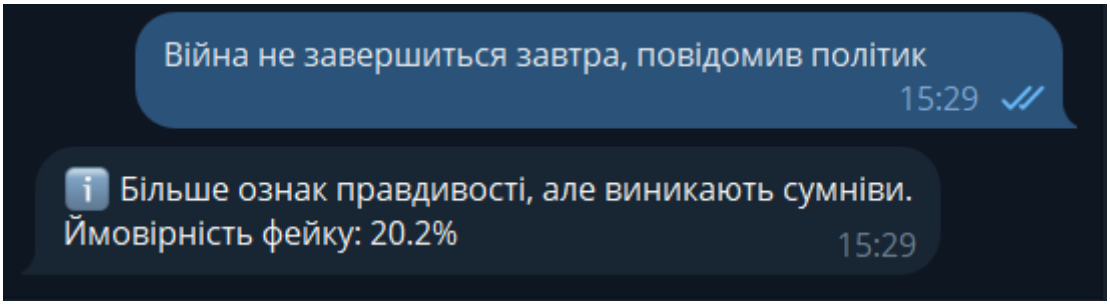

Рисунок 4.17 – Приклад класифікації новини з середнім рівнем ймовірності фейку

Таким чином, адміністративний модуль, обробка помилок та захисні механізми формують комплексну підсистему, яка гарантує стабільну роботу всієї системи. Завдяки валідації, контролю стану, структурованому оновленню даних та інформативним повідомленням система залишається надійною, масштабованою та зручною для подальшої експлуатації.

Висновки до розділу

У межах четвертого розділу було детально розглянуто процес програмної реалізації системи автоматичного визначення фейкових новин. Представлено архітектурну структуру програмного забезпечення, описано логіку взаємодії між сервером, моделлю машинного навчання та ботом, а також проаналізовано роботу основних модулів: попередньої обробки тексту, векторизації, класифікації та адміністративного керування датасетом.

Було продемонстровано застосування ключових функціональних компонентів, включно з механізмами перевірки введень, виявлення дубльованих повідомлень, виконання повторного навчання моделі та відображення результатів користувачу. Скріншоти програмного коду та приклади роботи бота підтверджують коректність реалізації та функціональну завершеність системи.

Загалом програмне забезпечення є цілісним, модульним та масштабованим рішенням, здатним ефективно обробляти текстові повідомлення в режимі реального часу, виконувати класифікацію з високою точністю та забезпечувати зручні інструменти адміністрування. Результати роботи підтверджують, що система готова до подальшого розширення, оптимізації та інтеграції у реальні інформаційні процеси.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1 Опис концепції та ціннісної пропозиції продукту

У сучасному інформаційному середовищі темпи поширення дезінформації постійно зростають. Соціальні мережі, месенджери та інформаційні платформи стали не лише інструментами комунікації, а й каналами поширення маніпулятивних чи неправдивих матеріалів, які здатні впливати на громадську думку, створювати соціальну напругу або формувати хибні уявлення щодо подій. У таких умовах зростає потреба у створенні інструментів, які б дозволяли оперативно аналізувати текстову інформацію та оцінювати ступінь її достовірності. Саме на розв’язання цього завдання спрямований розроблений стартап-проєкт.

Основна ідея стартапу полягає у створенні інтелектуальної системи, яка на основі методів машинного навчання автоматично визначає рівень фейковості текстових повідомлень. Продукт реалізується у вигляді інтерактивного сервісу, доступного користувачам через бота, що забезпечує швидку взаємодію та мінімальний поріг входу для широкого кола користувачів. Система працює в режимі реального часу, надаючи миттєвий аналіз тексту, визначаючи ймовірність того, що новина є неправдивою, та формуючи відповідне пояснення.

Ціннісна пропозиція продукту полягає у поєднанні технологічної складності та простоти використання. Користувачу не потрібно володіти технічними знаннями чи спеціальними навичками фактчекінгу — достатньо надіслати текст новини до бота, після чого система виконає повну обробку: нормалізацію, векторизацію, аналіз на основі класифікаційної моделі та формування відповіді. Завдяки каліброваній моделі машинного навчання користувач отримує не лише результат «фейк/правда», але й відсоткову оцінку, яка підвищує довіру до функціонування системи.

Особливої уваги заслуговує можливість самонавчання моделі шляхом розширення датасету. Адміністратор може додавати нові приклади фейкових та правдивих новин безпосередньо через інтерфейс бота, що робить продукт адаптивним до актуального інформаційного простору. У свою чергу, механізм повторного навчання забезпечує

регулярне оновлення моделі, покращення точності та підвищення її ефективності при обробці нових видів повідомлень.

Розроблений стартап здатен виконувати кілька важливих соціальних функцій. По-перше, він сприяє підвищенню рівня медіаграмотності населення, оскільки дозволяє користувачам перевіряти інформацію швидко та з мінімальними зусиллями. По-друге, система може стати корисним інструментом для журналістів, редакцій, державних установ та освітніх закладів, допомагаючи їм у процесі аналізу та перевірки новин. По-третє, її застосування сприяє зменшенню впливу інформаційних маніпуляцій та зниженню ризику поширення неправдивих матеріалів.

З технічної точки зору, продукт використовує сучасні підходи до обробки природної мови, поєднуючи TF-IDF-векторизацію зі стійкими класифікаційними алгоритмами. Така архітектура дозволяє забезпечити стабільну роботу системи навіть при значних обсягах даних. Разом з інтерактивним інтерфейсом бота це створює гнучке, масштабоване та технологічно актуальне рішення, здатне до подальшого розвитку та інтеграції у комерційні, освітні або інформаційні процеси.

Таким чином, концепція стартапу базується на поєднанні високотехнологічних методів аналізу даних, зручності користування та практичної користі для суспільства. Продукт не лише виконує прикладну функцію, але й має потенціал стати важливим інструментом у боротьбі з дезінформацією, підвищуючи довіру до інформаційного середовища та формуючи культуру відповідального споживання контенту.

5.2 Аналіз цільової аудиторії та сценаріїв застосування

Розроблений стартап-проєкт орієнтований на широку аудиторію, яка стикається з проблемою дезінформації у повсякденному житті. Оскільки система працює у форматі бота, її використання не потребує спеціальних знань, що значно розширює потенційний спектр користувачів.

Основну частину аудиторії становлять звичайні користувачі, які хочуть швидко перевірити сумнівну інформацію, новину або текстове повідомлення. Для них цінність полягає у можливості миттєво отримати оцінку достовірності контенту без необхідності проводити додатковий аналіз чи шукати незалежні джерела. Такий

сценарій використання особливо актуальний у соціальних мережах і месенджерах, де фейкові матеріали поширюються найшвидше.

Важливим сегментом аудиторії є журналісти та редакції, які працюють з великими обсягами інформації. Для них система може виступати допоміжним інструментом первинної оцінки повідомлень. Хоча бот не замінює повноцінну фактчекінгову перевірку, він може слугувати фільтром, який скорочує час на первинний аналіз великої кількості новин.

Окрему групу користувачів можуть становити освітні заклади — викладачі, студенти гуманітарних спеціальностей та учні старших класів. Система може застосовуватися в межах курсів медіаграмотності, як приклад практичного використання технологій штучного інтелекту для перевірки достовірності інформації. У такому форматі продукт виконує навчальну функцію і водночас демонструє принципи обробки текстових даних у реальних умовах.

Потенційними користувачами є також державні установи та організації, що займаються аналізом інформаційних потоків, інформаційною безпекою або протидією пропаганді. Система може бути інтегрована у внутрішні робочі процеси, виконуючи роль автоматизованого помічника для первинної оцінки текстів.

Сценарії застосування продукту диверсифіковані й можуть включати: швидку перевірку отриманих повідомлень у месенджерах; аналіз новин, що поширюються у медіапросторі; автоматичне опрацювання текстів адміністраторами для доповнення датасету; створення допоміжних сервісів на основі API. Така багатосценарність підтверджує практичну універсальність розробленого рішення та його здатність адаптуватися до різних потреб користувачів.

У результаті цільова аудиторія проєкту є широкою та різнорідною, що створює потенціал для масштабування і подальшого розвитку стартапу. Гнучкість інтерфейсу бота та можливість інтеграції в інші системи дозволяють розглядати продукт як багатопрофільний сервіс із високими перспективами у сфері інформаційної безпеки.

5.3 Аналіз ринку та конкурентів

Ринок інструментів для перевірки достовірності інформації перебуває у стадії активного розвитку, що зумовлено зростанням обсягів цифрового контенту та

поширенням дезінформації. Хоча повноцінні автоматизовані системи аналізу новин на основі штучного інтелекту ще не досягли масової популярності, на ринку існує низка рішень, з якими доцільно порівняти запропонований стартап-проєкт.

Серед міжнародних аналогів можна відзначити проєкти типу FactCheck.org, Snopes, PolitiFact та інші відомі фактчекінгові організації. Проте їх робота здебільшого базується на ручному аналізі інформації експертами, що робить такі системи малопридатними для швидкої автоматичної перевірки текстів. Вони не забезпечують миттєву відповідь користувачу, а отже, не можуть конкурувати з рішеннями, побудованими на машинному навчанні та здатними працювати у режимі реального часу.

У світовому просторі існують і частково автоматизовані системи, проте більшість із них орієнтовані на обробку великих масивів даних і використовуються медіаорганізаціями, а не пересічними користувачами. Таким чином, розроблений стартап займає окрему нішу — швидка індивідуальна перевірка текстових повідомлень через бота, що є більш доступною та практичною для широкої аудиторії.

В українському контексті найбільш відомими платформами фактчекінгу є StopFake, По той бік новин, VoxCheck, однак вони також працюють у редакційному форматі й не пропонують інтерактивних інструментів миттєвої автоматичної оцінки. Сервіси, орієнтовані на глибинний аналіз дезінформаційних кампаній, існують переважно в межах недержавних організацій і не представлені у формі доступних користувацьких сервісів.

Отже, прямої конкуренції у сегменті автоматичної класифікації новин за допомогою бота фактично немає. Це дає стартапу важливу перевагу — можливість зайняти ринок, що ще не заповнений подібними рішеннями. Унікальність полягає у поєднанні кількох характеристик: доступності, простоти використання, інтерактивності, швидкості аналізу та можливості регулярного самонавчання моделі.

Крім того, одна з ключових конкурентних переваг полягає у гнучкій архітектурі системи. Вона дозволяє швидко адаптувати модель до змін у мовному просторі, оновлювати датасет, інтегрувати нові алгоритми машинного навчання і розширювати функціональність без значних технічних витрат. Це створює сприятливі умови для

подальшого масштабування проєкту, включно з можливістю виходу на міжнародний ринок.

Оцінюючи конкурентне середовище загалом, можна стверджувати, що розроблений стартап має чітку нішу та унікальні характеристики, які дозволяють йому ефективно виділятися серед існуючих рішень. Це підвищує його ринковий потенціал і створює передумови для успішної комерціалізації продукту.

5.4 Бізнес-модель та план розвитку продукту

Ефективна бізнес-модель є важливим компонентом розвитку будь-якого стартапу, оскільки визначає механізми комерціалізації, ціннісні потоки та підходи до масштабування продукту. Розроблена система класифікації фейкових новин має потенціал для застосування у різних сферах — від індивідуального користування до інтеграції в інформаційні та корпоративні екосистеми. Це створює можливість побудови гнучкої бізнес-моделі, яка може адаптуватися до потреб ринку та різних категорій користувачів.

Основою бізнес-моделі є надання базового функціоналу у вільному доступі через бота, що забезпечує легкий старт використання та швидке залучення користувачів. Такий підхід значно знижує бар’єр входу та сприяє органічному зростанню аудиторії. Для більш вимогливих користувачів та організацій може бути реалізована розширена функціональність у форматі преміум-доступу. До таких можливостей належать: аналіз великих обсягів текстів, розширена статистика, пріоритетна обробка запитів, інтеграція з API та можливість навчання моделі під специфічні потреби.

Важливою складовою моделі монетизації може стати підписка для медіаорганізацій, журналістських редакцій, освітніх закладів та аналітичних центрів. Ці структури працюють з великими потоками інформації, а автоматизований інструмент первинної перевірки контенту може значно скоротити витрати часу та підвищити продуктивність їхньої діяльності. У корпоративному середовищі можливе створення окремих приватних моделей, натренованих на внутрішніх даних замовника, що відкриває потенціал для додаткових комерційних послуг.

Модель також передбачає можливість розширення функціоналу за рахунок продажу API-доступу для сторонніх розробників. Це дозволить інтегрувати систему

в інші сервіси або продукти, наприклад новинні агрегатори, системи модерації контенту чи освітні платформи. Таким чином, продукт може набути характеристик екосистемного рішення, а не лише окремого інструменту.

Одним із ключових елементів бізнес-моделі є здатність системи до самооновлення та покращення. Правильно організований процес збору нових прикладів та автоматизоване перевчання моделі дозволяють підтримувати її актуальність у динамічному інформаційному середовищі. У перспективі це може стати додатковою конкурентною перевагою, оскільки більшість традиційних фактчекінгових платформ не використовують настільки гнучкі механізми оновлення.

План розвитку продукту передбачає кілька етапів. Перший етап — завершення основної функціональності та запуск MVP-версії через бота для широкого кола користувачів. На цьому етапі важливо зібрати зворотний зв’язок, оцінити якість роботи моделі та визначити напрями покращення. Наступним етапом є впровадження API-інтерфейсу та створення панелі адміністратора, яка дозволить структурувати датасет, контролювати перевчання моделі та детальніше аналізувати дані.

Третій етап передбачає інтеграцію з освітніми та медіаплатформами, що може суттєво збільшити популярність продукту та розширити його функціональність. Паралельно з цим може відбуватися впровадження нових моделей машинного навчання, включно з алгоритмами трансформерного типу, які забезпечують більш точну обробку текстів. У довгостроковій перспективі планується можливість масштабування системи для багатомовної підтримки, а також створення мобільного застосунку, який інтегруватиме функціональність бота у зручному інтерфейсі.

Таким чином, бізнес-модель стартапу поєднує доступність базового функціоналу, можливість розширеної комерційної пропозиції та потенціал для масштабування у різних напрямках. Поетапний розвиток продукту, орієнтація на сучасні технології та фокус на реальних потребах користувачів створюють передумови для стабільного зростання та успішності проєкту в умовах сучасного інформаційного середовища.

Висновки до розділу

У цьому розділі було здійснено комплексний аналіз стартап-проєкту, присвяченого автоматичній класифікації фейкових новин на основі методів машинного навчання. Розглянуто концепцію продукту, його ключову ціннісну пропозицію та визначено основні переваги, що відрізняють систему від традиційних підходів до фактчекінгу. Показано, що запропоноване рішення є не лише технологічно актуальним, але й соціально значущим, оскільки сприяє формуванню більш надійного та свідомого інформаційного простору.

Аналіз цільової аудиторії продемонстрував широкий спектр потенційних користувачів — від пересічних громадян до професійних журналістів, освітніх установ та організацій, що працюють у сфері інформаційної безпеки. Така різноманітність підтверджує універсальний характер продукту та його здатність інтегруватися у різні сфери діяльності.

Дослідження конкурентного середовища показало, що на ринку переважають ручні або напівавтоматизовані рішення, які не забезпечують користувача швидкою та доступною перевіркою інформації. Це створює унікальну можливість для розвитку стартапу, оскільки продукт займає нішу, яка поки що не заповнена аналогічними технологіями у форматі інтерактивного сервісу.

Сформована бізнес-модель визначає шлях комерціалізації проєкту через поєднання безкоштовного доступу до базового функціоналу та розширених можливостей преміум-рівня. Також представлено поетапний план розвитку продукту, що включає вдосконалення алгоритмів, створення інфраструктури для масштабування та можливість виходу на корпоративний та міжнародний ринки.

Загалом проведений аналіз підтверджує потенційні переваги та життєздатність стартап-проєкту. Система має технологічний, соціальний та комерційний потенціал, що робить її перспективним рішенням у сучасних умовах зростання інформаційних загроз і необхідності формування ефективних.

ВИСНОВОК

У магістерській дипломній роботі було виконано повний цикл створення інтелектуальної системи для автоматизованого визначення фейкових новин на основі аналізу текстових повідомлень. У ході дослідження проаналізовано особливості поширення дезінформації, сучасні методи NLP та підходи до машинного навчання, що дозволило сформувати вимоги до структури та функціональності майбутнього програмного комплексу.

У межах математичного та інформаційного забезпечення розроблено алгоритмічну основу системи, що включає комбіновану TF-IDF векторизацію, використання моделі LinearSVC із подальшим калібруванням та впровадження метрик для оцінювання якості (Accuracy, F1, ROC-AUC). Результати експериментів продемонстрували стабільну роботу моделі та високу точність класифікації у задачі визначення фейкових текстів.

Програмна реалізація охопила створення серверного застосунку, модуля машинного навчання, механізмів обробки помилок та безпечного перенавчання моделі. Було реалізовано зручний функціонал для адміністратора — додавання нових прикладів, оновлення моделі та отримання службових повідомлень. Для кінцевих користувачів створено інтуїтивний інтерфейс взаємодії через бота, який забезпечує швидку оцінку достовірності тексту та надає рекомендації.

Тестування системи підтвердило її працездатність, стійкість і здатність до подальшого масштабування. Створений програмний комплекс має практичну цінність, може бути інтегрований у реальні сервіси інформаційної безпеки та має потенціал розвитку — зокрема, у напрямі розширення датасету, багатомовної підтримки та застосування сучасніших нейронних моделей.

Виконана робота демонструє ефективність поєднання методів NLP, машинного навчання та автоматизованої взаємодії з користувачем, а отриманий результат може бути основою для майбутніх досліджень та вдосконалення систем боротьби з дезінформацією.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. B. Downey A. (2015). Think Python: How to Think Like a Computer Scientist. 292 p.;
2. Miguel Grinberg (2016). Flask Web Development: Developing Web Applications with Python. 256 p.;
3. Mark Lutz (2025). Learning Python: Powerful Object-Oriented Programming 6th Edition. 1169 p.;
4. Hoobyar T., Dotz T. (2018). NLP: The Essential Guide to Neuro-Linguistic Programming. 543 p.;
5. Ready R., Burton K. (2018). Neuro-linguistic Programming For Dummies. 448 p.;
6. Dr. Hidaia Mahmood Alassouli (2021). Introductory Guideline for Using Twilio Programmable Messaging and Programmable Voice Services. 112 p.;
7. Steven Bird, Ewan Klein, Edward Loper (2019). Natural Language Processing with Python. 504 p.;
8. Christopher D. Manning, Hinrich Schütze (2021). Foundations of Statistical Natural Language Processing. 680 p.;
9. Trevor Hastie, Robert Tibshirani, Jerome Friedman (2022). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 745 p.;
10. Ian Goodfellow, Yoshua Bengio, Aaron Courville (2016). Deep Learning. 800 p.;
11. Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani (2021). An Introduction to Statistical Learning (2nd Edition). 610 p.;
12. Python documentation. URL: https://www.python.org/doc/ (дата звернення: 20.10.2025);
13. Flask’s documentation. URL: https://flask.palletsprojects.com/en/stable/ (дата звернення: 05.05.2025);
14. Ngrok API. URL: https://ngrok.com/ (дата звернення: 02.06.2025);
15. NLP - Natural Language Processing with Python. URL: https://www.udemy.com/course/nlp-natural-language-processing-with-python/ (дата звернення: 10.06.2025);

16. TensorFlow Decision Forests. URL: https://www.tensorflow.org/decision_forests (дата звернення: 12.07.2025);
17. Scikit-learn documentation. URL: https://scikit-learn.org/stable/ (дата звернення: 15.07.2025);
18. Pandas documentation. URL: https://pandas.pydata.org/docs/ (дата звернення: 21.08.2025);
19. NLTK Book online. URL: https://www.nltk.org/book/ (дата звернення: 10.09.2025);
20. ISO/IEC 27001:2022 Information security, cybersecurity and privacy protection — Requirements. 45 p.

ДОДАТКИ

app.py

```
# app.py
from flask import Flask, request
import requests
import joblib
import csv, os, re, subprocess
from threading import Lock

app = Flask(__name__)

# ==== НАЛАШТУВАННЯ ====
TOKEN = "token"     #
ADMIN_ID = 787******
API_URL = f" {TOKEN}"

DATA_PATH   = "Data.csv"
MODEL_PATH  = "fake_news_model.pkl"
VECT_PATH   = "vectorizer.pkl"

# Обмеження вводу
MIN_WORDS  = 3
MAX_CHARS  = 1500

# СТАН адмін-діалогів
state_lock = Lock()
admin_states = {}  # {chat_id: {"mode": None|"await_truth"|"await_fake"}}
retrain_lock = Lock()

# ==== ХЕЛПЕРИ ====
def send_message(chat_id, text, reply_markup=None, parse_mode="Markdown"):
    payload = {"chat_id": chat_id, "text": text, "parse_mode": parse_mode}
    if reply_markup: payload["reply_markup"] = reply_markup
    r = requests.post(f"{ API_URL}/sendMessage", json=payload)
    if r.status_code != 200:
        print("[send_message ERR]", r.status_code, r.text)

# анти-дублікат (по хешу нормалізованого тексту)
import hashlib, time
recent_hashes = {}  # {chat_id: (hash, ts)}
DUP_WINDOW = 30     # сек

def is_duplicate(chat_id, text):
    key = (preprocess(text)).strip()
    h = hashlib.sha1(key.encode("utf-8")).hexdigest()
    prev = recent_hashes.get(chat_id)
    now = time.time()
    if prev and prev[0] == h and now - prev[1] < DUP_WINDOW:
        return True
    recent_hashes[chat_id] = (h, now)
    return False


def admin_keyboard():
    return {
        "keyboard": [
```

```
            [{"text": "➕ Додати правдиву"}, {"text": "❗ Додати фейкову"}],
            [{"text": "🔄 Перенавчити модель"}, {"text": "❌ Скасувати"}],
        ],
        "resize_keyboard": True
    }

def ensure_data_exists():
    if not os.path.exists(DATA_PATH):
        with open(DATA_PATH, "w", newline="", encoding="utf-8") as f:
            csv.writer(f).writerow(["text", "label"])

def count_words(text: str) -> int:
    return len(re.findall(r"[A-Za-zA-Яа-яЇїІіЄєҐґ0-9']+", (text or "")))

def too_short_msg():
    return f"⚠️ Текст занадто короткий. Мінімум {MIN_WORDS} слова."

def too_long_msg():
    return f"⚠️ Текст надто довгий. Максимум {MAX_CHARS} символів.\nНадішли коротший виклад новини."

def normalize_space(s: str) -> str:
    s = (s or "").strip()
    s = re.sub(r"\s+", " ", s)
    return s

def append_example(text: str, label: int):
    """Додає приклад у Data.csv з валідаціями. Повертає True/False."""
    txt = normalize_space(text)
    if count_words(txt) < MIN_WORDS:
        print(f"[WARN] not saved, too short: {txt!r}")
        return False
    if len(txt) > MAX_CHARS:
        print(f"[WARN] not saved, too long ({len(txt)} chars)")
        return False

    ensure_data_exists()
    with open(DATA_PATH, "a", newline="", encoding="utf-8") as f:
        csv.writer(f).writerow([txt, label])
    return True

def preprocess(s: str) -> str:
    s = (s or "").lower()
    s = re.sub(r"[^a-za-яїієґ0-9 ]", " ", s)
    s = re.sub(r"\s+", " ", s).strip()
    return s

def load_artifacts():
    model = joblib.load(MODEL_PATH) if os.path.exists(MODEL_PATH) else None
    vect  = joblib.load(VECT_PATH)  if os.path.exists(VECT_PATH)  else None
    return model, vect

model, vectorizer = load_artifacts()

def retrain_and_reload():
    """Запускає train.py та перевантажує артефакти."""
    with retrain_lock:
        res = subprocess.run(
            ["python", "train.py"],
            capture_output=True,
            text=True,
        )
        print("[TRAIN OUT]\n", res.stdout)
```

```
        print("[TRAIN ERR]\n", res.stderr)

        if res.returncode != 0:
            # якщо тренування впало — не підміняємо модель
            return False, f"❌ Помилка тренування (код {res.returncode}). Дивись логи сервера."

        m, v = load_artifacts()
        if m is None or v is None:
            return False, "⚠️ Не вдалося підвантажити модель/векторизатор після тренування."

        global model, vectorizer
        model, vectorizer = m, v
        return True, "🤖 Модель оновлена і готова до роботи."

# ==== ВЕБХУК ====
@app.route("/webhook", methods=["POST"])
def webhook():
    data = request.get_json(silent=True) or {}


    # --- callback_query ---
    if "callback_query" in data:

        return "ok"

    if "message" not in data:
        return "ok"

    msg = data["message"]
    chat_id = msg["chat"]["id"]
    text = msg.get("text", "") or ""
    is_admin = (chat_id == ADMIN_ID)
    # ===== Анти-дублікат для не-команд =====
    with state_lock:
        mode = admin_states.get(chat_id, {}).get("mode")

    is_command = text.startswith("/")

    if (not is_command) and ((not is_admin) or (is_admin and mode is None)):
        if is_duplicate(chat_id, text):
            send_message(chat_id, "⚠️ Схоже, ти надіслав те саме повідомлення щойно. Спробуй інший текст.")
            return "ok"

    if text.strip().lower() == "/ping":
        send_message(chat_id, "pong ☑")
        return "ok"

    if text.strip().lower() == "/start":
        send_message(chat_id, "Привіт! Я бот для попереднього аналізу новин. Надішли короткий виклад новини (мінімум 3 слова, максимум 1500 символів). Напр.: _«Сонце сходить на сході»_.")
        return "ok"

    if text.strip().lower() == "/help":
        send_message(chat_id, "ℹ *Як користуватись*\n• Мінімум 3 слова, максимум 1500 символів.\n• Краще додай джерело або контекст.\n• Отримаєш оцінку: правда/фейк/не впевнений.")
        return "ok"

    if text.strip().lower() == "/policy":
        send_message(chat_id, "⚠️ Це інструмент попереднього аналізу. Результат не є остаточною фактчек-оцінкою.")
        return "ok"
```

```
    # /admin — показати меню
    if is_admin and text.strip() == "/admin":
        with state_lock:
            admin_states[chat_id] = {"mode": None}
        send_message(chat_id, "👋 Адмін-панель:", reply_markup=admin_keyboard())
        return "ok"

    # Швидкі префікси без кнопок (опційно зручно)
    if is_admin and text.lower().startswith("truth:"):
        clean = text.split(":", 1)[1].strip()
        if count_words(clean) < MIN_WORDS:
            send_message(chat_id, too_short_msg())
            return "ok"
        if len(clean) > MAX_CHARS:
            send_message(chat_id, too_long_msg())
            return "ok"
        ok = append_example(clean, 0)
        send_message(chat_id, "☑ Додано як правдиву. Натисни «🔄 Перенавчити модель»." if ok else too_short_msg())
        return "ok"

    if is_admin and text.lower().startswith("fake:"):
        clean = text.split(":", 1)[1].strip()
        if count_words(clean) < MIN_WORDS:
            send_message(chat_id, too_short_msg())
            return "ok"
        if len(clean) > MAX_CHARS:
            send_message(chat_id, too_long_msg())
            return "ok"
        ok = append_example(clean, 1)
        send_message(chat_id, "⚠️ Додано як фейкову. Натисни «🔄 Перенавчити модель»." if ok else too_short_msg())
        return "ok"

    # Кнопки адмін-меню
    if is_admin and text == "❌ Скасувати":
        with state_lock:
            admin_states[chat_id] = {"mode": None}
        send_message(chat_id, "☑ Скасовано.", reply_markup=admin_keyboard())
        return "ok"

    if is_admin and text == "➕ Додати правдиву":
        with state_lock:
            admin_states[chat_id] = {"mode": "await_truth"}
        send_message(chat_id, f"Надішли текст правдивої новини (мінімум {MIN_WORDS} слова, максимум
{MAX_CHARS} символів):")
        return "ok"

    if is_admin and text == "❗ Додати фейкову":
        with state_lock:
            admin_states[chat_id] = {"mode": "await_fake"}
        send_message(chat_id, f"Надішли текст фейкової новини (мінімум {MIN_WORDS} слова, максимум
{MAX_CHARS} символів):")
        return "ok"

    if is_admin and text == "🔄 Перенавчити модель":
        send_message(chat_id, "⏳ Перенавчаю…")
        ok, msg_txt = retrain_and_reload()
        send_message(chat_id, msg_txt, reply_markup=admin_keyboard())
        return "ok"
```

```
    # Прийом тексту після вибору режиму
    with state_lock:
        mode = admin_states.get(chat_id, {}).get("mode")

    if is_admin and mode == "await_truth":
        clean = text.strip()
        if count_words(clean) < MIN_WORDS:
            send_message(chat_id, too_short_msg())
            return "ok"
        if len(clean) > MAX_CHARS:
            send_message(chat_id, too_long_msg())
            return "ok"
        ok = append_example(clean, 0)
        if ok:
            with state_lock:
                admin_states[chat_id] = {"mode": None}
            send_message(chat_id, "☑ Додано до правдивих. Не забудь «🔄 Перенавчити модель».",
reply_markup=admin_keyboard())
        else:
            send_message(chat_id, too_short_msg())
        return "ok"

    if is_admin and mode == "await_fake":
        clean = text.strip()
        if count_words(clean) < MIN_WORDS:
            send_message(chat_id, too_short_msg())
            return "ok"
        if len(clean) > MAX_CHARS:
            send_message(chat_id, too_long_msg())
            return "ok"
        ok = append_example(clean, 1)
        if ok:
            with state_lock:
                admin_states[chat_id] = {"mode": None}
            send_message(chat_id, "⚠️ Додано до фейкових. Не забудь «🔄 Перенавчити модель».",
reply_markup=admin_keyboard())
        else:
            send_message(chat_id, too_short_msg())
        return "ok"



        # === КЛАСИФІКАЦІЯ ДЛЯ ВСІХ КОРИСТУВАЧІВ ===
    if model is None or vectorizer is None:
        send_message(chat_id, "⚠️ Модель ще не готова. Попроси адміна її натренувати.")
        return "ok"

    # Валідації вводу для користувача
    if count_words(text) < MIN_WORDS:
        send_message(chat_id, too_short_msg() + "\n\nНапр.: «Сонце сходить на сході»")
        return "ok"
    if len(text) > MAX_CHARS:
        send_message(chat_id, too_long_msg())
        return "ok"

    processed = preprocess(text)
    X = vectorizer.transform([processed])

    if hasattr(model, "predict_proba"):
        proba_fake = float(model.predict_proba(X)[0, 1])  # ймовірність фейку

        base_thr = 0.55  # рекомендований поріг з train.py
```

```
        if proba_fake >= base_thr + 0.25:       # >= 0.80
            verdict = "❗ Ймовірно фейк (висока ймовірність)."
        elif proba_fake >= base_thr:            # 0.55–0.80
            verdict = "⚠️ Є ознаки фейку."
        elif proba_fake >= base_thr - 0.20:     # 0.35–0.55
            verdict = "🧐 Недостатньо даних. Перевір у декількох незалежних джерелах."
        elif proba_fake >= 0.20:                # 0.20–0.35
            verdict = "ℹ Більше ознак правдивості, але виникають сумніви."
        else:                                   # < 0.20
            verdict = "☑ Ймовірно правдива новина."

        msg = f"{verdict}\nЙмовірність фейку: {proba_fake:.1%}"
    else:
        # запасний варіант, якщо з якоїсь причини нема predict_proba
        pred = model.predict(X)[0]
        msg = "❗ Фейк!" if pred == 1 else "☑ Правдива новина."

    send_message(chat_id, msg)
    return "ok"




if __name__ == "__main__":
    ensure_data_exists()
    app.run(port=5000, debug=True)
```

train.py

```
# train.py
import re
from pathlib import Path

import joblib
import numpy as np
import pandas as pd
from sklearn.calibration import CalibratedClassifierCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    f1_score,
    roc_auc_score,
)
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.pipeline import FeatureUnion
from sklearn.svm import LinearSVC

# Шляхи
DATA_PATH = Path("Data.csv")
MODEL_PATH = Path("fake_news_model.pkl")
VECT_PATH = Path("vectorizer.pkl")
METRICS_PATH = Path("metrics.txt")

MIN_WORDS = 3
MAX_CHARS = 1500
```

```
def preprocess(text: str) -> str:
    """Проста нормалізація для укр/лат тексту."""
    s = (text or "").lower()
    s = re.sub(r"[^a-za-яїієґ0-9 ]", " ", s)
    s = re.sub(r"\s+", " ", s).strip()
    return s

def count_words(text: str) -> int:
    return len(re.findall(r"[A-Za-zA-Яа-яЇїІіЄєҐґ0-9']+", (text or "")))

# Невеликий список укр. стоп-слів (можеш доповнити)
UK_STOPWORDS = [
    "і",
    "й",
    "та",
    "але",
    "а",
    "що",
    "це",
    "цею",
    "у",
    "в",
    "на",
    "до",
    "з",
    "із",
    "за",
    "від",
    "як",
    "також",
    "або",
    "то",
    "для",
    "про",
    "при",
    "над",
    "під",
    "без",
    "коли",
    "де",
    "куди",
    "тут",
    "там",
    "вони",
    "ми",
    "ви",
    "я",
    "ти",
    "він",
    "вона",
    "воно",
    "їх",
    "них",
    "нас",
    "вас",
]

def main():
    # ===== 1. Завантаження даних =====
    if not DATA_PATH.exists():
        raise FileNotFoundError(f"Файл {DATA_PATH} не знайдено")
```

```
df = pd.read_csv(DATA_PATH, on_bad_lines="skip")  # криві рядки пропускаємо

if not {"text", "label"}.issubset(df.columns):
    raise ValueError("Data.csv повинен мати колонки: text,label")

df["text"] = df["text"].astype(str)

# фільтр по довжині / кількості слів
df = df[
    (df["text"].str.len() <= MAX_CHARS)
    & (df["text"].apply(count_words) >= MIN_WORDS)
]

if df.empty:
    raise ValueError("Після фільтрації дані порожні. Перевір Data.csv.")

# препроцес
df["text"] = df["text"].apply(preprocess)

X_text = df["text"].values
y = df["label"].astype(int).values

X_train_text, X_test_text, y_train, y_test = train_test_split(
    X_text, y, test_size=0.2, random_state=42, stratify=y
)

# ===== 2. Векторизатори =====
word_vect = TfidfVectorizer(
    analyzer="word",
    ngram_range=(1, 3),  # уні/бі/тріграми слів
    min_df=2,
    max_df=0.95,
    sublinear_tf=True,
    stop_words=UK_STOPWORDS,
)

char_vect = TfidfVectorizer(
    analyzer="char",
    ngram_range=(3, 5),  # символи — добре ловлять орфографію/мостику
    min_df=2,
    sublinear_tf=True,
)

vectorizer = FeatureUnion(
    [("word", word_vect), ("char", char_vect)],
    transformer_weights={"word": 2.0, "char": 1.0},
)

X_train = vectorizer.fit_transform(X_train_text)
X_test = vectorizer.transform(X_test_text)

# ===== 3. Модель: LinearSVC + калібрування =====
base_clf = LinearSVC(
    class_weight="balanced",
    random_state=42,
)

clf = CalibratedClassifierCV(
    estimator=base_clf,
    cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42),
    method="sigmoid",
```

```
    )

    clf.fit(X_train, y_train)

    # ===== 4. Оцінка =====
    y_proba = clf.predict_proba(X_test)[:, 1]  # ймовірність "фейк"
    # базовий поріг 0.5
    y_pred = (y_proba >= 0.5).astype(int)

    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    try:
        auc = roc_auc_score(y_test, y_proba)
    except Exception:
        auc = None

    report = classification_report(y_test, y_pred, digits=3)

    print("Accuracy:", round(acc, 4))
    print("F1:", round(f1, 4))
    if auc is not None:
        print("ROC-AUC:", round(auc, 4))
    print(report)

    # ===== 5. Підбір рекомендованого порогу за F1 =====
    thresholds = np.linspace(0.2, 0.8, 25)
    best_thr = 0.5
    best_f1 = 0.0

    for thr in thresholds:
        preds_thr = (y_proba >= thr).astype(int)
        f1_thr = f1_score(y_test, preds_thr)
        if f1_thr > best_f1:
            best_f1 = f1_thr
            best_thr = thr

    print(
        f"Рекомендований поріг для класу 'фейк' за F1: {best_thr:.2f} (F1={best_f1:.4f})"
    )

    # ===== 6. Запис метрик у файл =====
    auc_str = f"{auc:.4f}" if auc is not None else "N/A"
    text = (
        f"Accuracy: {acc:.4f}\n"
        f"F1 (thr=0.50): {f1:.4f}\n"
        f"ROC-AUC: {auc_str}\n"
        f"Best_threshold_F1_for_fake: {best_thr:.2f}\n"
        f"Best_F1_at_that_threshold: {best_f1:.4f}\n\n"
        f"{report}"
    )
    METRICS_PATH.write_text(text, encoding="utf-8")

    # ===== 7. Збереження моделі й векторизатора =====
    joblib.dump(clf, MODEL_PATH)
    joblib.dump(vectorizer, VECT_PATH)
    print("Збережено:", MODEL_PATH, "та", VECT_PATH)

if __name__ == "__main__":
    main()
```