Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: Розробка програм для автоматичного виявлення фейкових новин у соціальних мережах

Виконав: студент IV курсу
групи 2ПІ-20Б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Волосенко Віталій Федорович
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Чехместрук Р.Ю.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ОТ Войцеховська О.В.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ Романюк О.Н.
«10» червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк _____
«12» березня 2024 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Волосенку Віталію Федоровичу

1. Тема роботи – розробка програм для автоматичного виявлення фейкових новин у соціальних мережах.

Керівник роботи: Чехместрук Роман Юрійович, к.т.н. доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи: середовище розробки PyCharm, мова розробки Python, бібліотека scikit-learn, бібліотека tkinter, алгоритм наївного Байєса, лінійний класифікатор.

4. Зміст текстової частини: вступ, аналіз розвитку технологій аналізу, порівняльний аналіз аналогів, аналіз методів розв'язання задачі, постановка задач дипломної роботи, розробка моделі та алгоритмів; тестування; розробка інструкції користувача; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу: титульний слайд; мета, об'єкт та предмет дослідження, загальний огляд проблеми; аналіз сучасних підходів; методи і технології; інтерфейс користувача; модель і алгоритми; тестування; практична цінність; апробація та публікації результатів роботи; висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Чехместрук Р.Ю., к.т.н., доцент кафедри ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ п\п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану технологій аналізу інформації	13.03.24 – 30.03.24	Виконав
2	Вибір середовища та мови розробки	02.04.24 – 15.04.24	Виконав
3	Розробка інтерфейсу, моделі та алгоритмів програмного застосунку	20.04.24 – 30.04.24	Виконав
4	Розробка модуля класифікації тексту	01.05.24 – 09.05.24	Виконав
5	Розробка модуля потоків виконання та нормалізації вхідних даних	10.05.24 – 19.05.24	Виконав
6	Тестування	20.05.24 – 24.05.24	Виконав
7	Оформлення матеріалів до захисту БДР	30.05.24 – 03.06.24	Виконав

Студент _____ **В. Ф. Волосенко**
(підпис) (ініціали та прізвище)

Керівник бакалаврської дипломної роботи _____ **Р. Ю. Чехместрук**
(підпис) (ініціали та прізвище)

Анотація

УДК 004.912.032.26

Волосенко В.Ф. Розробка програм для автоматичного виявлення фейкових новин в соціальних мережах: бакалаврська дипломна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 52 с.

На укр. мові. Бібліогр. назв: 21 ; рис. : 29; табл. 2.

У бакалаврській дипломній роботі було розроблено програму для автоматичного виявлення фейкових новин у соціальних мережах. Проведено аналіз сучасних підходів до виявлення дезінформації з використанням методів машинного навчання та обробки тексту. Обґрунтовано необхідність розробки нових методів для виявлення фейкових новин з метою підвищення точності та ефективності в умовах постійно зростаючого обсягу інформації.

Розроблено моделі класифікації тексту та алгоритми обробки даних. Використовуючи сучасні технології машинного навчання, створено моделі, які дозволяють з високою точністю визначати дезінформацію.

На основі проведених досліджень розроблено програмне забезпечення, що включає модулі класифікації тексту, обробки потоків виконання та нормалізації вхідних даних. Проведене тестування підтвердило достовірність теоретичних досліджень та ефективність реалізованих алгоритмів.

Програмні модулі, розроблені в проєкті можуть бути використані дослідницькими групами, новинними порталами для аналізу та вивчення поширення фейкової інформації.

Ключові слова: фейкові новини, соціальні мережі, машинне навчання, обробка тексту, програмне забезпечення.

Annotation

UDC 004.912.032.26

Volosenko V.F. Development of software for automatic detection of fake news in social networks: bachelor's thesis in the specialty 121 - Software Engineering, educational program - Software Engineering. Vinnytsia: VNTU, 2024. 52 p.

In Ukrainian. Bibliography: titles: 21; figures: 29; tables: 2.

The bachelor's thesis developed software for the automatic detection of fake news in social networks. An analysis of modern approaches to the detection of disinformation using machine learning methods and text processing was conducted. The necessity of developing new methods for detecting fake news to improve accuracy and efficiency in the context of the ever-increasing volume of information is substantiated.

Models for text classification and data processing algorithms were developed. Using modern machine learning technologies, models were created that allow for high-accuracy identification of disinformation.

Based on the conducted research, software was developed that includes modules for text classification, execution flow processing, and normalization of input data. Testing confirmed the reliability of theoretical research and the effectiveness of the implemented algorithms.

The software modules developed in the project can be used by research groups and news portals for the analysis and study of the spread of fake information.

Keywords: fake news, social networks, machine learning, text processing, software.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Зі зростанням впливу соціальних мереж як основного джерела інформації для багатьох людей, поширення фейкових новин стає серйозною загрозою для суспільства та демократичних процесів[1]. Потенційно шкідлива інформація може впливати на громадську думку, політичні вибори, економічні ринки та інші аспекти суспільного життя. Існує необхідність в розробці та впровадженні ефективних інструментів для автоматичного виявлення фейкових новин у соціальних мережах. Це важлива задача, оскільки потрібно забезпечити достовірність інформації, яка доходить до користувачів, та підтримати надійність соціальних мереж як засобу комунікації.

Сучасні технології, які дозволяють залишатися анонімним в Інтернеті, сприяють поширенню фейкових новин. Тепер будь-хто може створити та поширити дезінформацію з метою отримання прибутку або створення хаосу. Це підтверджується тисячами скарг у соціальних мережах, які переповнені фейковими новинами [2].

Довіра до достовірних джерел новин також знизилась, оскільки люди більше не можуть бути впевненими у правдивості інформації, яку вони отримують. Компанії, волонтерські організації, індивідуали та навіть уряди витрачають ресурси на боротьбу з поширенням фейкових новин, але результати є незадовільними. Країни, що вводять закони проти фейкових новин і загрожують штрафами та навіть ув'язненням, стикаються з проблемою визначення винних. Ці закони часто призводять до криміналізації пересічних громадян, які вірять у фейкові новини та поширюють їх. Це призводить до того, що ті, кого закони мали захищати, стають їх жертвами, і свідчить про те, що проста заборона не є ефективним засобом боротьби з фейковими новинами.

Деякі компанії вкладають значні зусилля щоб знайти людей, які проаналізують кожну новину, що з'являється у соціальних мережах, та оцінюють

її достовірність. Однак цей підхід не завжди ефективний через великі витрати на підтримку великої кількості фахівців і вразливість до упередженості або особистих політичних переконань, що може спричинити неправильну класифікацію реальних, але неприйнятних новин як фейкових.

Волонтери об'єднуються, щоб створити бази даних із сайтів, на яких публікуються фейкові новини, з метою фільтрації такої інформації. Однак цей підхід зіткнувся з проблемою в тому, що створення сотень таких сайтів стає легко доступним, тому бази даних завжди будуть відстаючими за часом і можуть бути застарілими на момент їхньої публікації.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є створення програмного забезпечення, яке здатне автоматично виявляти фейкові новини на основі різноманітних аналітичних методів та алгоритмів машинного навчання. Розробка такої програми може допомогти виявляти та фільтрувати недостовірну інформацію, забезпечуючи користувачів достовірними джерелами новин і сприяючи зміцненню довіри до інформаційного простору.

Основними задачами дослідження є:

- провести аналіз стану технологій аналізу інформації;
- розробити інтерфейс програмного застосунку;
- розробити алгоритми для ефективної обробки великого обсягу даних з соціальних мереж;
- розробити модуль класифікації тексту;
- створити модель машинного навчання для автоматичної класифікації новин на реальні та фейкові заснованої на їх характеристиках;
- провести тестування розробленого програмного забезпечення.

Об’єкт дослідження. Об’єктом дослідження є процес виявлення та фільтрації фейкових новин в інтернеті та соціальних мережах.

Ключові етапи дослідження включають аналіз та класифікацію типових характеристик фейкових новин, розробку алгоритмів для автоматичного виявлення цих характеристик у текстових контентах, створення та налаштування програмного забезпечення на основі накопичених знань та алгоритмів.

Предмет дослідження. Предметом дослідження є розробка програмного забезпечення для виявлення фейкових новин, яке базується на аналізі тексту та застосуванні методів машинного навчання.

Методи дослідження. У процесі досліджень використовувались: методи машинного навчання, зокрема лінійний класифікатор та метод наївного Байєса; алгоритми обробки тексту для аналізу змісту новинних статей; методи багатовимірного аналізу для роботи з великими масивами даних; алгоритми текстової аналітики для перевірки ефективності та точності розроблених методів.

Новизна отриманих результатів. Запропоновано новий підхід до автоматичного виявлення фейкових новин у соціальних мережах, що поєднує методи обробки тексту та машинного навчання, зокрема лінійний класифікатор та метод наївного Байєса. Розроблене програмне забезпечення включає модулі для класифікації тексту, обробки потоків виконання та нормалізації вхідних даних. Це дозволяє підвищити точність та ефективність виявлення дезінформації, адаптуючись до нових типів фейкових новин і забезпечуючи користувачам надійну перевірку контенту в режимі реального часу.

Практична цінність отриманих результатів полягає у створенні ефективного програмного забезпечення для автоматичного виявлення фейкових новин у соціальних мережах. Розроблені алгоритми та модулі можуть бути інтегровані в існуючі платформи для моніторингу та фільтрації контенту, що дозволить значно знизити поширення дезінформації. Це, у свою чергу,

сприятиме підвищенню якості інформації, доступної користувачам, і зменшенню негативного впливу фейкових новин на суспільство.

Особистий внесок здобувача. Весь внесок у цю роботу був зроблений особисто автором, включаючи ідею проекту, його проектування, розробку, тестування, а також аналіз потреб користувачів і ринку для забезпечення максимальної ефективності та корисності продукту. Автор самостійно визначив методи та підходи, розробив алгоритми і програмне забезпечення, а також провів усі необхідні експерименти та тестування, що підтвердили працездатність та надійність розробленого рішення.

Структура та обсяг роботи. Пояснювальна записка до бакалаврської дипломної роботи складається з чотирьох розділів.

У першому розділі проаналізовано розвиток технологій аналізу інформації з використанням машинного навчання та порівняльний аналіз існуючих програм, спрямованих на виявлення фейкових новин. Також обговорено методи розв'язання завдань проєкту та обґрунтовано їх вибір, що допомагає визначити оптимальний функціонал створюваного програмного застосунку.

У другому розділі розглянуто завдання та сформульовано вхідні дані для обраного методу розв'язання. Це дозволяє визначити архітектуру програмної розробки, розробити моделі, структури та алгоритми роботи програмних модулів. Також описано інтерфейс користувача з використанням сучасних дизайнерських підходів.

У третьому розділі проведено варіантний аналіз та обґрунтовано вибір засобів реалізації програмного продукту. Розроблено програмні модулі з описом використаних технологій розробки та діаграмами.

У четвертому розділі описано метод тестування програмного застосунку та представлені результати тестування усіх режимів роботи програми. Також описано інструкцію користувача з детальними покроковими інструкціями щодо використання програмного забезпечення.

Апробація результатів здійснена на конференції XXIV Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів (2024) і опубліковані в тезах доповіді цієї конференції [3].

Пояснювальна записка містить 52 сторінок тексту, 29 рисунків, 2 таблиці, 4 додатки, 21 найменування в списку використаних джерел.

1 АНАЛІЗ РОЗВИТКУ ТЕХНОЛОГІЙ АНАЛІЗУ ІНФОРМАЦІЇ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану технологій аналізу інформації з використанням машинного навчання

В сучасному інформаційному середовищі велика увага приділяється розробці та застосуванню методів машинного навчання для аналізу текстової інформації. Машинне навчання використовується для автоматичного виявлення патернів, характерних для фейкових новин, а також для побудови моделей, що дозволяють класифікувати новини як достовірні або недостовірні.

Одним з основних методів машинного навчання, що застосовується в аналізі тексту, є навчання з учителем. Цей підхід передбачає навчання моделі на основі великої кількості прикладів, де для кожної новини визначено, чи є вона фейковою. Популярними алгоритмами для класифікації тексту є метод опорних векторів (SVM), наївний Баєсів класифікатор, а також нейронні мережі.

Крім того, для аналізу тексту використовуються методи обробки природної мови (NLP), що дозволяють витягувати ключові слова та фрази, визначати семантичну схожість між текстами, а також аналізувати сентимент тексту.

Важливим аспектом аналізу тексту є робота з великими обсягами даних. Для ефективної обробки великої кількості текстових даних використовуються розподілені системи обробки даних, такі як Apache, Hadoop або Spark.

У контексті аналізу інформації з використанням машинного навчання виникають також питання етики та безпеки. Наприклад, існує ризик використання алгоритмів для маніпулювання громадською думкою або для збільшення поширення фейкових новин. Також важливо враховувати проблеми з анонімністю та конфіденційністю даних при обробці особистої інформації користувачів.

Крім того, зростаюча складність інформаційного ландшафту вимагає розвитку нових підходів до аналізу та класифікації даних. Одним із таких підходів є глибоке навчання (deep learning), яке дозволяє побудувати складні нейронні мережі для автоматичного виявлення взаємозв'язків у великих обсягах даних. Використання глибокого навчання може покращити точність класифікації тексту та зробити систему більш ефективною в роботі з новими, раніше невідомими патернами фейкових новин.

Значний інтерес також викликає застосування алгоритмів аналізу зображень для виявлення фейкових новин, які містять фотографії або відео. Методи комп'ютерного зору та глибокого навчання можуть виявити аномалії у візуальних зображеннях, що можуть свідчити про маніпуляції чи фотомонтаж. Це може бути особливо корисним у контексті виявлення фото- та відеофейків, які можуть бути поширені з метою дезінформації.

Отже розвиток технологій аналізу інформації з використанням машинного навчання відкриває широкі можливості для боротьби з поширенням фейкових новин. Використання новітніх методів та алгоритмів дозволить побудувати ефективні інструменти для виявлення та фільтрації дезінформації в інтернеті, сприяючи підвищенню довіри до інформаційного середовища та забезпечуючи користувачів достовірною інформацією.

1.2 Порівняльний аналіз аналогів

В інформаційному суспільстві, насиченому потоком інформації, проблема розповсюдження фейкових новин стає все більш актуальною та складною. Сучасні технології дозволяють створювати та поширювати дезінформацію з непередбачуваною швидкістю та легкістю, що підриває довіру до інформаційних джерел та загрожує стабільності суспільства. У цьому контексті розробка ефективних інструментів для виявлення та фільтрації фейкових новин стає надзвичайно важливою задачею. Для цього необхідно ретельно проаналізувати

наявні методи та технології, що застосовуються в цій сфері, зокрема з використанням машинного навчання, та виявити їхні переваги та недоліки.

Більшість існуючих рішень мають кілька недоліків, які ускладнюють або роблять незручними їх щоденне використання для різних категорій користувачів. Деякі рішення обмежені лише до використання на конкретній платформі, що ускладнює перевірку новин з різних джерел. Інші варіанти спираються на добровільну діяльність користувачів та їх власну оцінку, що може призвести до упереджених і неточних результатів.

У Facebook існує програма для виявлення фейкових новин (рис. 1.1), яка базується на отриманні зворотного зв'язку від користувачів. Вона називається Facebook Fact-Checking [4]. Ця програма направляє новини на перевірку до фахівців, які перевіряють новину на дійсність. Якщо фахівці визначають, що новина є неправдивою, вона приховується від користувачів, а групи, що її розповсюджують, піддаються контролю. Однак є кілька недоліків цього підходу: він залежить від зворотного зв'язку користувачів, потребує постійної підтримки великою кількістю фахівців для перевірки зростаючої кількості новин, і обмежений лише однією соціальною мережею, що обмежує його використання.

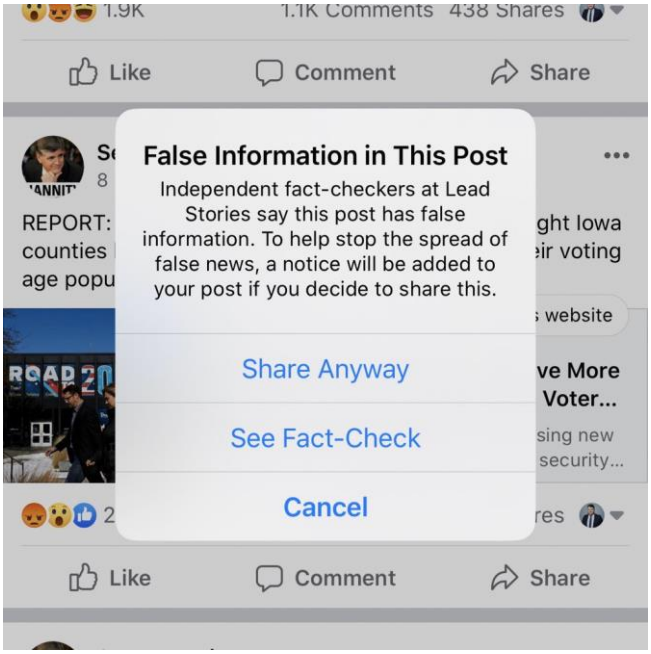

Рисунок 1.1 – Робота програми Facebook Fact-Checking

Розширення для браузера Chrome під назвою «Fake news debunker by InVID & WeVerify» (рис. 1.2) спрямоване на боротьбу з фейковими новинами [5]. Воно розроблене спільними зусиллями InVID і WeVerify, спрямованими на підвищення рівня безпеки та боротьби з дезінформацією в мережі. Це розширення пропонує користувачам інструменти для перевірки достовірності відео та зображень, які вони зустрічають під час перегляду веб-сторінок. Воно надає можливість перевіряти джерела та контекст зображень та відео, що допомагає відокремити правдиву інформацію від маніпуляцій та фейкових матеріалів.

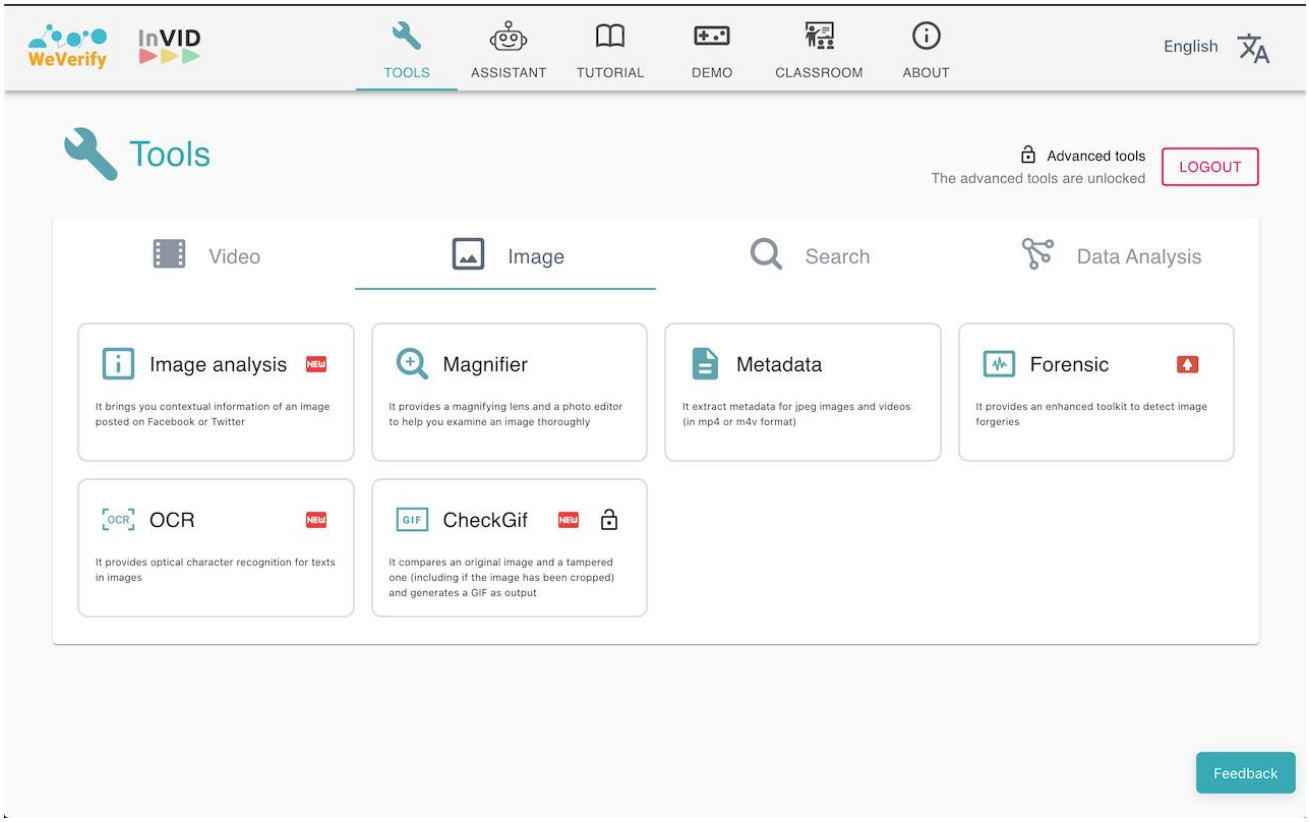

Рисунок 1.2 – Інтерфейс розширення Fake news debunker

Основні функції цього розширення включають ідентифікацію дати та місця зйомки зображень або відео, пошук схожих зображень у мережі та перевірку їх автентичності, а також перевірку метаданих відео для виявлення можливих змін або монтажу. Використання цього розширення допомагає

користувачам бути більш обізнаними та обережними під час споживання вмісту в Інтернеті, зменшуючи вплив фейкових новин і дезінформації на їх рішення та думки.

Один з головних недоліків цього розширення – воно доступне лише для користувачів браузера Chrome. Також, воно не дозволяє перевіряти правдивість текстових новин, лише аналізує зображення та відео.

В Україні запустили фактчек-бот (рис. 1.3) для відстеження фейків. Зазначається, що «Перевірка» є ботом-рятівником від фейкових новин, неперевіреної інформації, відвертої брехні та російської пропаганди [6].

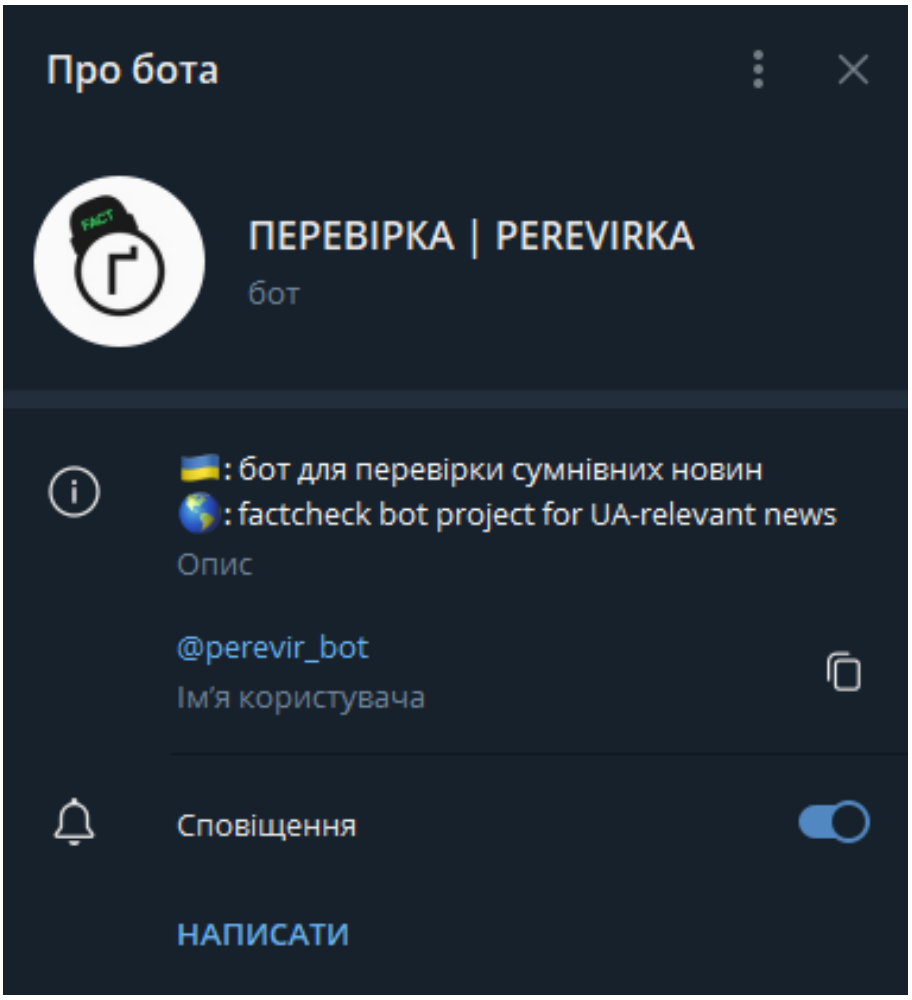

Рисунок 1.3 – бот «Перевірка» у соціальній мережі Telegram

Зокрема за допомогою цього боту можна розрізнити фейкові новини в соцмережах, у політиці, визначити фейкові новини російських засобів масової інформації, спрямовані на боротьбу з Україною. Щоб розпізнати фейк, необхідно надіслати боту інформацію, яку потрібно перевірити на справжність. Надіслане посилання відразу потрапляє команді, яка вміє швидко перевіряти великі масиви інформації із застосуванням штучного інтелекту. Після надсилання новини відповідь надійде протягом кількох хвилин.

Розробка програмного забезпечення для виявлення фейкових новин потребує ретельного планування та розробки. Це включає в себе необхідність володіння навичками програмування, знання алгоритмів обробки тексту та аналізу даних. Передбачається створення алгоритмів, які дозволять автоматично аналізувати текст новин та виявляти ознаки фейкових матеріалів.

Основною частиною розробки є створення моделі машинного навчання, яка буде навчатися розпізнавати та класифікувати новини на основі різних параметрів, таких як використання емоційних слів, наявність фактів та їх перевірка, а також аналіз стилю письма.

Результати порівняння аналогів зведено в таблицю 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерії	Fake news debunker	Facebook Fact-Checking	Бот "Перевірка" в Телеграмі	Розроблювана програма
Машинне навчання	-	-	-	+
Аналіз тексту	-	+	+	+
Доступність для користувачів	-	-	+	+
Швидкість аналізу	+	-	+	+

Продовження до таблиці 1.1

Критерії	Fake news debunker	Facebook Fact-Checking	Бот "Перевірка" в Телеграмі	Розроблювана програма
Розширені можливості	+	-	-	+
Загальна оцінка	40%	20%	60%	100%

Усі згадані рішення мають свої обмеження, які ускладнюють їх широке застосування. Робота над дипломною роботою може вирішити багато проблем, з якими інші рішення, які доволі часто покладаються на реальних людей для того, щоб аналізувати кожну новину, тоді, коли було б набагато ефективніше навчити це робити машину, яка б могла вчитися на неправильних результатах та не була б упередженою.

1.3 Аналіз методів розв’язання задачі

Вирішення проблеми в машинному навчанні часто ускладнюється пошуком відповідного інструменту для оцінки роботи. Різні методи оцінки можуть бути більш або менш ефективними в залежності від типу даних та конкретної проблеми. Головна задача програми – виявлення фейкових новин. Для цього ми маємо вибірку з більше ніж 50 прикладами, яку можна легко отримати, оскільки існує багато доступних відкритих даних. Зазвичай ми маємо дві категорії міток для виявлення фейкових новин: справжні та фейкові. Кожен приклад вибірки має відповідну мітку для визначення категорії.

Якщо наша вибірка має більше ніж 100 000 зразків, найкращим застосуванням буде алгоритм лінійної моделі зі стохастичним градієнтним спуском. У разі незадовільних результатів застосуємо метод апроксимації ядра.

Якщо розмір вибірки менше ніж 100 000 зразків, використовуватимемо алгоритм класифікації методом лінійних опорних векторів. У разі невдачі

можемо використати метод наївного Байєса для текстових даних або метод найближчих сусідів для зображень. Якщо навіть ці методи не працюють, можемо використати метод С-підтримки векторної класифікації або ансамблеві методи.

Метод лінійних опорних векторів є одним із найпопулярніших алгоритмів машинного навчання з вчителем [7]. Він застосовується для класифікації та регресії. Основна мета полягає в тому, щоб знайти найкращу лінію або межу рішення, яка може розділити простір даних на класи. Це дозволяє легко визначити категорію для нової точки даних у майбутньому. Межа рішення, яка найкраще розділяє дані, називається гіперплощиною. Метод лінійних опорних векторів обирає екстремальні точки або вектори, які допомагають побудувати цю гіперплощину. Ці екстремальні точки називаються опорними векторами. На рисунку 1.4 наведено діаграму цього методу.

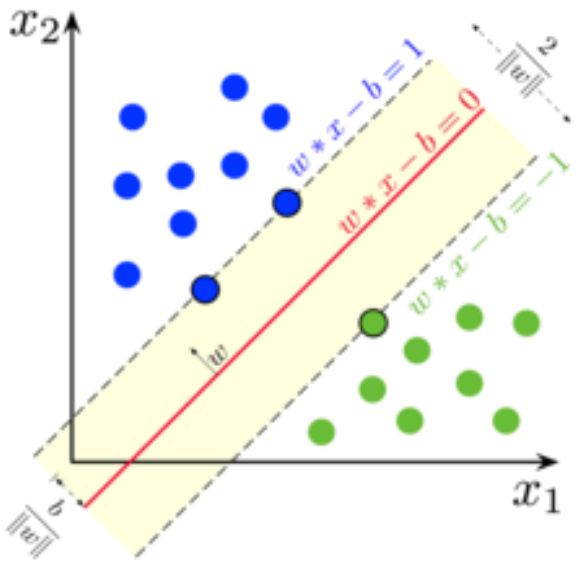

Рисунок 1.4 – Діаграма методу лінійних опорних векторів

Ще один підхід полягає в використанні алгоритму наївного Байєса. Наївний Байєс – це група контрольованих алгоритмів класифікації машинного навчання на основі теореми Байєса.

Це проста методика класифікації, але має високу функціональність. Вони знаходять застосування, коли розмірність вхідних даних висока. Складні задачі

класифікації також можуть бути реалізовані за допомогою наївного байєсового класифікатора [8].

Під час роботи з безперервними даними часто прийнято вважати, що безперервні значення, пов’язані з кожним класом, розподіляються відповідно до нормального розподілу. Гаусівський наївний Байєс підтримує безперервні ознаки та моделі, кожна з яких відповідає нормальному розподілу. Підхід до створення простої моделі полягає в припущенні, що дані описуються гаусівським розподілом без коваріації (незалежних розмірів) між вимірами. Ця модель може бути підібрана, просто знайшовши середнє значення та стандартне відхилення точок у кожній мітці, що є все, що потрібно для визначення такого розподілу.

Розглянувши переваги та недоліки кожного методу, було вирішено вибрати найефективніші методи для розробки програмного застосунку було прийнято рішення використовувати класифікатори лінійного типу та наївного Байєса.

1.4 Постановка задач дипломної роботи

Проаналізувавши переваги та недоліки існуючих програмних засобів для виявлення фейкових новин у соціальних мережах, було визначено наступні завдання, які необхідно виконати для успішної розробки програмного забезпечення:

- провести аналіз стану технологій аналізу інформації;
- розробити інтерфейс програмного застосунку;
- розробити алгоритми для ефективної обробки великого обсягу даних з соціальних мереж;
- розробити модуль класифікації тексту;
- створити модель машинного навчання для автоматичної класифікації новин на реальні та фейкові заснованої на їх характеристиках;
- провести тестування розробленого програмного забезпечення.

1.5 Висновки

В розділі було досліджено предметну область виявлення фейкових новин в соціальних мережах. Були розглянуті програмні засоби такі як Fake news debunker, Facebook Fact-Checking, бот "Перевірка". Кожен з цих засобів має свої переваги і недоліки, що були проаналізовані.

Аналіз дозволив вибрати найефективніші методи для розробки програмного застосунку. Так як більшість відкритих баз даних з навчальними даними надають

значно менше ніж 100,000 записів у одній базі та з огляду на те, що класифікація тексту включає в себе існування помічених даних, було прийнято рішення використовувати класифікатори лінійного типу та наївного Байєса.

2 РОЗРОБКА МОДЕЛІ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз даних для розробки програм виявлення фейкових новин в соціальних мережах

Для реалізації програмного застосунку для виявлення фейкових новин в соціальних мережах використано мову програмування Python а також алгоритми лінійних класифікаторів та наївного Байєса.

Python [9] – це мова високого рівня програмування, яку називають другою за популярністю в світі. Вона відома своєю простотою та читабельністю синтаксису, що робить її дуже популярною серед початківців та досвідчених програмістів. Python підтримує різні парадигми програмування, такі як процедурне, об'єктно-орієнтоване та функціональне програмування. Вона має велику стандартну бібліотеку, яка включає різноманітні корисні функції та інструменти для роботи з різними типами даних, мережевими протоколами, файловою системою тощо. Її використовують для розробки веб-застосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та кросплатформеність.

Існує багато алгоритмів побудованих на основі цих типів, тому в програмному застосунку буде реалізація чотирьох з них – два лінійного типу, та два типу наївного Байєса.

Створений програмний застосунок використовує декілька додаткових бібліотек для виконання своїх функцій.

По-перше, для аналізу новин вирішено використовувати бібліотеку scikit-learn. Це інструмент машинного навчання для Python, який надає прості та ефективні засоби для прогнозування та аналізу даних. Scikit-learn побудована на базі інших популярних бібліотек, таких як NumPy, SciPy та matplotlib, і доступна

для використання у різних сферах діяльності [10]. Ця бібліотека пропонує різноманітні функції для роботи з даними, включаючи класифікацію, регресію, кластеризацію, зменшення розмірності, вибір моделей та попередню обробку даних.

По-друге, для перетворення необроблених табличних даних у формат, придатний для аналізу за допомогою scikit-learn, була використана бібліотека pandas. Pandas - це інструмент для аналізу та маніпулювання даними з відкритим вихідним кодом, який забезпечує швидке, потужне та просте використання. Він побудований на мові програмування Python і надає гнучкі можливості для роботи з табличними даними [11].

По-третє, для забезпечення можливості зберігання та експортування створених моделей вирішено використати вбудований модуль pickle. Цей модуль реалізує двійкові протоколи для серіалізації та десеріалізації об'єктної структури Python [12]. Процес перетворення ієрархії об'єктів Python у потік байтів відомий як «піклінг», а зворотна операція, коли потік байтів перетворюється назад у ієрархію об'єктів, – «анпіклінг». Модуль pickle дозволяє зберігати більшість підтримуваних структур даних Python на жорсткому диску з можливістю подальшого використання та розповсюдження.

По-четверте, для забезпечення безперервної роботи застосунку буде використано модуль threading. Цей модуль дозволяє створювати потоки виконання, що дозволяє виділити роботу графічного інтерфейсу та функцій, що вимагають тривалого обчислення, у окремі потоки [13]. Це забезпечує зворотній зв'язок графічного інтерфейсу на введення користувача, тим часом як функції продовжують виконуватися.

Розробка цих програмних модулів потребує глибокого розуміння у програмуванні на Python. Для реалізації цих модулів будуть використані різноманітні інструменти, зокрема інтегроване середовище розробки PyCharm. Буде проведено тестування програмних модулів для забезпечення точності та швидкості виявлення фейкових новин в соціальних мережах.

2.2 Розробка інтерфейсу програмного застосунку

Існує декілька популярних бібліотек для створення графічного інтерфейсу, і переважна більшість з них може бути використана з Python. Під час розробки інтерфейсу програми було використано бібліотеку Tkinter [14].

Бібліотека Tkinter є стандартним інструментом для роботи з графічним інтерфейсом користувача (GUI) у Python. Вона базується на наборі інструментів Tk, який є одним з найпоширеніших інструментів для створення GUI. Tkinter надає зручний інтерфейс для створення вікон, кнопок, текстових полів, меню, та інших елементів інтерфейсу.

Основним кольором інтерфейсу обрано білий. На рисунку 2.1 зображено головний екран застосунку.

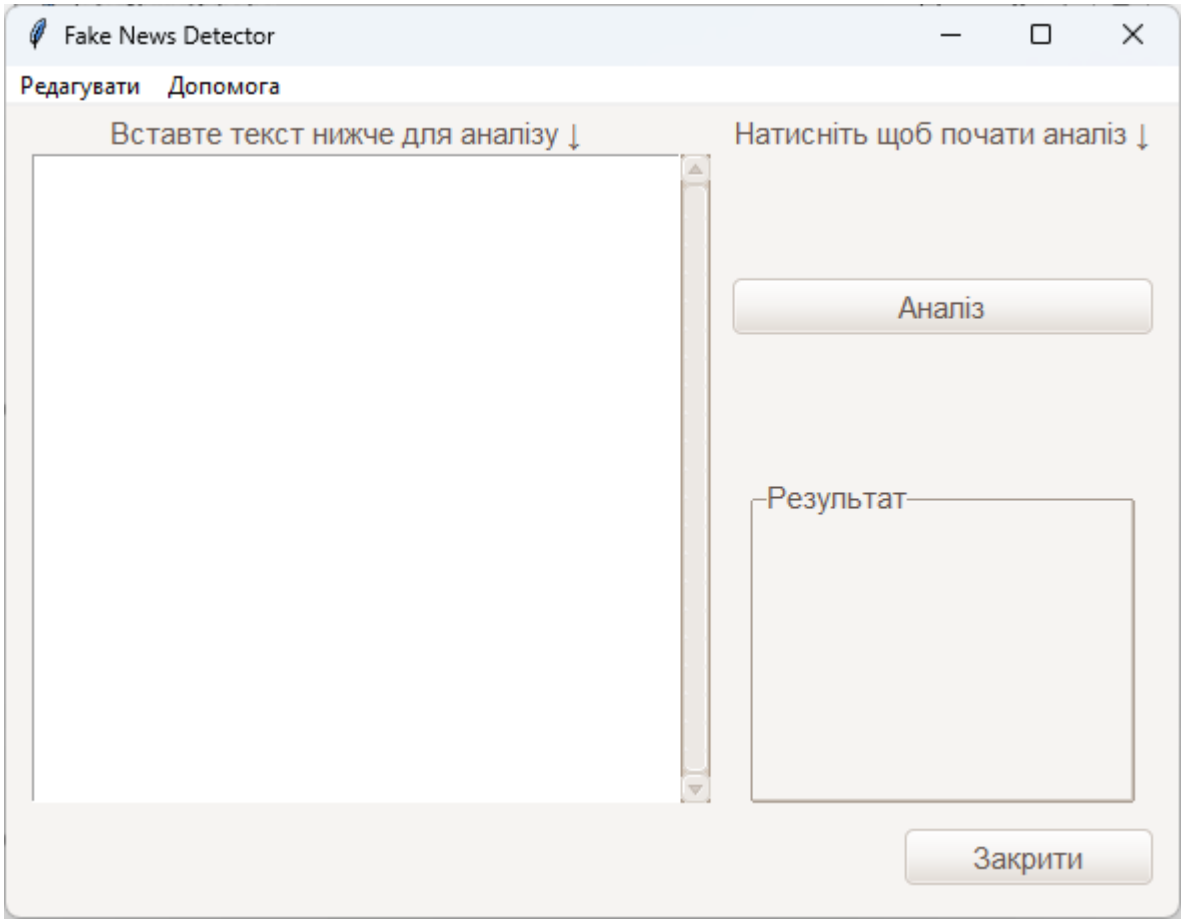

Рисунок 2.1 – Головне вікно програми

Також було розроблено меню програмного застосунку (рис. 2.2) для простого користування додатком.

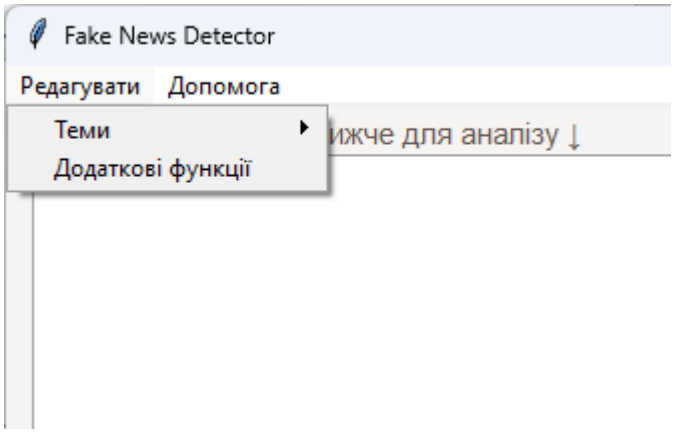

Рисунок 2.2 – Меню програмного застосунку

Якщо у пункті меню «Редагувати», обрати додаткові функції, вікно програмного застосунку розшириться, де будуть додаткові функції керування програмою, для машинного навчання. Вікно розширеного застосунку зображено на рисунку 2.3.

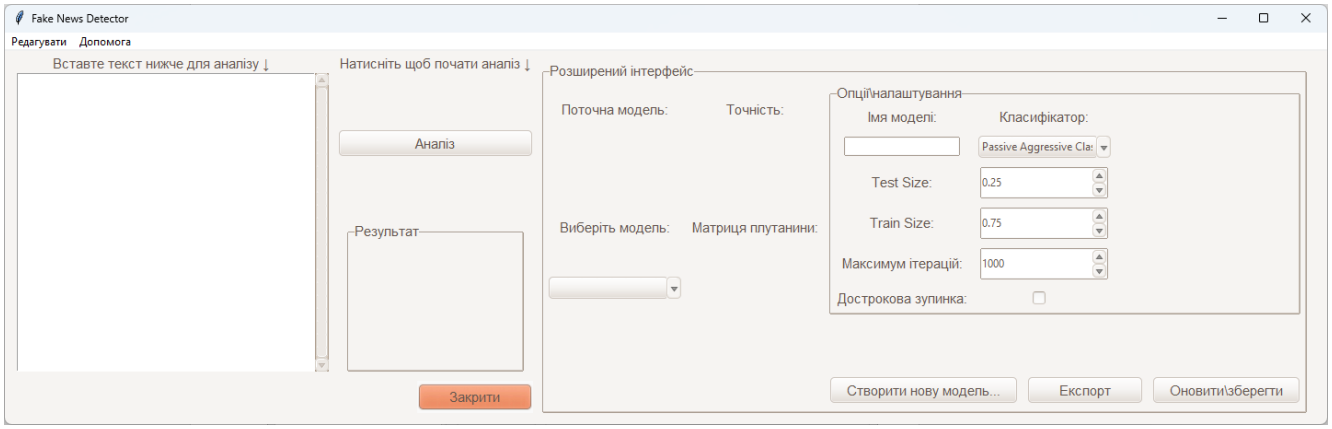

Рисунок – 2.3 – Вікно з додатковими функціями

У пункті «Редагувати» було додано можливість змінювати тему інтерфейсу за допомогою випадаючого списку «Теми» (рис. 2.4). Користувач має

можливість вибрати бажаний стиль інтерфейсу з доступних йому опцій. Після вибору стилю, зміни відразу відображаються на головному вікні програми, щоб користувач міг оцінити вигляд і вибрати оптимальний стиль для своїх потреб. Ця функція дозволяє користувачам налаштовувати інтерфейс програми згідно їхніх вподобань та стандартів дизайну.

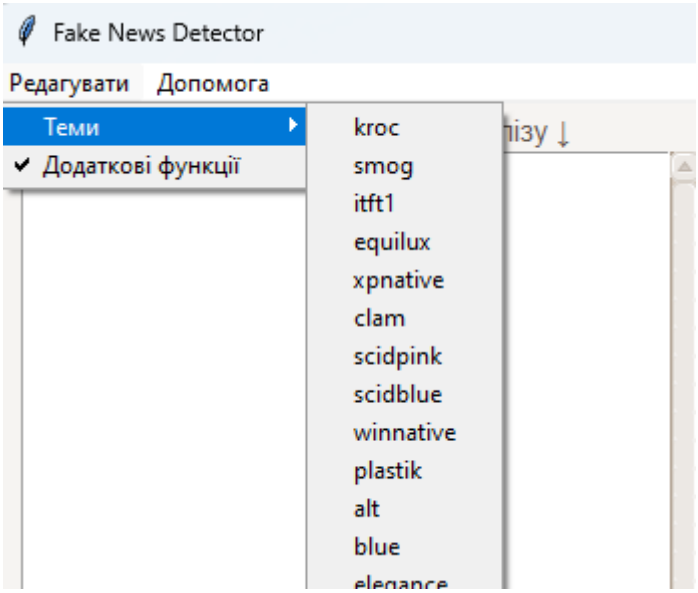

Рисунок 2.4 – Можливість вибрати стиль інтерфейсу

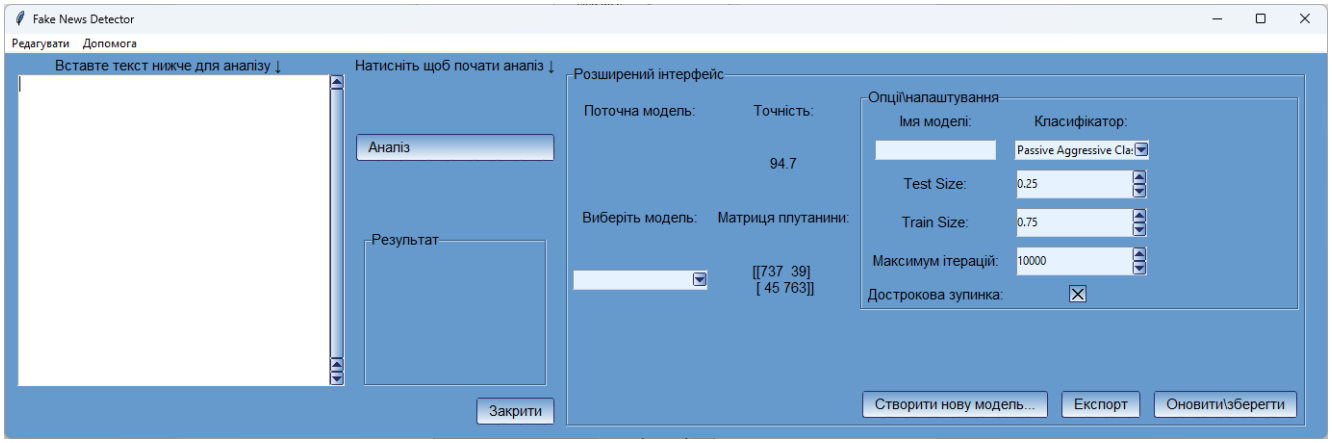

Рисунок 2.5 – Змінена тема інтерфейсу

Також можна відкрити вікно «Про програму» де буде інформація про розробника та мову програмування додатку. Зображення вікна зображено на рисунку 2.6.

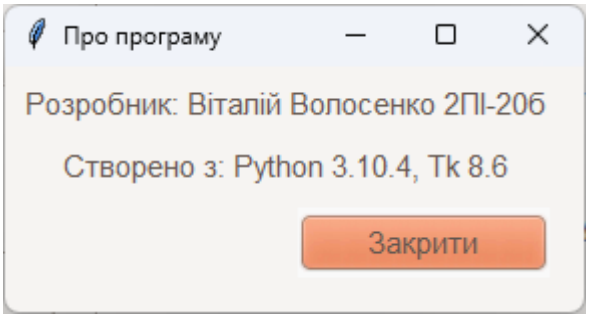

Рисунок 2.6 – Вікно «Про програму»

Інтерфейс було розроблено за допомогою бібліотеки Tkinter у середовищі PyCharm з використанням мови програмування Python. Використання Tkinter у середовищі PyCharm дозволило зручно розробляти та налагоджувати графічний інтерфейс для програми.

2.3 Розробка моделі системи

Для кращого розуміння роботи програмного застосунку для виявлення фейкових новин, вирішено створити модель функціонування системи з використанням UML діаграми діяльності.

Згідно з вказаною діаграмою, користувач для початку взаємодії із програмою повинен вибрати модель яка буде аналізувати новини. У випадку, якщо моделі немає, програма повідомить про це та попросить створити нову модель. Для створення нової моделі користувачу необхідно обрати навчальні дані у форматі csv. Якщо файл дійсний, модель почне навчатися за обраним користувачем алгоритмом, аналізуючи ці дані, в іншому випадку програма повідомить про помилку. Коли програма закінчить навчання моделі, стане можливий аналіз текстових новин. Після вибору новини, програма проводить аналіз тексту за допомогою алгоритмів машинного навчання та обробки

природної мови. Якщо результати аналізу вказують на високу ймовірність фейковості новини, програма відзначає її як потенційно фейкову та надає користувачеві відповідне повідомлення. У разі успішного аналізу та відсутності підозрілих ознак, програма повідомляє користувача про те що новина правдива. Крім того, користувач має можливість змінити налаштування і перенавчити модель на інший алгоритм. Після завершення навчання моделі, користувач може експортувати її в формат pickle в любе місце на жорсткому диску для подальшого використання.

На рисунку 2.7 зображено UML діаграму діяльності програмного застосунку.

Рисунок 2.7 – Діаграма діяльності програмного застосунку

Діаграма діяльності була розроблена за допомогою веб-застосунку Drawio.

2.4 Розробка алгоритмів роботи системи

Для розробки програмного застосунку, для виявлення фейкових новин в соціальних мережах необхідно розробити загальний алгоритм, згідно якого буде працювати програмний модуль (рис. 2.8).

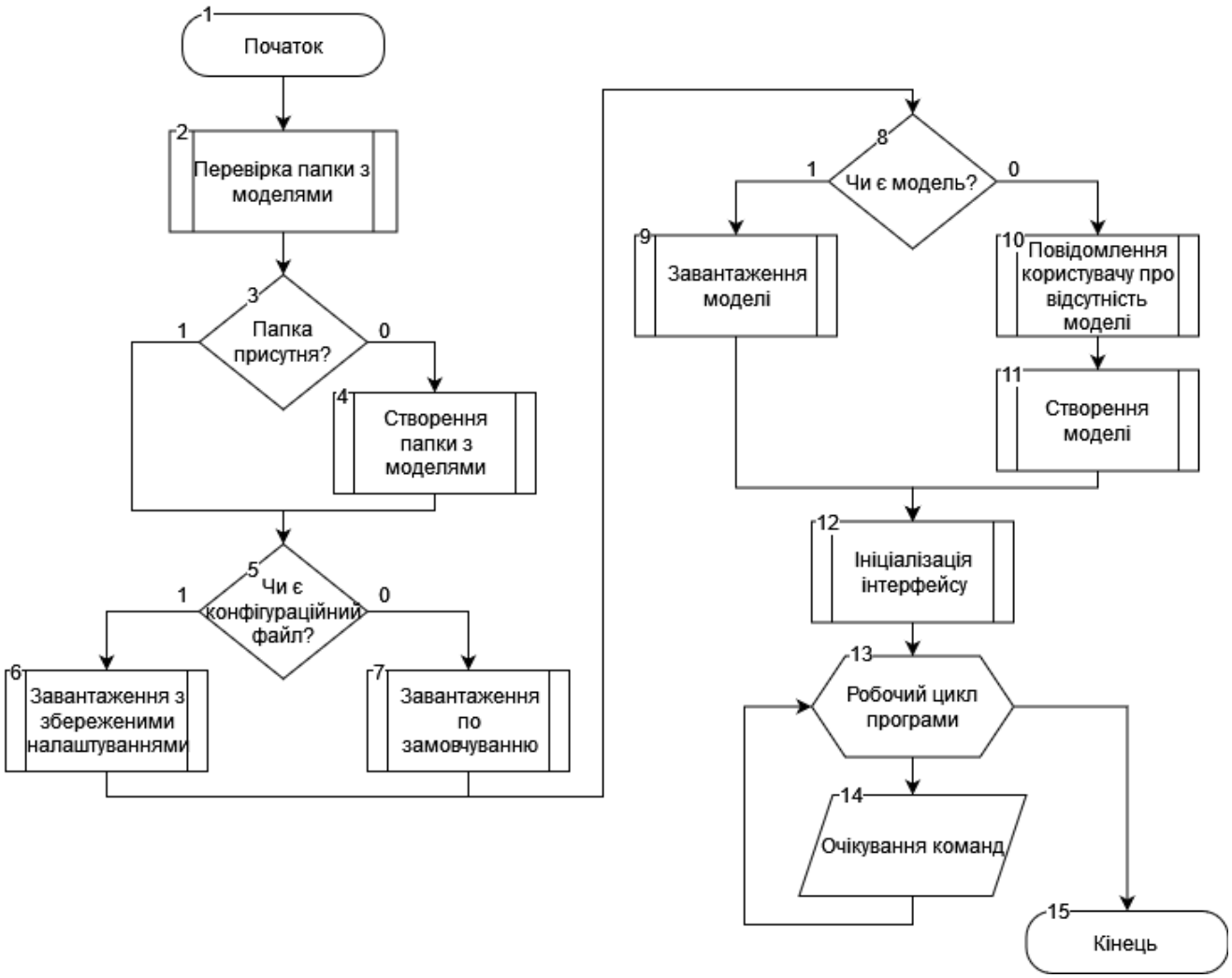

Рисунок 2.8 – Загальний алгоритм програмного застосунку

При кожному запуску програми відбувається кілька кроків. Спочатку система перевіряє наявність папки з моделями в поточному місці знаходження

програми. Якщо така папка вже існує, виконання програми продовжується безперервно. У випадку, якщо папка відсутня, програма автоматично створює її, а потім продовжує виконання.

Після цього, система перевіряє наявність конфігураційного файлу з налаштуваннями користувача. Якщо такий файл існує, програма продовжує завантаження з врахуванням збережених налаштувань. У випадку відсутності файлу, будуть використані значення за замовчуванням.

Після цього проводиться аналіз доступних тем для користувача та ініціалізація інтерфейсу. Після успішного завершення цих кроків програма повністю завантажується і розпочинається її стандартний робочий цикл.

Періодично програма оновлює графічний інтерфейс і перевіряє наявність команд користувача у черзі команд. У випадку відсутності команд цикл повторюється, але якщо в черзі з'явиться команда, виконується відповідна дія в залежності від типу команди. Якщо отримана команда є командою закриття програми, то вона закривається.

Якщо отримана команда є командою створення нової моделі, то спочатку створюється новий потік даних, який дозволяє виконувати створення моделі, а в той же час, продовжувати користування програмою. В іншому випадку графічний інтерфейс не буде реагувати на команди користувача, поки не завершиться виконання попередньої команди. У разі, якщо команда користувача є командою зміни налаштувань, то спочатку змінюються обрані налаштування, а потім вони зберігаються поверх старих у файлі конфігурації. Після цього цикл повторюється. У всіх інших випадках виконується команда та повертається до початку циклу.

Коли інтерфейс програми завантажився, користувач має можливість вставляти текст новин у відповідне поле введення. Після цього він може запустити процес аналізу, натиснувши відповідну кнопку. Програма обробляє введений текст згідно алгоритму, що визначає, чи є новина фейковою чи правдивою. Алгоритм визначення фейковості новин зображено на рисунку 2.9.

Для того щоб зробити процес більш зручним, інтерфейс програми забезпечує користувача зворотнім зв'язком про стан обробки тексту. Наприклад, після натискання кнопки для аналізу, користувач може бачити індикатор завантаження, який показує, що обробка триває. Це допомагає уникнути непорозумінь та забезпечує більш комфортний користувацький досвід.

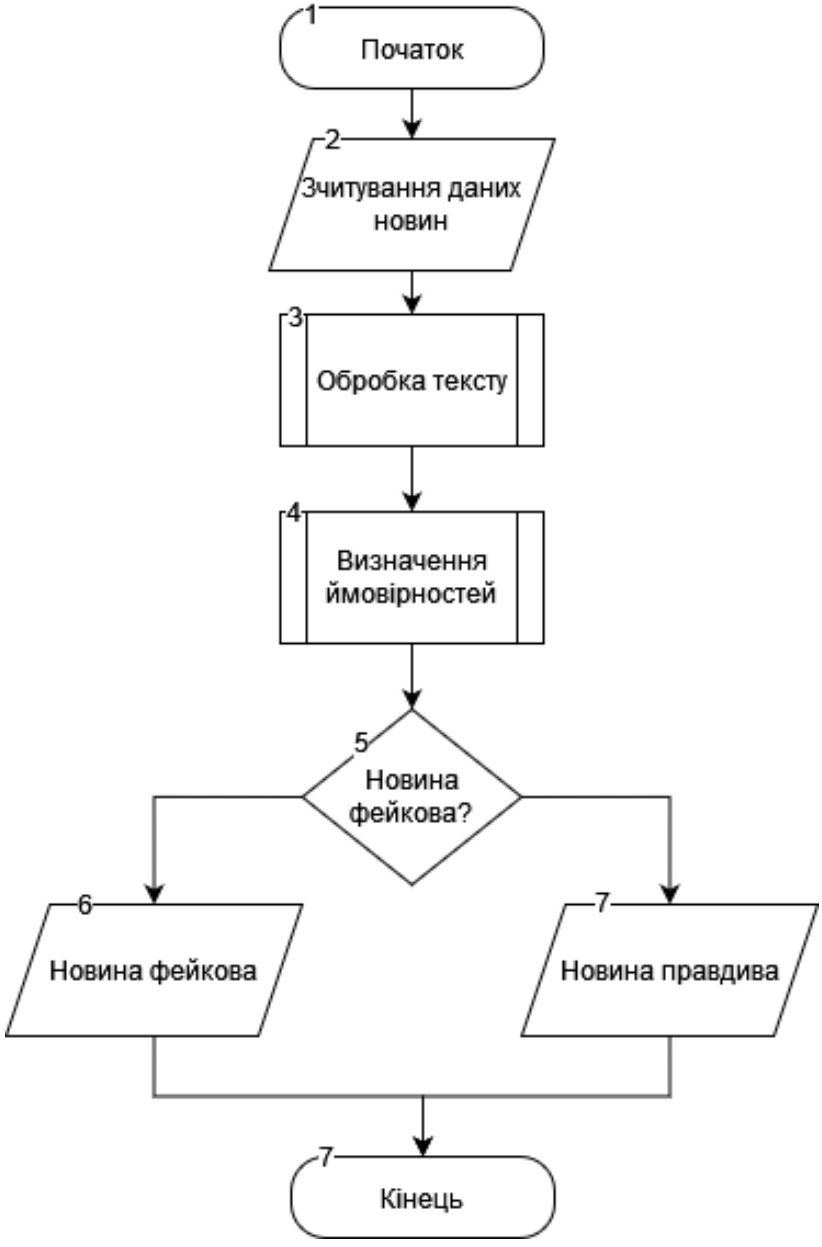

Рисунок 2.9 – Алгоритм визначення фейкових новин

Алгоритм, зображений на рисунку 2.9, включає кілька важливих етапів. Спершу відбувається попередня обробка тексту, де видаляються зайві символи

та структуруються дані. Після цього текст аналізується на наявність специфічних маркерів, які можуть вказувати на фейковість новини. Ці маркери можуть включати неправильні факти, упереджені висловлювання, відсутність достовірних джерел та інше.

Наступним етапом алгоритму є оцінка вірогідності новини за допомогою моделі машинного навчання. Модель була навчена на великій кількості навчальних даних, що дозволяє їй ефективно розрізняти їх. На завершення, програма надає користувачу результат аналізу, вказуючи, наскільки висока ймовірність того, що новина є фейковою.

2.5 Висновки

У другому розділі було проведено аналіз вхідних та вихідних даних програмних засобів, розглянуто процес створення графічного інтерфейсу програмного застосунку. Також розроблено основний алгоритм роботи програми та алгоритм визначення фейкових новин. Крім того, була розроблена модель роботи програмного застосунку за допомогою UML діаграми.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ ВИЗНАЧЕННЯ ФЕЙКОВИХ НОВИН З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Під час розробки програмного забезпечення для виявлення фейкових новин в соціальних мережах з використанням машинного навчання важливо обрати відповідні засоби для розробки кожного модуля. Python – це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об’єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом з його інтерпретованим характером роблять його ідеальною мовою для створення сценаріїв і швидкої розробки додатків у багатьох сферах на більшості платформ.

Tkinter – це стандартна бібліотека мови програмування Python, яка використовується для створення графічного інтерфейсу користувача (GUI). Вона надає набір інструментів і об'єктів, які дозволяють розробникам створювати вікна, кнопки, поля введення, меню, таблиці та інші елементи GUI для взаємодії з користувачем. Tkinter базується на Tk Toolkit, який є одним з найпоширеніших інструментів для створення GUI у світі програмування. Tkinter надає простий та зрозумілий інтерфейс для створення програм з графічним інтерфейсом, що робить її популярним вибором для початківців у розробці програм на Python.

Ttkthemes – яка надає розширення для стандартної бібліотеки tkinter.ttk. Вона дозволяє використовувати різноманітні теми оформлення для створення красивих та сучасних графічних інтерфейсів у програмах, що базуються на Tkinter. За допомогою ttkthemes можна легко налаштовувати вигляд інтерфейсу, обираючи різні стилі та теми оформлення, такі як світлий, темний, матовий тощо. Це дозволяє розробникам створювати інтерфейси, які відповідають сучасним тенденціям дизайну та відповідають потребам користувачів.

Scikit-learn – це бібліотека машинного навчання для мови програмування Python. Вона надає широкий набір інструментів для роботи з різноманітними завданнями машинного навчання, такими як класифікація, регресія, кластеризація, виявлення аномалій, відбір ознак та багато інших. Scikit-learn базується на інших популярних бібліотеках Python, таких як NumPy, SciPy та matplotlib, і забезпечує легкий і простий у використанні інтерфейс для розв'язання складних задач машинного навчання. Вона є однією з найбільш поширених і популярних бібліотек для машинного навчання у спільноті Python [15].

У роботі буде використано такий функціонал:

- sklearn.linear_model.PassiveAggressiveClassifier;
- sklearn.linear_model.Perceptron;
- sklearn.naive_bayes.MultinomialNB;
- sklearn.naive_bayes.ComplementNB.

Scipy – це відкрита бібліотека, яке надає ефективні та потужні інструменти для наукових обчислень та технічного обчислювального моделювання в мові програмування Python [16]. Вона базується на бібліотеці NumPy та надає додатковий функціонал для роботи зі спеціальними функціями, оптимізацією, інтегруванням, обробкою сигналів, обробкою зображень, статистикою, алгебраїчними рівняннями та іншими задачами. Scipy включає в себе широкий спектр підмодулів, які охоплюють різноманітні області наукових обчислень, що дозволяє виконувати складні обчислення та аналізувати дані в наукових, інженерних, математичних та інших областях досліджень. Це робить scipy потужним інструментом для розв'язання різних завдань у сфері наукових досліджень та технічного моделювання.

Joblib – це бібліотека Python, яка надає можливості для ефективного збереження та завантаження результатів обчислень, використовуючи механізми кешування та серіалізації. Основна мета joblib – це забезпечити швидкий та простий спосіб збереження проміжних результатів обчислень, щоб уникнути

повторного обчислення при наступних викликах функцій [17]. Ця бібліотека особливо корисна для завдань машинного навчання та обробки даних, де може бути потрібно зберегти інтермедіатні результати, такі як навчені моделі, вектори ознак, матриці підтримки та інші об'єкти. Joblib дозволяє зберігати ці об'єкти на диску та легко завантажувати їх для подальшого використання без необхідності повторного обчислення.

Отже вибір засобів для розробки програми виявлення фейкових новин в соціальних мережах з використанням машинного навчання включає Python, tkinter, ttkthemes, scikit-learn, scipy, joblib а також двох алгоритмів лінійного типу та двох типу наївного Байєса має вирішальне значення для розробки програмного застосунку [18]. Поєднання цих інструментів та сильних навичок програмування дозволяє створити надійний продукт з найкращим функціоналом та зручністю інтерфейсу користувача.

3.2 Розробка модуля класифікації тексту для визначення фейкових новин

При розробці модуля для визначення фейкових новин важливо враховувати ряд технічних аспектів. Для визначення фейкових новин необхідно використовувати алгоритми машинного навчання, які здатні аналізувати текстові дані та визначати їх достовірність. Також важливо враховувати різноманітні метрики та аналітичні засоби для оцінки результатів класифікації фейкових новин.

У даному модулі використано різні алгоритми машинного навчання такі як наївного Байєса та лінійні класифікатори для класифікації текстових даних на фейкові та нефейкові новини. Основна задача полягатиме в тому, щоб алгоритми здатні були ефективно аналізувати текстові дані та робити правильні прогнози щодо їхньої достовірності [19]. Клас Detector (рис. 3.1), що знаходиться у модулі, містить у собі більшість змінних та методів, які є загальними для усіх наслідуючих класів.

Він ініціалізує модель з параметрами, завантажує дані з CSV-файлу (якщо переданий шлях до файлу), розділяє дані на навчальну та тестову вибірки, виконує векторизацію тексту, навчає класифікатор та робить прогнози на тестових даних.

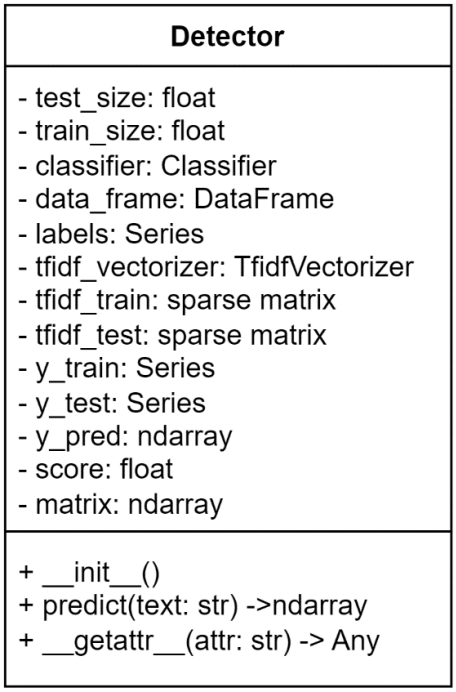

Рисунок 3.1 – Діаграма класу Detector

Класи LinearDetector та BayesDetector, що наслідуються від класу Detector є двома більш спеціалізованими версіями цього класу. Вони не передбачені для прямого використання, але можуть бути корисними під час створення додаткових класифікаторів, виступаючи платформою для них.

Вони призначені для створення моделей класифікації з використанням лінійних та байєсових алгоритмів відповідно (див. рисунки 3.2 – 3.3). Вони розширюють функціональність базового класу Detector, додаючи можливість налаштування додаткових параметрів, таких як максимальна кількість ітерацій для лінійних моделей або розмір вибірок для навчання та тестування. Клас LinearDetector дозволяє створювати моделі класифікації з використанням лінійних класифікаторів, таких як Passive Aggressive Classifier та Perceptron, з

можливістю налаштування параметрів цих моделей. Клас BayesDetector використовує байєсові класифікатори, такі як multinomial naive bayes та complement naive bayes, для створення моделей класифікації тексту.

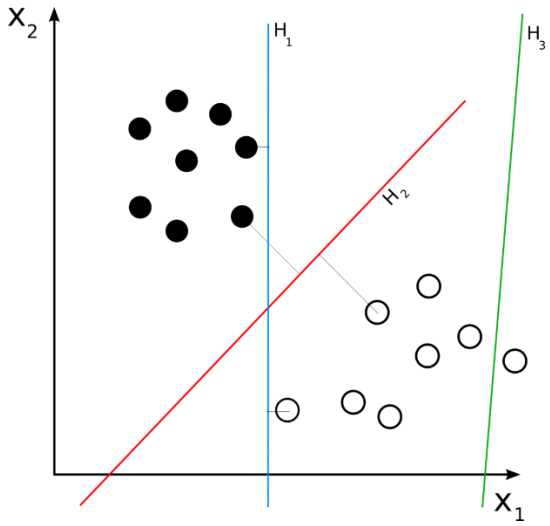

Рисунок 3.2 – Схема лінійного класифікатора

В цьому випадку чорні та білі точки може бути класифіковано нескінченною кількістю лінійних класифікаторів. H1 (синій) класифікує їх правильно, як і H2 (червоний). H2 може вважатися «кращим» у тому сенсі, що він також є як найдальшим від обох груп. H3 (зелений) не здатен правильно класифікувати ці точки.

Рисунок 3.3 – Схема роботи алгоритму наївного Байєса

Також є класи PAClassifier та Percept, що є нащадками LinearDetector та ComplNB з MultiNB, що у свою чергу є нащадками BayesDetector. Ці класі передбачені для використання як класифікатори тексту. Кожен клан реалізує свій вид класифікатора. Дані класи зображені на рисунку 3.4.

Окремо потрібно відмітити, що у програмі у кожен окремий момент часу може бути тільки один ініціалізований клас з перерахованих, або жодного. Це відбувається через те, що при зміні моделі, її клас зберігається на диск, та зникає з робочої пам’яті програми та заміщується класом нової моделі.

3.3 Розробка модуля потоків виконання та нормалізації вхідних даних

Головна задача модуля потоків виконання та нормалізації вхідних даних полягає в забезпеченні ефективного та оптимізованого виконання обробки великого обсягу даних. Цей модуль дозволяє розділити виконання довгих операцій на окремі потоки, щоб уникнути блокування головного потоку програми та забезпечити правильну роботу інтерфейсу користувача. Використання потоків дозволяє програмі продовжувати виконання інших завдань, навіть якщо одна або кілька довгих операцій ще не завершено.

Все починається з ініціалізації прогрес-бару для візуалізації процесу завантаження або обчислення. Після цього створюється новий потік, який запускає функцію create_model. Це дозволяє виконувати функцію create_model у фоновому режимі, не блокуючи основний потік виконання програми.

Для аналізу текстових даних спочатку необхідно створити модель яка буде навчатись на певних даних. Це представляє функція create_model, яка викликається для створення моделі класифікатора з використанням даних, які вводить користувач у графічному інтерфейсі програми. Вона перевіряє коректність введених параметрів, таких як розміри тестового і тренувального наборів даних, потім спробує створити модель на основі вибраного класифікатора та інших параметрів. Якщо створення моделі вдалося, вона зберігається у файлі, а інтерфейс оновлюється для відображення результатів. У

випадку помилки під час створення моделі або збереження необхідних файлів виводиться повідомлення про помилку для користувача.

Також існує клас Configurator задля того, щоб зробити можливим зберігання налаштувань користувача на диск. Цей клас реалізує усю потрібну логіку та керується двома простими для споживача його функціоналу методами: get_prefs() та update_pref(). update_pref() використовується при зміні налаштувань, а get_prefs() при потребі зчитати інформацію про налаштування. Клас Configurator завжди створюється під час запуску застосунку у єдиному екземплярі.

3.4 Висновки

У третьому розділі було обґрунтовано вибір технологій та інструментів, які були використані для розробки програмних модулів. Після проведення аналізу програмного продукту та його завдань було вирішено скористатися мовою програмування Python. Для зручності розробки графічного інтерфейсу користувача було використано бібліотеку Tkinter. У роботі з машинним навчанням була використана бібліотека Scikit-learn. Для аналізу тексту з цієї бібліотеки використовувався функціонал лінійних класифікаторів та наївного Байєрса. У розділі також було описано розробку основних програмних модулів для автоматичного виявлення фейкових новин в соціальних мережах

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Тестування – це процес перевірки програмного забезпечення з метою виявлення помилок та дефектів [20]. Його ціль – підтвердити, що програмний продукт працює правильно і відповідає очікуванням користувачів. Тестування включає в себе запуск програми з різними вхідними даними та умовами, а також порівняння отриманих результатів з очікуваними. Воно може виконуватися як вручну, так і за допомогою автоматизованих тестів. Тестування допомагає забезпечити якість програмного забезпечення і знижує ризик виявлення проблем під час його експлуатації.

Графічний інтерфейс застосунку є саме тією стороною програмного забезпечення, яку бачать та з якою взаємодіють користувачі, що робить її одним з головних елементів. Інтерфейс має бути зрозумілими, не перевантаженим зайвими функціями та, як найголовніше, – давати чіткий зворотній зв’язок про дії користувача та стан системи в теперішньому часі [21].

Тестування – це процес, який включає кілька важливих етапів. Спочатку проводиться підготовка тестових даних, яка полягає у зборі або створенні набору даних з реальними та фейковими новинами. Наступним кроком є розробка тестових випадків, які охоплюють різні сценарії та типи новин.

Після підготовки тестових даних і випадків починається етап виконання тестів, під час якого програма запускається з тестовими даними, і її реакція на кожен випадок перевіряється. Це може включати введення різних новин у програму і перевірку, як вона класифікує їх як фейкові чи справжні.

Після завершення тестування проводиться аналіз результатів, щоб визначити ефективність програми у виявленні фейкових новин. Якщо в процесі тестування виявлено помилки або недоліки, розробники вносять відповідні корективи в програмний код для покращення його роботи.

Останнім етапом є повторне тестування, яке проводиться після внесення змін, щоб переконатися, що виправлення помилок не призвело до появи нових проблем. Цей цикл тестування-виправлення може повторюватися декілька разів, доки не буде досягнуто відповідного рівня якості та надійності програми.

На рисунку 4.1 зображено інформаційне повідомлення де з’являється вікно з попередженням, що програма не може знайти жодної моделі та рекомендує створити модель.

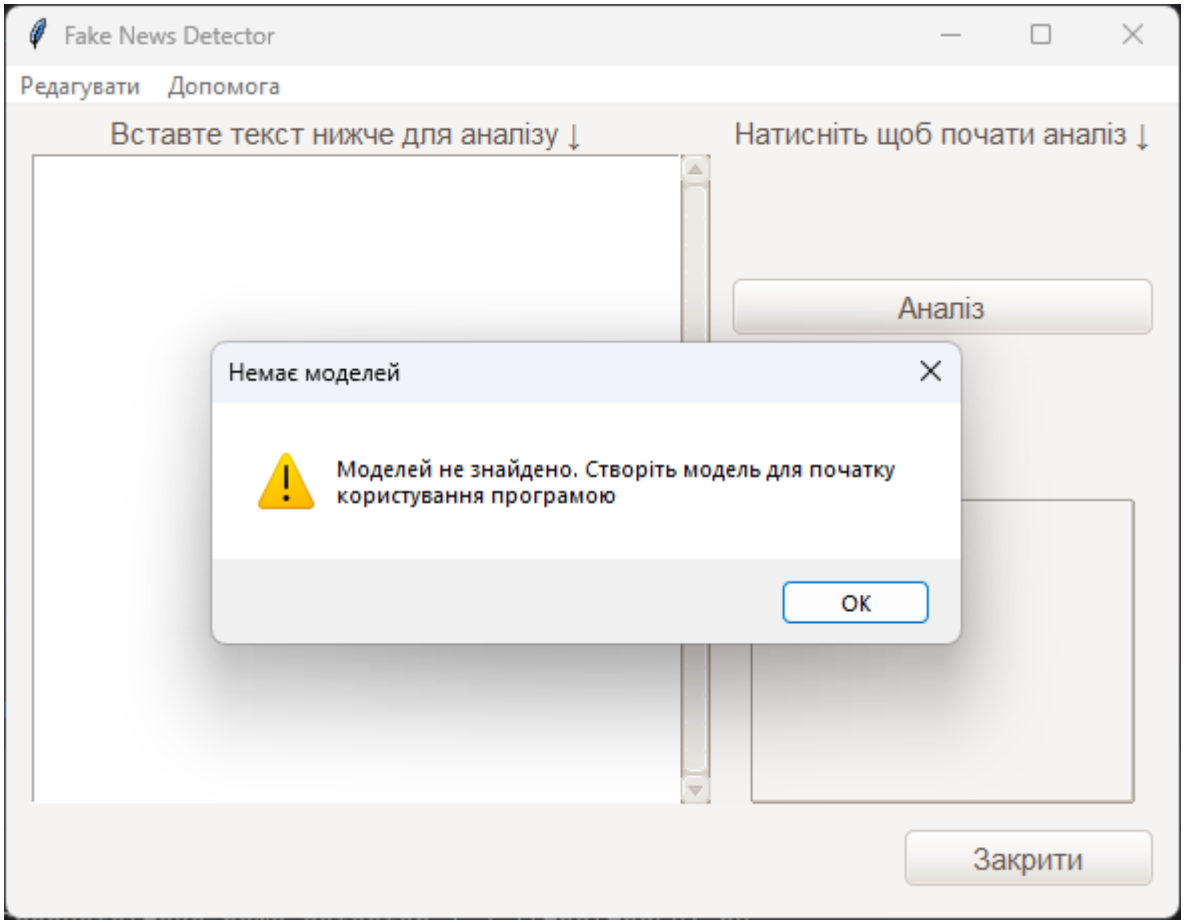

Рисунок 4.1 – Інформаційне повідомлення

При спробі зберегти модель з потенційно неправильними параметрами з’являється вікно, яке запитує користувача чи дійсно він впевнений у операції і пропонує йому змінити значення навчальної моделі (рис. 4.2).

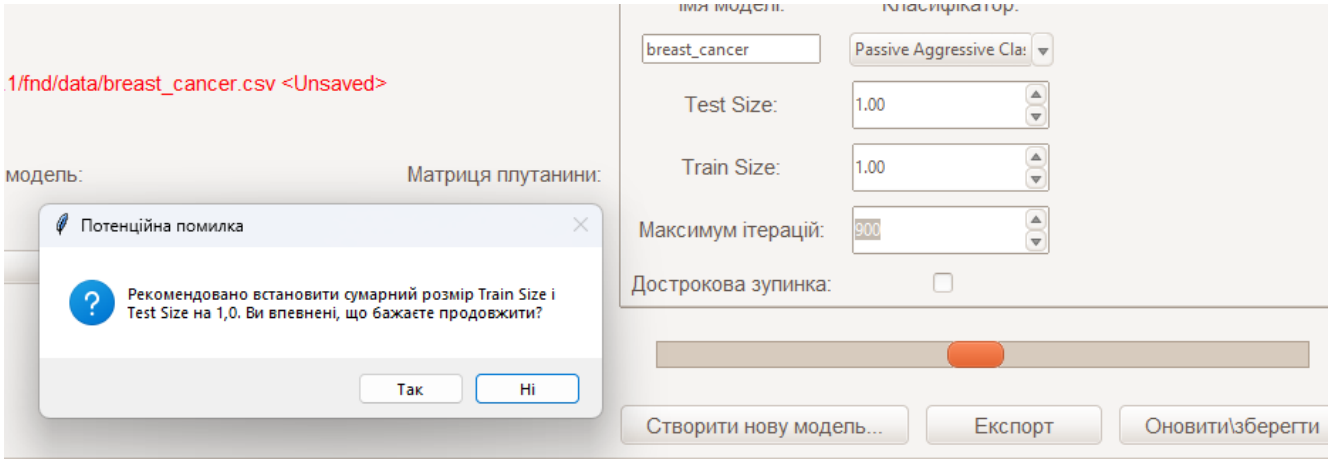

Рисунок 4.2 – Інформаційне повідомлення про потенційну помилку

Якщо користувач спробує завантажити файл з даними для навчання моделі формату відмінного від csv з'являється вікно з помилкою, що говорить користувачу про неможливість виконання операції та рекомендує, який тип файлу бажано використовувати.

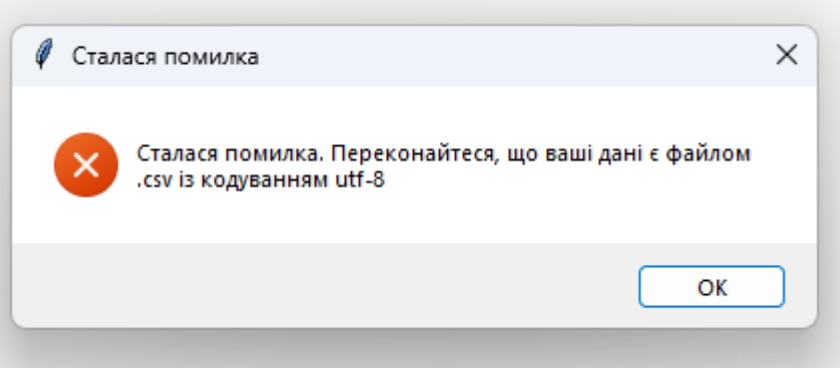

Рисунок 4.3 – Помилка завантаження файлу

Також було важливо переконатися, що програма правильно реагує на різні типи новин і може відрізняти реальні новини від фейкових у різних контекстах. Це дозволить користувачам впевнено використовувати програму для перевірки достовірності інформації, яку вони знаходять у соціальних мережах та інших джерелах. Таким чином, тестування на реальних та фейкових новинах підтвердило, що програмний застосунок може надійно визначати достовірність

новин у реальному часі і бути корисним інструментом для виявлення фейкових новин.

Під час тестування програми на фейковій новині, зображеній на рисунку 4.4, було виявлено, що програмний застосунок працює ефективно та вірно визначає новини як фейкові або реальні. Після аналізу вмісту новини та обробки її програмою, було отримано відповідь, яка відображає правильний результат. Таке тестування дозволило підтвердити, що програмний модуль для визначення фейкових новин виконує свою функцію з високою точністю та надійністю.

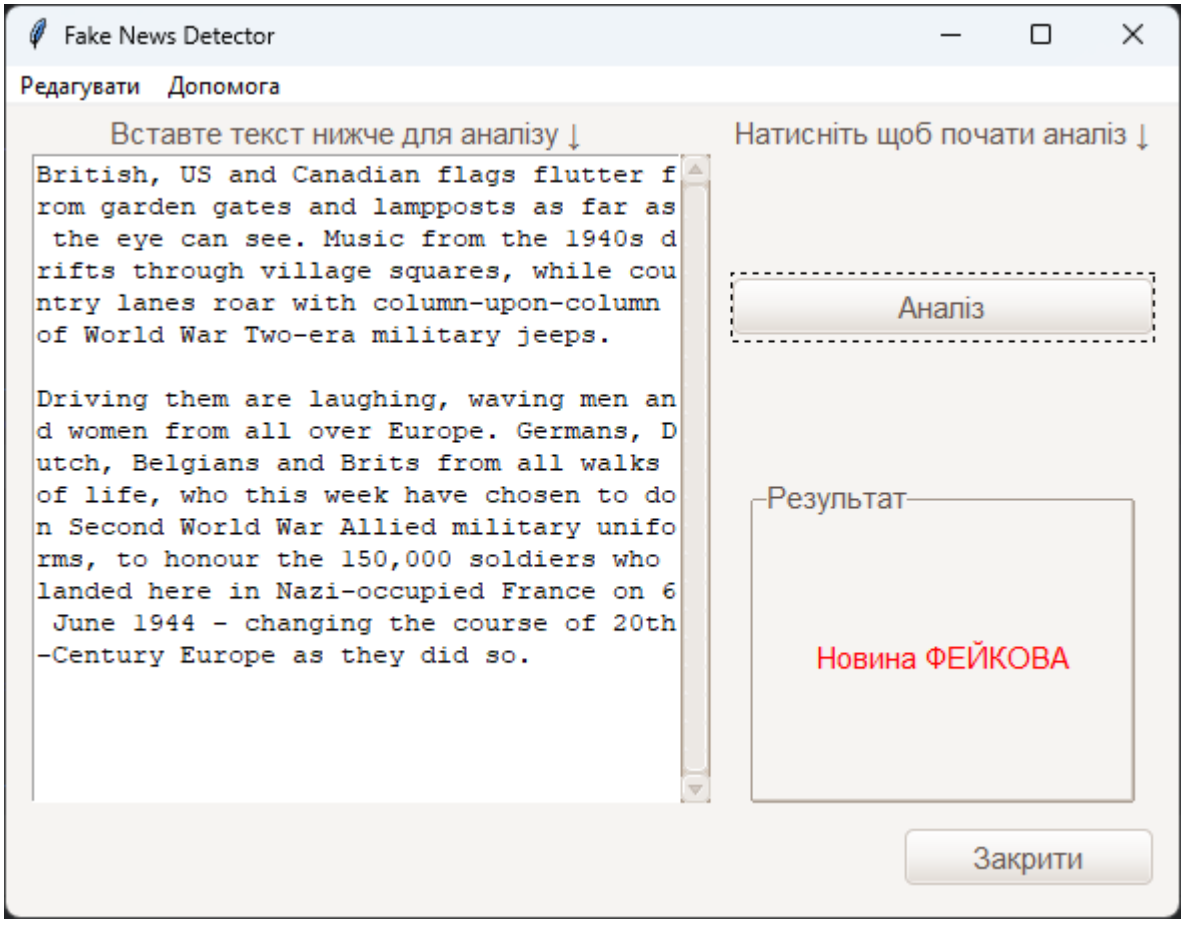

Рисунок 4.4 – Результат тестування на фейковій новині

Під час тестування на реальній новині було перевірено, наскільки ефективно програма визначає її достовірність. Результати показали (рис. 5.5), що програмний застосунок вірно класифікує реальні новини, що підтверджує його

здатність працювати з реальними даними і надавати коректні результати. Таке тестування дозволило переконатися, що програма може ефективно працювати з реальними новинами і визначати їх достовірність з високою точністю.

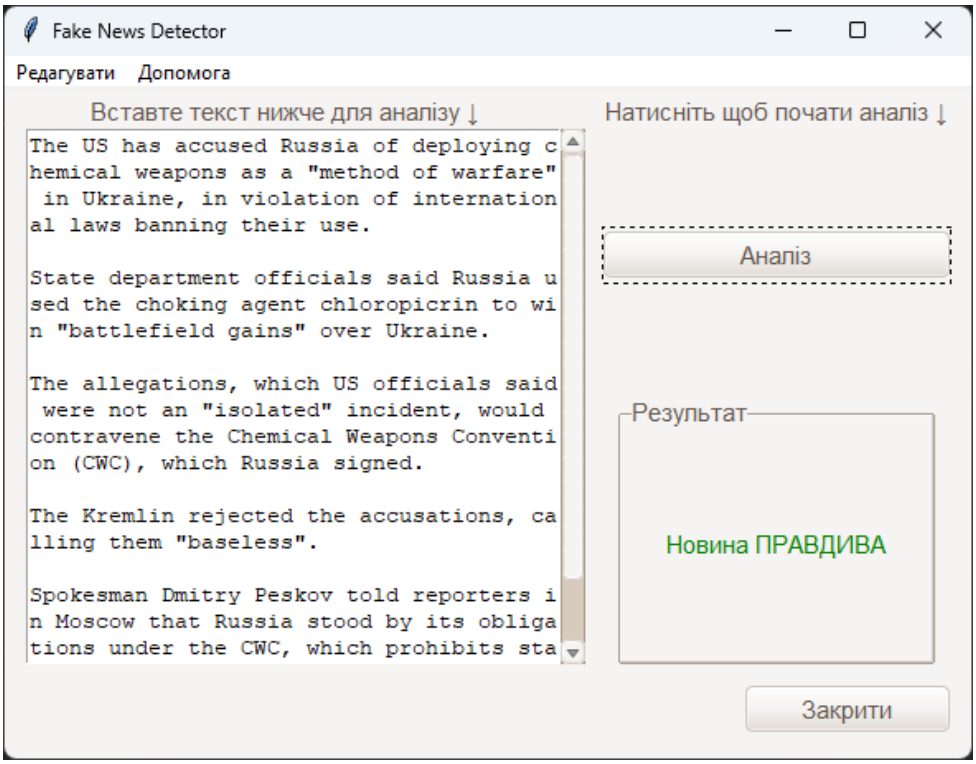

Рисунок 4.5 – Результат тестування на правдивій новині

Для перевірки точності розробленої базової моделі було вирішено протестувати її в реальних умовах, використовуючи вже промарковані новини з тієї частини даних, наданих Університетом Вікторії, які не були включені до навчального набору даних. Загалом було обрано 100 новин, з яких 41 виявилися фейковими, а 59 – реальними. Результати тестування можна побачити на рисунку 4.6.

```
Набір даних "test_news.csv" склав: 100 записів
З них промарковані: 59 -- REAL; 41 -- FAKE
Тестуєма модель: basic
* ------------------------------------------------------------------------------------------------------------------------ *
Результат: правильно вгаданих фейкових -- 36, правильно вгаданих реальних -- 40, неправильно вгаданих фейкових -- 5, неправильно вгаданих реальних -- 19
Загальна точність склала:  0.76
```

Рисунок 4.6 – Тестування точності

Таким чином, при загальній точності в 76% можна говорити про достатню ефективність моделі для того щоб користуватись програмою. Використання більшого обсягу навчальних даних та більшої кількості системних ресурсів може суттєво покращити результати.

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі щодо мінімальної та рекомендованої конфігурації серверного комп’ютера можна знайти в таблиці 4.1.

Також описано, як використовувати програмний продукт. Вона пояснює, як запустити програму, орієнтуватись по її інтерфейсу та виконувати основні операції. Інструкція також описує доступні функції та їх призначення, надаючи користувачеві повний огляд можливостей програми. Крім того, інструкція містить інформацію про те, як виконувати різноманітні завдання або операції за допомогою програми, що дозволяє користувачам ефективно використовувати програмний продукт для досягнення своїх цілей.

Таблиця 4.1 – Вимоги до апаратного забезпечення

Вимоги до конфігурування	Рекомендовано	Мінімально
Операційна система	Windows 10, Windows 11	Windows xp, windows 7
Процесор	800 mhz processor	500 mhz processor
Оперативна пам'ять	4 ГБ	2 ГБ
Відеокарта	Intel UHD 600 or higher	DirectX 8.1 level Graphics Card
Вільний простір на диску	2 ГБ	1 ГБ
Монітор	Роздільна здатність 1920x1080	Роздільна здатність 1366x768

Програмний продукт розроблено таким чином, щоб задовольнити потреби двох різних категорій користувачів. Перша категорія складається з тих, хто

шукає просте рішення, яке не вимагає спеціальних навичок або додаткового налаштування. Для цієї групи користувачів програма має бути легкою у використанні і не потребувати глибокого розуміння її функціоналу.

Друга категорія користувачів володіє спеціальними знаннями та може мати доступ до розширених можливостей програми. Вони можуть навчати власні моделі за допомогою власних наборів даних, налаштовувати параметри окремих моделей, спостерігати за їхньою ефективністю та маніпулювати ними різними способами. Для цієї категорії користувачів програма повинна надавати більше гнучкості та можливостей для роботи з моделями та їх налаштуванням.

На рисунку 4.7 можна побачити, як виглядає головне вікно застосунку. Програмний застосунок має простий та зрозумілий інтерфейс, саме те, що потрібно першому типу користувача, який бажає, щоб застосунок був легким у роботі, та не потребував додаткових умінь.

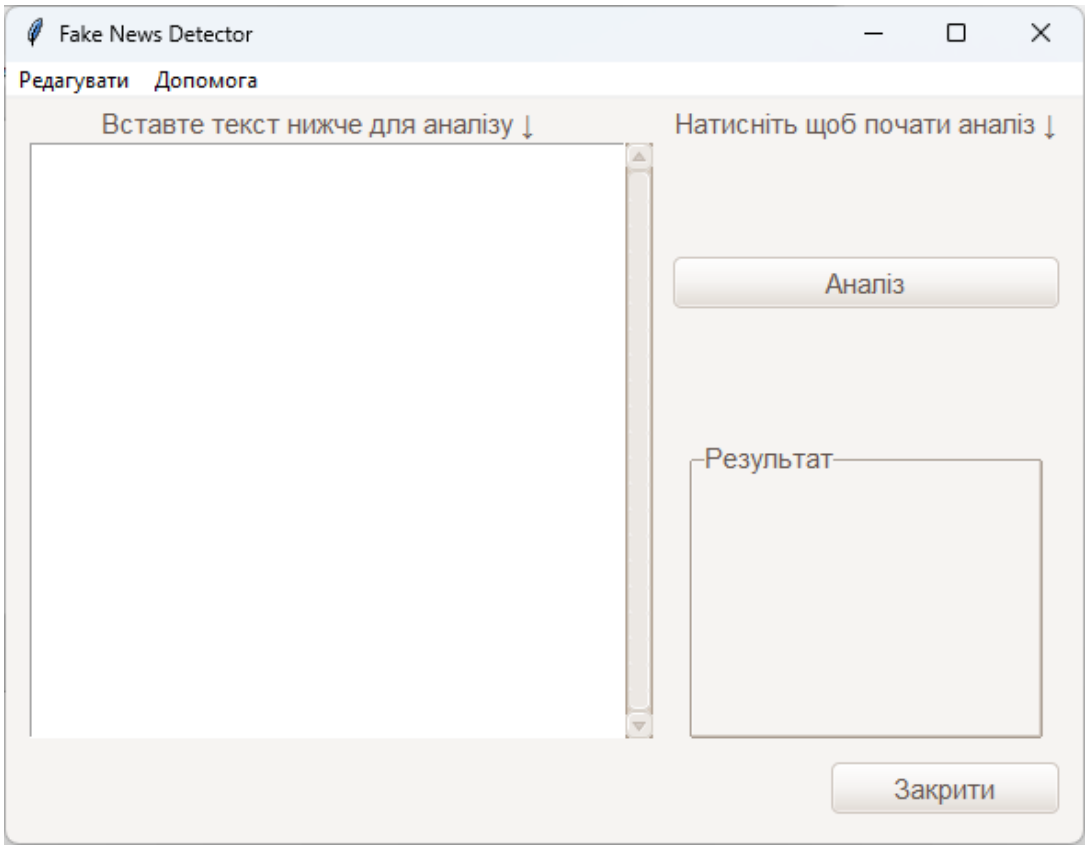

Рисунок 4.7 – Головне вікно програми

Використання застосунку є досить легким, адже кожне поле супроводжується підказкою.

Звідси, користувач може проаналізувати текст методом копіювання чи прямого вводу у текстове поле, натиснути кнопку «Аналіз» та отримати результат: «новина ФЕЙКОВА» або «новина ПРАВДИВА», що зображено на рисунку 4.8.

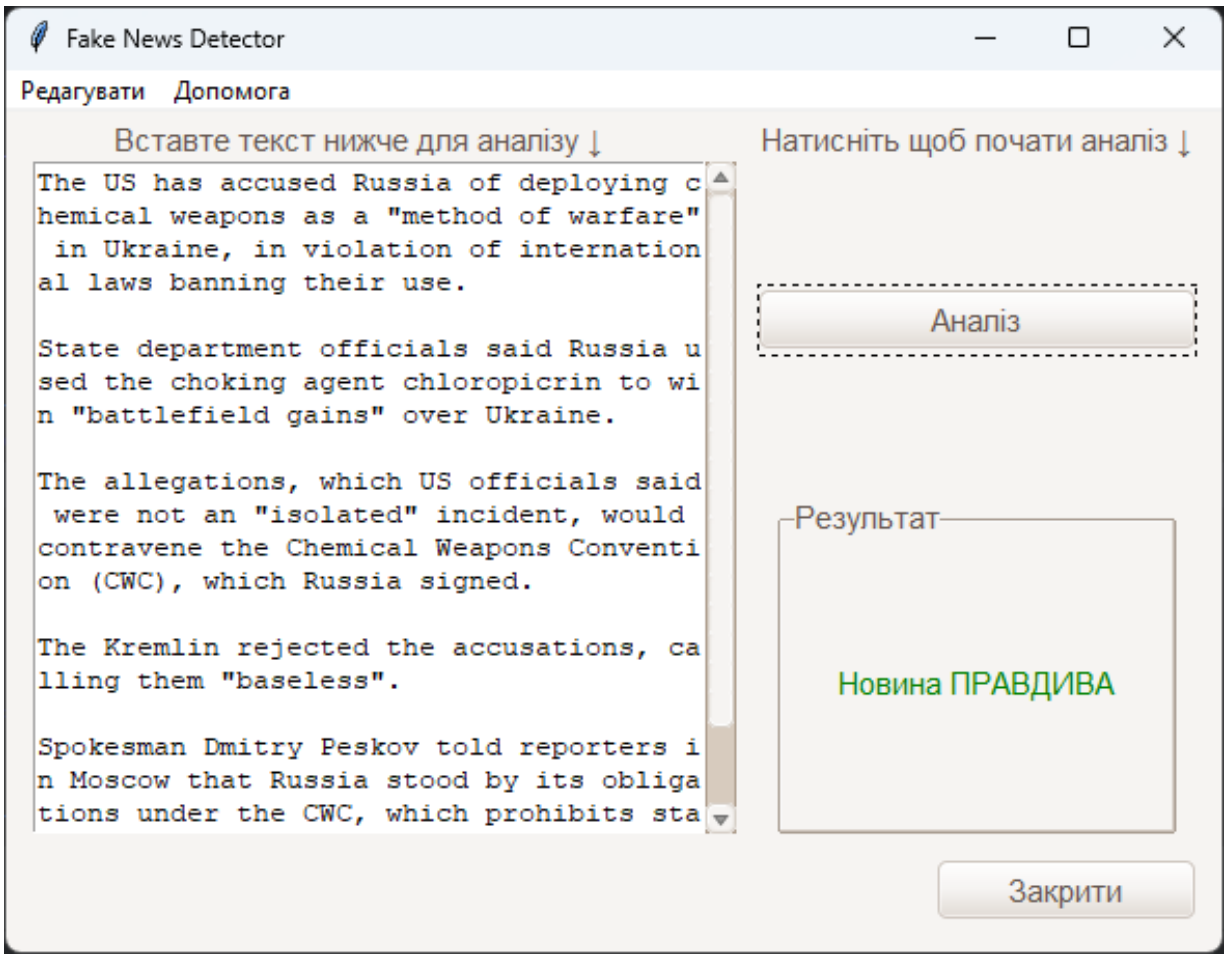

Рисунок 4.8 – Робота програмного застосунку

У верхній частині вікна розташовані дві кнопки: «Редагувати» та «Допомога». При натисканні на «Допомога» відкривається секція «Про програму», де можна знайти інформацію про розробника та основні використані технології. Вікно про програму зображено на рисунку 4.9.

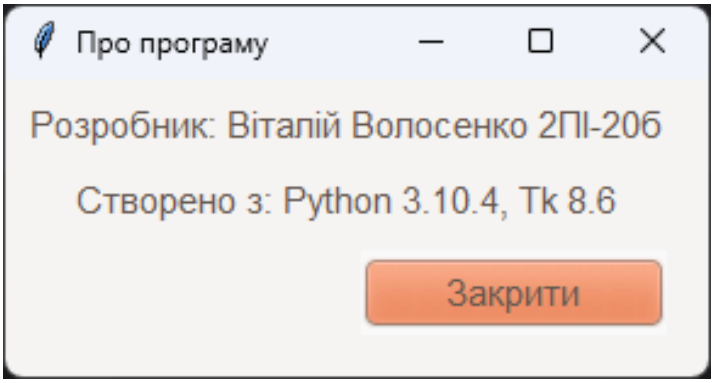

Рисунок 4.9 – Вікно «Про програму»

При натисканні на кнопку «Редагувати» користувач отримує доступ до двох секцій «Теми» та «Додаткові функції».

На рисунку 4.10 зображено що секція «Теми» дає можливість обрати одну з декількох графічних тем користувача.

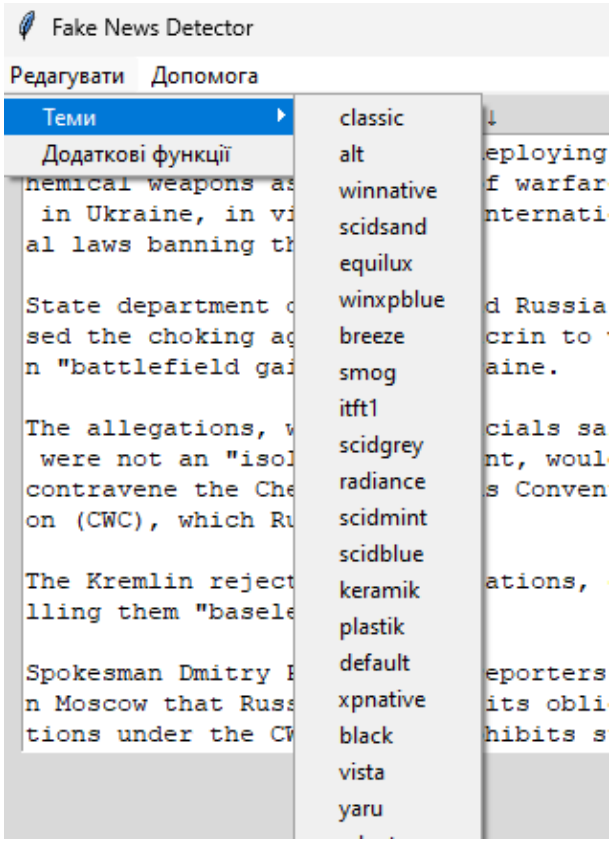

Рисунок 4.10 – Секція «Теми»

Інші користувачі, хто бажає мати більше контролю над їхніми моделями можуть активувати просунутий користувацький інтерфейс за допомогою натискання на секцію «Додаткові функції». На рисунку 4.11 зображено головне вікно програми з додатковим полем з параметрами та можливістю створення нових моделей, їх зміни та експортування.

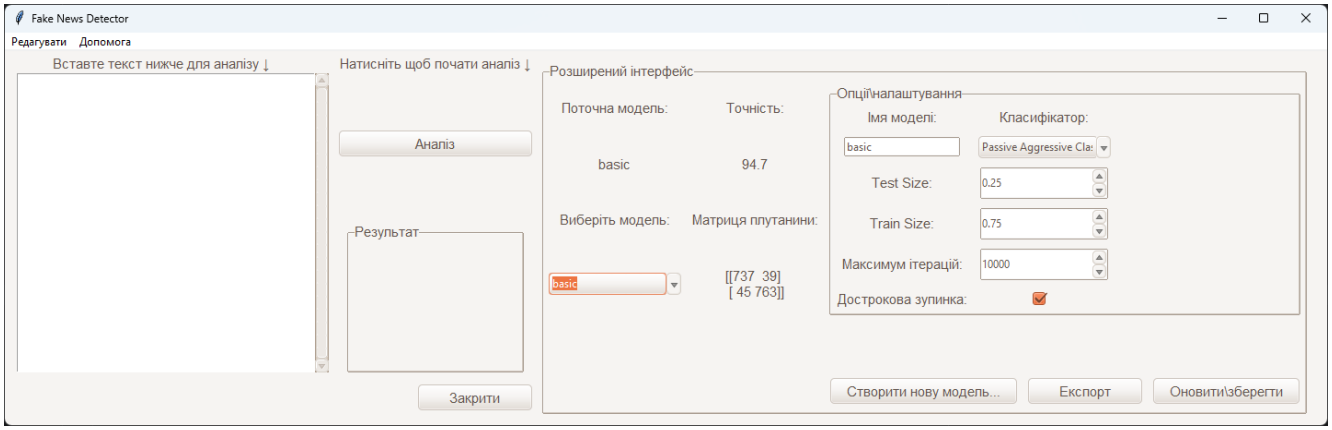

Рисунок 4.11 – Вікно програмного застосунку з розширеним інтерфейсом

У просунутому користувацькому інтерфейсі можна виділити три основні частини.

Перша частина містить інформацію про точність, назву та результати навчання обраної моделі. Тут можна змінити поточну модель, яка буде використовуватися для оцінки тексту.

У другій частині розташоване вікно "Опції/налаштування", яке дозволяє змінити налаштування існуючої моделі або вносити корективи під час створення нової. Можна змінити ім'я, обрати інший класифікатор (наприклад, пасивно-агресивний класифікатор, перцептрон, многочленний наївний Байєс або доповнюючий наївний Байєс). Також можна встановити пропорцію між даними, які будуть використовуватися для тренування та тестування, ввести кількість бажаних ітерацій для підвищення точності моделі та вказати, чи потрібно припинити навчання, коли точність перестає зростати.

Далі розташовано три кнопки: «Створити нову модель», «Експорт», «Оновити\Зберегти».

Кнопка «Створити нову модель» дозволяє створити модель для навчання за даними які користувач може обрати сам. На рисунку 4.12 зображено процес обирання даних користувачем.

Кнопка «Експорт» дозволяє користувачеві зберегти навчену модель на диску.

Кнопка «Оновити\Зберегти» дозволяє зберегти зміни до існуючої моделі, або зберегти нову модель, для якої обрано певні дані та налаштування її навчання. Процес навчання моделі зображено на рисунку 4.13.

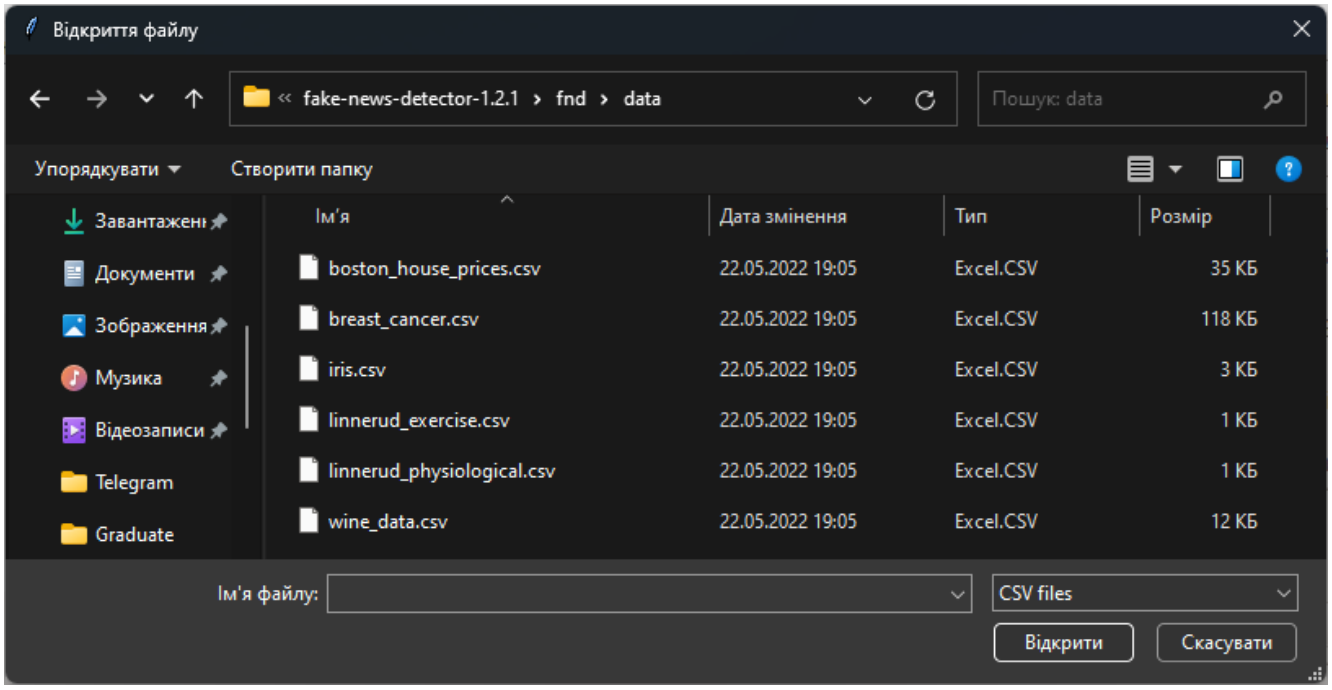

Рисунок 4.12 – Процес вибору даних для створення нової моделі

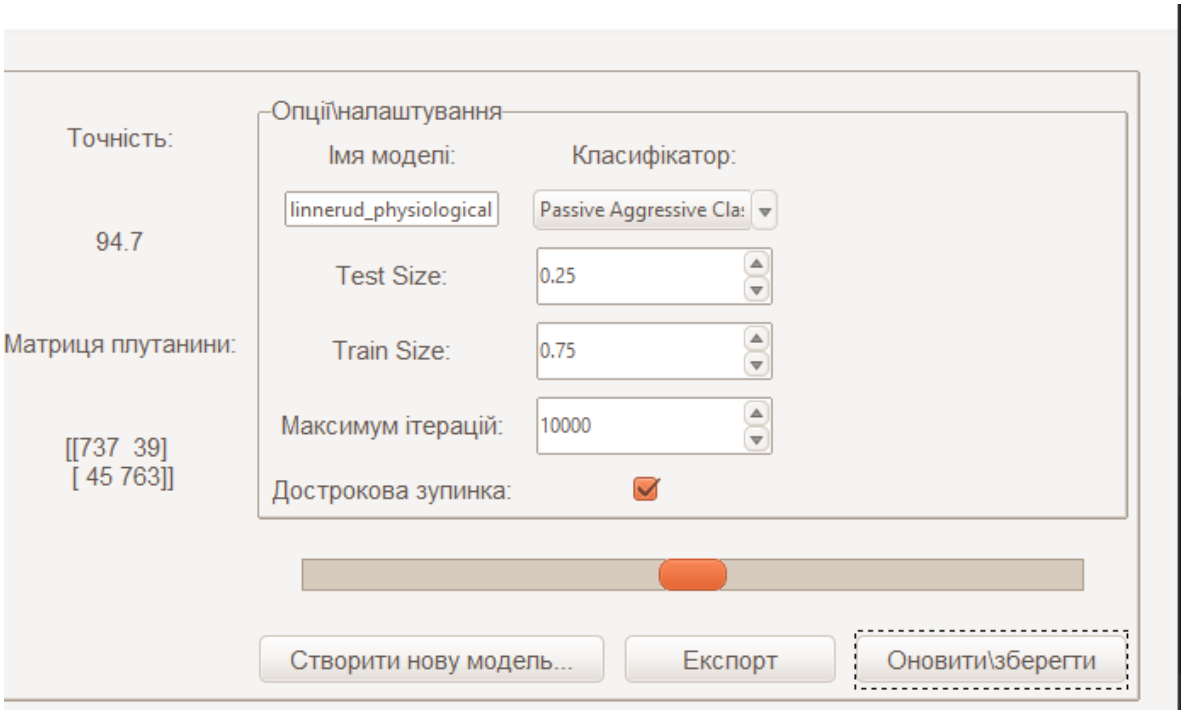

Рисунок 4.13 – Процес навчання моделі

Всі налаштування, обрані користувачем, зберігаються після закриття програми, забезпечуючи безперервний користувацький досвід. Наприклад, якщо користувач змінив тему інтерфейсу або встановив просунутий режим відображення, програма автоматично відкриватиметься в обраному вигляді при наступному запуску. Це дозволяє користувачу уникнути необхідності повторно налаштовувати програму при кожному її відкритті, що значно підвищує зручність та ефективність її використання.

4.3 Висновки

У четвертому розділі було проведено всебічне тестування програмних модулів програми для виявлення фейкових новин за допомогою машинного навчання. Було ретельно перевірено, як програма реагує на різні помилкові ситуації, що можуть виникнути під час її роботи, а також як вона обробляє вхідні дані різних форматів. Тестування включало аналіз стійкості програми до некоректних даних, перевірку її продуктивності та стабільності роботи. Для забезпечення максимальної точності та надійності результатів тестування було

застосовано різні підходи та методології. Зокрема, було використано тестові сценарії з різними типами помилкових даних, такими як неповні або некоректні текстові вхідні дані, а також дані, що не відповідають очікуваному формату. Проведене тестування та розробка інструкції користувача дозволили не лише виявити та усунути потенційні проблеми, але й значно покращити зручність використання програми, що сприяє її ефективній та надійній роботі в різних умовах експлуатації.

ВИСНОВКИ

Під час виконання бакалаврської дипломної роботи було розроблено програму автоматичного виявлення фейкових новин в соціальних мережах. Під час розробки використовувалось інтегроване середовище розробки PyCharm та мова програмування Python.

Було досліджено предметну область виявлення фейкових новин в соціальних мережах та розглянуто аналоги програмного застосунку. Кожен з цих засобів має свої переваги і недоліки, що були проаналізовані. На основі цього аналізу були сформульовані основні завдання дипломного проєкту.

Під час аналізу технологій та засобів розробки було обґрунтовано вибір в користь бібліотек Tkinter, Scikit-learn, Scipy, joblib та інших.

У бакалаврській дипломній роботі вирішено наступні задачі:

- проведено аналіз стану технологій аналізу інформації;
- розроблено інтерфейс програмного застосунку;
- розроблено алгоритми для ефективної обробки великого обсягу даних з соціальних мереж;
- розроблено модуль класифікації тексту;
- створено модель машинного навчання для автоматичної класифікації новин на реальні та фейкові заснованої на їх характеристиках;
- проведено тестування розробленого програмного забезпечення.

Було проведено аналіз вхідних та вихідних даних програмних засобів, розглянуто процес створення графічного інтерфейсу програмного застосунку. Також розроблено основний алгоритм роботи програми та алгоритми визначення фейкових новин. Крім того, була розроблена модель роботи програмного застосунку за допомогою UML діаграми.

Після проведення тестування виявлено, що програмний застосунок працює добре і відповідає вимогам. Він надійно розпізнає фейкові новини у соціальних мережах та надає користувачам зручний та інтуїтивно зрозумілий інтерфейс для

використання. Тестування також показало, що програма ефективно використовує обрані технології та інструменти для досягнення своїх цілей.

Загалом, розроблений програмний застосунок є потужним інструментом для боротьби з фейковими новинами, який допомагає користувачам бути краще обізнаними та захищеними в інформаційному просторі. Програма пройшла тестування, що підтверджує її актуальність та високу ефективність.

ЛІТЕРАТУРА

1. Як розпізнати фейк? - правила The Huffington Post [Електронний ресурс] – Режим доступу до ресурсу: https://cert.gov.ua/recommendation/30

2. Narwal, B. Fake News and Digital Media. – 2021. 234 с.

3. Волосенок В. Ф., Чехместрук Р. Ю. Розробка програм для автоматичного виявлення фейкових новин в соціальних мережах // Стан, досягнення та перспективи інформаційних систем і технологій: матеріали XXIV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів, Одеса, 18-19 квітня 2024 р. Одеса: Видавництво ОНТУ, 2024. 498 с.

4. About checking facts on Facebook, Instagram and Threads [Електронний ресурс] – Режим доступу до ресурсу: https://www.facebook.com/business/help/2593586717571940

5. InVID WeVerify extension [Електронний ресурс] – Режим доступу до ресурсу: https://chromewebstore.google.com/detail/fake-news-debunker-by-inv/mhccpoafgdgbhnjfhkcmgknndkeenfhe

6. В Україні запустили фактчек-бот для відстеження фейків [Електронний ресурс] – Режим доступу до ресурсу: https://www.ukrinform.ua/rubric-technology/3440117-v-ukraini-zapustili-faktcekbot-dla-vidstezenna-fejkiv.html

7. Машинне навчання, метод опорних векторів [Електронний ресурс] – Режим доступу до ресурсу: https://stud.com.ua/25073/menedzhment/mashinne_navchannya

8. David, R. Bayesian Statistics: The Theory and Its Applications. – 2022. 94 с.

9. Python is a programming language that lets you work quickly and integrate systems more effectively [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/

10. Kravchenko, S., Gryshkun, Y., & Vlasenko, O. Machine Learning Classification Methods Using scikit-learn. – 2023. 42 с.

11. Pandas – Python Data Analysis Library [Електронний ресурс] – Режим доступу до ресурсу: https://pandas.pydata.org/

12. Pickle – Python object serialization [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/pickle.html

13. An Intro to Threading in Python [Електронний ресурс] – Режим доступу до ресурсу: https://realpython.com/intro-to-python-threading/

14. Керівництво з використання Tkinter у Python: [Електронний ресурс] – Режим доступу до ресурсу: https://dukarnia.com.ua/articles/kerivnictvo-z-vikoristannya-tkinter-u-python-pochatok-roboti-ta-stvorennya-novogo-vikna-T-A9-

15. Smola, A., & Vishwanathan, S. Introduction to Machine Learning. – 2022. 86 с.

16. Fundamental algorithms for scientific computing in Python [Електронний ресурс] – Режим доступу до ресурсу: https://scipy.org/

17. Joblib: running Python functions as pipeline jobs [Електронний ресурс] – Режим доступу до ресурсу: https://joblib.readthedocs.io/en/stable/#joblib-running-python-functions-as-pipeline-jobs

18. Khanna R. Machine Learning / R.Khanna, M.Awad [Електронний ресурс] – Режим доступу до ресурсу: https://link.springer.com/chapter/10.1007/978-1-4302-5990-9_1

19. Croce D. GAN-BERT: Generative Adversarial Learning for Robust Text Classification with a Bunch of Labeled Examples / D.Croce, G.Castellucci, R.Basili [Електронний ресурс] – Режим доступу до ресурсу: https://aclanthology.org/2020.acl-main.191/

20. Що таке тестування програмного забезпечення? [Електронний ресурс] – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/shho-take-testuvannya-programnogo-zabezpechennya/

21. Що таке дизайн графічного інтерфейсу користувача GUI? [Електронний ресурс] – Режим доступу до ресурсу: https://training.qatestlab.com/blog/technical-articles/what-is-graphical-user-interface-design/

ДОДАТКИ

54

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
_____ О. Н. Романюк
«12» березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка програм для автоматичного виявлення фейкових новин у соціальних мережах» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ к.т.н., доц. каф. ПЗ Чехместрук Р.Ю.
«12» березня 2024 р.
Виконав:
_____ студент гр. 2ПІ-20б Волосенко В.Ф.
«12» березня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка програм для автоматичного виявлення фейкових новин у соціальних мережах». Галузь застосування – соціальні мережі.

2. Підстава для розробки

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р. ректора ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки

Метою та призначенням розробки є створення програмного забезпечення, яке здатне виявляти фейкові новини на основі алгоритмів машинного навчання. Користувач зможе фільтрувати недостовірну інформацію, забезпечуючи себе достовірними джерелами і зміцненню довіри до інформаційного простору.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

1. Волосенко В. Ф., Чехместрук Р. Ю. Розробка програм для автоматичного виявлення фейкових новин у соціальних мережах // Стан, досягнення та перспективи інформаційних систем і технологій: матеріали XXIV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів, Одеса, 18-19 квітня 2024 р. Одеса: Видавництво ОНТУ, 2024. 498 с.

5. Технічні вимоги

Вихідні дані до роботи: середовище розробки PyCharm, мова розробки Python, бібліотека scikit-learn, бібліотека tkinter, алгоритм наївного Байєса, лінійний класифікатор.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БДР.
2. Технічне завдання.
3. Лістинги програми

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ

9. Стадії та етапи розробки:

№ п\п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану технологій аналізу інформації	13.03.24 – 30.03.24
2	Вибір середовища та мови розробки	02.04.24 – 15.04.24
3	Розробка інтерфейсу, моделі та алгоритмів програмного застосунку	20.04.24 – 30.04.24
4	Розробка модуля класифікації тексту	01.05.24 – 09.05.24
5	Розробка модуля потоків виконання та нормалізації вхідних даних	10.05.24 – 19.05.24
6	Тестування	20.05.24 – 24.05.24
7	Оформлення матеріалів до захисту БДР	30.05.24 – 03.06.24

10. Порядок контролю та прийняття

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програм для автоматичного виявлення фейкових новин у соціальних мережах

Тип роботи: БДР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Чехместрук Р.Ю.

Оригінальність	90,2
Схожість	9,8

Аналіз звіту подібності

▪ **Запозичення, виявлені у роботі, оформленні коректно і не містять ознак плагіату.**

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цілісності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховання недобросовісних запозичень.

Особа, відповідальна за перевірку Черноволик Г.О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи Волосенко В.Ф.

Керівник роботи Чехместрук Р.Ю.

Додаток В – Лістинг програми

configurator.py

```
import configparser
import os

class Configurator:
    def __init__(self, folder='./settings', file_name='preferences.ini'):
        self.config = configparser.ConfigParser()

        # subject of rewrite to accept one path whole
        self._file_path = folder + '/' + file_name

        if not os.path.exists(folder):
            os.makedirs(folder)
            self.config.add_section('Preferences')
            self.config['Preferences']['theme'] = 'arc'
            self.config['Preferences']['mode'] = 'simple'
            self.update_pref()

        self.config.read(self._file_path)

    def update_pref(self, **kwargs):
        for key, value in kwargs.items():
            self.config['Preferences'][key] = value
        with open(self._file_path, 'w') as configfile:
            self.config.write(configfile)

    def get_prefs(self):
        return dict(self.config.items('Preferences'))


if __name__ == '__main__':
    test = Configurator()
    test.get_prefs()

#theme = config['Preferences']['theme']
#mode = config['Preferences']['mode']
```

fnd.py

```
"""Text classification

This module is intended to be used for classifying text as either
fake news or not and defines the following classes:

- `Detector`, a classifier superclass
- `LinearDetector`, a linear classifer superclass
- `BayesDetector`, a bayes classifer superclass
- `PAClassifier`, a linear passive aggressive classifier
- `Percept`, a linear perceptron classifier
```

```
- `MultiNB`, a multinomial naive bayes classifier
- `ComplNB`, a complement naive bayes classifier

__docformat__ = 'restructuredtext'

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import PassiveAggressiveClassifier, Perceptron
from sklearn.naive_bayes import MultinomialNB, ComplementNB
from sklearn.metrics import accuracy_score, confusion_matrix

class Detector:
    def __init__(
            self, classifier,
            data='',
            test_size=0.25,
            train_size=0.75,
            column_title='text',
            data_frame=None) -> None:

        self.test_size = test_size
        self.train_size = train_size

        self.classifier = classifier

        if data == '':
            self.data_frame = data_frame
        else:
            self.data_frame = pd.read_csv(data, encoding='utf-8')

        self.labels = self.data_frame.label
        x_train, x_test, self.y_train, self.y_test = train_test_split(
            self.data_frame[column_title], self.labels, test_size=self.test_size, train_size=self.train_size)
        self.tfidf_vectorizer = TfidfVectorizer(stop_words='english')
        self.tfidf_train = self.tfidf_vectorizer.fit_transform(x_train)
        self.tfidf_test = self.tfidf_vectorizer.transform(x_test)

        self.classifier.fit(self.tfidf_train, self.y_train)
        self.y_pred = self.classifier.predict(self.tfidf_test)
        self.score = accuracy_score(self.y_test, self.y_pred)
        self.matrix = confusion_matrix(
            self.y_test, self.y_pred, labels=[
                'FAKE', 'REAL'])

    def predict(self, text):
        # using the same vectoriser?
        # list() divides sentences into letters, interesting
        vec_newtest = self.tfidf_vectorizer.transform([text])
        t_pred = self.classifier.predict(vec_newtest)
        return t_pred
```

```
    def __getattr__(self, attr):  # fix for Pickle
        """Return None for all unknown attributes and raise an exception for all unknown dunder-
attributes"""
        if attr.startswith('__') and attr.endswith('__'):
            raise AttributeError
        return None

class LinearDetector(Detector):
    def __init__(
            self, classifier,
            data='',
            test_size=0.25,
            train_size=0.75,
            max_iter=1000,
            early_stopping=False,
            column_title='text',
            data_frame=None) -> None:

        self.max_iter = max_iter
        self.early_stopping = early_stopping

        Detector.__init__(
            self, classifier(
                max_iter=self.max_iter,
                early_stopping=self.early_stopping),
            data,
            test_size,
            train_size,
            column_title,
            data_frame)

class PAClassifier(LinearDetector):
    def __init__(
            self,
            data='',
            test_size=0.25,
            train_size=0.75,
            max_iter=1000,
            early_stopping=False,
            column_title='text',
            data_frame=None) -> None:

        LinearDetector.__init__(
            self, PassiveAggressiveClassifier,
            data,
            test_size,
            train_size,
            max_iter,
            early_stopping,
            column_title,
```

```
            data_frame)

    def __str__(self) -> str:
        return 'Passive Aggressive Classifier'

class Percept(LinearDetector):
    def __init__(
            self,
            data='',
            test_size=0.25,
            train_size=0.75,
            max_iter=1000,
            early_stopping=False,
            column_title='text',
            data_frame=None) -> None:

        LinearDetector.__init__(
            self, Perceptron,
            data,
            test_size,
            train_size,
            max_iter,
            early_stopping,
            column_title,
            data_frame)

    def __str__(self) -> str:
        return 'Perceptron'

class BayesDetector(Detector):
    def __init__(
            self, classifier,
            data='',
            test_size=0.25,
            train_size=0.75,
            column_title='text',
            data_frame=None) -> None:

        Detector.__init__(
            self, classifier(),
            data,
            test_size,
            train_size,
            column_title,
            data_frame)

class MultiNB(BayesDetector):
    def __init__(
            self,
            data='',
            test_size=0.25,
```

```
            train_size=0.75,
            column_title='text', data_frame=None, **kwargs) -> None:

        BayesDetector.__init__(
            self, MultinomialNB,
            data,
            test_size,
            train_size,
            column_title,
            data_frame)

    def __str__(self) -> str:
        return 'Multinomial Naive Bayes'

class ComplNB(BayesDetector):
    def __init__(
            self,
            data='',
            test_size=0.25,
            train_size=0.75,
            column_title='text', data_frame=None, **kwargs) -> None:

        BayesDetector.__init__(
            self, ComplementNB,
            data,
            test_size,
            train_size,
            column_title,
            data_frame)

    def __str__(self) -> str:
        return 'Complement Naive Bayes'
```

fndGUI.py

```
import fnd
import pickle
import os
import threading
import configurator

from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from tkinter import filedialog
from pathlib import Path
from ttkthemes import themed_style

class FakeDetector:
```

```
def __init__(self, root) -> None:

    root.title('Fake News Detector')

    # Menu
    root.option_add('*tearOff', FALSE)
    menubar = Menu(root)
    root['menu'] = menubar
    menu_pref = Menu(menubar)
    menu_help = Menu(menubar)
    menu_theme = Menu(menu_pref)
    menubar.add_cascade(menu=menu_pref, label='Редагувати', underline=0)
    menubar.add_cascade(menu=menu_help, label='Допомога', underline=0)
    menu_pref.add_cascade(menu=menu_theme, label='Теми', underline=0)

    self.s = themed_style.ThemedStyle(root)
    for theme_name in self.s.theme_names():
        # Using the i=i trick causes your function to store the current
        # value of i at the time your lambda is defined, instead of waiting
        # to look up the value of i later.
        menu_theme.add_command(
            label=theme_name,
            command=lambda th=theme_name: self.update_theme(th))

    menu_help.add_command(
        label='Про програму',
        command=lambda: self.show_about(root), underline=0)

    self.configuration = configurator.Configurator()
    self.prefs = self.configuration.get_prefs()
    self.s.set_theme(self.prefs['theme'])

    self.interface_mode = StringVar()
    self.interface_mode.set(self.prefs['mode'])

    menu_pref.add_checkbutton(
        label='Додаткові функції',
        command=lambda: self.set_interface(
            self.interface_mode.get(),
            self.mainframe),
        variable=self.interface_mode,
        onvalue='advanced',
        offvalue='simple',
        underline=0)

    # Main frame
    self.mainframe = ttk.Frame(root, padding='3 3 12 12')
    self.mainframe.grid(column=0, row=0, sticky=(N, W, E, S))
    root.columnconfigure(0, weight=1)
    root.rowconfigure(0, weight=1)
```

```
# temp
self.set_interface(
    self.interface_mode.get(),
    self.mainframe)  # might want to rework
# temp
self.input_text = Text(self.mainframe, width=40, height=20)
self.input_text.grid(
    column=0, row=1, rowspan=2, sticky=(
        N, W, E, S), padx=(
        10, 0))
self.scroll_text = ttk.Scrollbar(
    self.mainframe, orient=VERTICAL, command=self.input_text.yview)
self.scroll_text.grid(
    column=1, row=1, rowspan=2, sticky=(
        N, S, W), padx=(0, 10))  # sticky=(N,S)
self.input_text.configure(yscrollcommand=self.scroll_text.set)

labelfr_result = ttk.Labelframe(self.mainframe, text='Результат')
labelfr_result.grid(
    column=2, row=2, sticky=(
        N, W, S, E), pady=(
        10, 0), padx=10)
self.analysis_result = StringVar()
self.result_lb = ttk.Label(
    labelfr_result,
    textvariable=self.analysis_result)
self.result_lb.grid(column=0, row=0)

labelfr_result.columnconfigure(0, weight=1)
labelfr_result.rowconfigure(0, weight=1)

ttk.Button(
    self.mainframe,
    text='Аналіз',
    state='активний',
    command=self.analyse).grid(
    column=2,
    row=1, sticky=(E, W))  # sticky=W
ttk.Button(self.mainframe, text='Закрити', command=root.destroy).grid(
    column=2, row=3, pady=(10, 0), sticky=(E, S))

ttk.Label(
    self.mainframe,
    text='Вставте текст нижче для аналізу \u2193').grid(
    column=0,
    row=0,
    sticky=(S, N))
ttk.Label(
    self.mainframe,
    text='Натисніть щоб почати аналіз \u2193').grid(
```

```
            column=2,
            row=0,
            sticky=S)

        if len(dir_list := os.listdir('./models')) != 0:
            if 'basic.pickle' in dir_list:
                self.unpickle_model('./models/basic.pickle')  # meh
            else:
                # have to check for exceptions down below
                self.unpickle_model('./models/' + dir_list[0])
        else:
            messagebox.showwarning(
                title='Немає моделей',
                message='Моделей не знайдено. Створіть модель для початку користування
програмою')

        # self.update_padding(self.mainframe)

        # Temporarily
        self.mainframe.columnconfigure((0, 2), weight=1)
        self.mainframe.rowconfigure((1, 2), weight=1)

        # for child in self.mainframe.winfo_children():
        #     name = str(child)
        #     if name == '.!frame.!scrollbar':
        #         child.grid_configure(padx=(0, 5), pady=5)
        #         continue
        #     elif name == '.!frame.!text':
        #         child.grid_configure(padx=(5, 0), pady=5)
        #         continue
        #     elif name == '.!frame.!button2':
        #         continue
        #     child.grid_configure(padx=5, pady=5)

        self.input_text.focus_set()
        root.bind('<Return>', self.analyse)

    # def update_padding(self, frame):
    #     columns, rows = frame.grid_size()
    #     frame.rowconfigure(rows, pad=5)
    #     frame.columnconfigure(columns, pad=5)

    def analyse(self, *args):
        # end-1c trim newline # there still is a newline \n
        try:
            result = self.model.predict(self.input_text.get('1.0', 'end-1c'))
        except AttributeError:
            messagebox.showerror(
                title='Немає моделей',
                message='Неможливо аналізувати. Створіть модель для початку користування
програмою')
```

```
            return
        if result == ['FAKE']:
            self.analysis_result.set('Новина ФЕЙКОВА')
            self.result_lb['foreground'] = 'red'
        elif result == ['REAL']:
            self.analysis_result.set('Новина ПРАВДИВА')
            self.result_lb['foreground'] = 'green'

    def set_interface(self, mode, mainframe):
        # Not very OOP; maybe rework
        if mode == 'advanced':
            self.configuration.update_pref(mode='advanced')
            self.labelfr_advanced = ttk.Labelframe(
                mainframe, text='Розширений інтерфейс')
            self.labelfr_advanced.grid(
                column=3, row=0, rowspan=4, sticky=(
                    N, W, S, E), pady=(10, 0), padx=10)
            ttk.Label(
                self.labelfr_advanced,
                text='Поточна модель:').grid(
                column=0,
                row=0)
            self.current_model = StringVar()
            self.current_model_lb = ttk.Label(
                self.labelfr_advanced,
                textvariable=self.current_model)
            self.current_model_lb.grid(
                column=0,
                row=1)
            ttk.Label(
                self.labelfr_advanced,
                text='Виберіть модель:').grid(
                column=0,
                row=2)
            self.selected_model = StringVar()
            self.cb_models = ttk.Combobox(
                self.labelfr_advanced,
                # maybe rework in case i need to check the directory in several
                # places? # should i also use lambda? # should update after
                # each click on update button # for now have to restart the
                # program
                values=self.get_models(),
                state='readonly', textvariable=self.selected_model)
            self.cb_models.grid(
                column=0,
                row=3)
            self.cb_models.bind(
                '<<ComboboxSelected>>',
                lambda e: self.unpickle_model(
                    './models/' +
                    self.selected_model.get() +
```

```
        '.pickle'))  # test without lambda for interest # no need to put self in?
ttk.Label(
    self.labelfr_advanced,
    text='Точність:').grid(
    column=1,
    row=0)
self.accuracy = StringVar()
ttk.Label(
    self.labelfr_advanced,
    textvariable=self.accuracy).grid(
    column=1,
    row=1)
ttk.Label(
    self.labelfr_advanced,
    text='Матриця плутанини:').grid(
    column=1,
    row=2)
self.stats = StringVar()
ttk.Label(
    self.labelfr_advanced,
    textvariable=self.stats).grid(
    column=1,
    row=3)

self.labelfr_options = ttk.Labelframe(
    self.labelfr_advanced, text=r'Опції\налаштування')
self.labelfr_options.grid(
    column=2, row=0, columnspan=3, rowspan=4, sticky=(
        N, W, S, E), pady=(10, 0), padx=10)
ttk.Label(
    self.labelfr_options,
    text='Імя моделі:').grid(
    column=0,
    row=0)
self.model_name = StringVar()
ttk.Entry(
    self.labelfr_options,
    textvariable=self.model_name).grid(
    column=0,
    row=1)
ttk.Label(
    self.labelfr_options,
    text='Класифікатор:').grid(
    column=1,
    row=0)
self.CLASSIFIER_MAPPING = {
    'Passive Aggressive Classifier': fnd.PAClassifier,
    'Perceptron': fnd.Percept,
    'Multinomial Naive Bayes': fnd.MultiNB,
    'Complement Naive Bayes': fnd.ComplNB}
self.CLASSIFIER_ITEMS = sorted(self.CLASSIFIER_MAPPING.keys())
```

```
self.selected_classifier = StringVar()
self.selected_classifier.set(
    'Passive Aggressive Classifier')  # self.CLASSIFIER_ITEMS[0] # tmp
self.cb_classifier = ttk.Combobox(
    self.labelfr_options,
    textvariable=self.selected_classifier,
    values=self.CLASSIFIER_ITEMS,
    state='readonly')
self.cb_classifier.grid(
    column=1,
    row=1)
self.cb_classifier.bind(
    '<<ComboboxSelected>>',
    lambda e: self.disable_widgets(
        self.CLASSIFIER_MAPPING[self.selected_classifier.get()]))
ttk.Label(
    self.labelfr_options,
    text='Test Size:').grid(
    column=0,
    row=2)
ttk.Label(
    self.labelfr_options,
    text='Train Size:').grid(
    column=0,
    row=3)
self.spin_test = DoubleVar()
self.spin_test.set(0.25)
# change readonly to validation
ttk.Spinbox(
    self.labelfr_options,
    from_=0.0,
    to=1.0,
    textvariable=self.spin_test,
    increment=0.05,
    format='%.2f',
    state='readonly').grid(
    column=1,
    row=2)
self.spin_train = DoubleVar()
self.spin_train.set(0.75)
ttk.Spinbox(
    self.labelfr_options,
    from_=0.0,
    to=1.0,
    textvariable=self.spin_train,
    increment=0.05,
    format='%.2f',
    state='readonly').grid(
    column=1,
    row=3)
ttk.Label(
```

```
    self.labelfr_options,
    text='Максимум ітерацій:').grid(
    column=0,
    row=4)
ttk.Label(
    self.labelfr_options,
    text='Дострокова зупинка:').grid(
    column=0,
    row=5)
self.spin_iter = IntVar()
self.spin_iter.set(1000)
self.sb_iter = ttk.Spinbox(
    self.labelfr_options,
    from_=0,
    to=10000,
    textvariable=self.spin_iter,
    increment=100)
self.sb_iter.grid(
    column=1,
    row=4)  # add progress bar for big numbers
self.spin_stopping = BooleanVar()  # not a spinbox: rename
self.spin_stopping.set(False)
self.ckbtn_stopping = ttk.Checkbutton(
    self.labelfr_options,
    variable=self.spin_stopping,
    onvalue=True,
    offvalue=False)
self.ckbtn_stopping.grid(
    column=1,
    row=5)
ttk.Button(
    self.labelfr_advanced,
    text='Створити нову модель...',  # maybe just Create new model?
    command=self.get_filename).grid(
    column=2,
    row=5,
    pady=(
        10,
        0))
ttk.Button(
    self.labelfr_advanced,
    text='Експорт',
    command=self.export_model).grid(
    column=3,
    row=5,
    pady=(
        10,
        0))
ttk.Button(
    self.labelfr_advanced,
    text='Оновити\\зберегти',  # rename to just save?
```

```
            command=self.thread_helper).grid(
            column=4,
            row=5,
            pady=(
                10,
                0))

        # tmp
        self.labelfr_advanced.columnconfigure((2, 3, 4), weight=1)
        self.labelfr_advanced.rowconfigure((4), weight=1)
        # tmp

        # testing
        for child in self.labelfr_advanced.winfo_children(
        ):  # create a function(frame, padding, except for)
            child.grid_configure(padx=5, pady=5)

        for child in self.labelfr_options.winfo_children():
            child.grid_configure(padx=5, pady=5)

    elif mode == 'simple':
        self.configuration.update_pref(mode='simple')
        try:
            self.labelfr_advanced
        except AttributeError:
            pass  # is it even legal?
        else:
            self.labelfr_advanced.destroy()

def get_models(self):
    return [os.path.splitext(file_name)[0]
            for file_name in os.listdir('./models')]

def disable_widgets(self, classifier_class):
    # isinstance(classifier,fnd.LinearDetector) or
    if issubclass(classifier_class, fnd.LinearDetector):
        self.ckbtn_stopping['state'] = 'enabled'
        self.sb_iter['state'] = 'enabled'
    else:
        self.ckbtn_stopping['state'] = 'disabled'
        self.sb_iter['state'] = 'disabled'

def export_model(self):
    path = filedialog.asksaveasfilename(
        initialfile='myModel.pickle',
        initialdir='./models',
        filetypes=[
            ("pickle files",
             ".pickle"),
            ("all files",
             "*.*")])  # file type; more flair
```

```
    # make sure that path is set
    if path == ():
        return
    self.pickle_model(
        self.model,
        path)  # file type; more flair

def update_theme(self, theme_name):
    self.configuration.update_pref(theme=theme_name)
    self.s.set_theme(theme_name)

def get_filename(self):
    self.file = filedialog.askopenfile(
        filetypes=[("CSV files", ".csv"), ("all files", "*.*")])  # check if cancelled
    # use regex to propose a default file name
    try:
        self.current_model.set(self.file.name + ' <Unsaved>')
    except AttributeError:
        return
    self.current_model_lb['foreground'] = 'red'  # make red
    self.cb_models.set('')
    # self.model_name.set(os.path.basename(os.path.normcase(self.file.name)))
    self.model_name.set(Path(self.file.name).stem)

def thread_helper(self):
    self.prog_bar = ttk.Progressbar(
        self.labelfr_advanced,
        orient=HORIZONTAL,
        length=460,
        mode='indeterminate')
    self.prog_bar.grid(row=4, column=2, columnspan=3)
    self.prog_bar.start()
    t = threading.Thread(target=self.create_model)
    t.start()
    # t.join()

# def multiprocess_model(self):
#     # use queue? # add a progress bar
#     p = multiprocessing.Process(target=self.create_model)
#     p.start()
#     p.join()
#     #p.close()

def update_after_create(self):
    self.set_labels(self.model)
    self.cb_models['values'] = self.get_models()
    self.prog_bar.stop()
    self.prog_bar.destroy()

def create_model(self):
    if self.spin_test.get() + self.spin_train.get() != 1.0:
```

```
            if messagebox.askyesno(
                    title='Потенційна помилка',
                    message='Рекомендовано встановити сумарний розмір Train Size і Test Size на 1,0. Ви впевнені, що бажаєте продовжити?',
                    default='no') is False:
                return
        try:
            self.model = self.CLASSIFIER_MAPPING[self.selected_classifier.get()](data=self.file, test_size=self.spin_test.get(
            ), train_size=self.spin_train.get(), max_iter=self.spin_iter.get(), early_stopping=self.spin_stopping.get())
            self.file.close()
        except (AttributeError, ValueError):
            try:
                self.model = self.CLASSIFIER_MAPPING[self.selected_classifier.get()](data_frame=self.model.data_frame, test_size=self.spin_test.get(
                ), train_size=self.spin_train.get(), max_iter=self.spin_iter.get(), early_stopping=self.spin_stopping.get())
            except AttributeError:
                messagebox.showerror(
                    title='Сталася помилка',
                    message='Сталася помилка. Переконайтеся, що ваші дані є файлом .csv із кодуванням utf-8')
                return

        self.pickle_model(
            self.model,
            './models/' +
            self.model_name.get() +
            '.pickle')

        self.update_after_create()

    def pickle_model(self, model, file_path):
        with open(file_path, 'wb') as f:
            pickle.dump(model, f, pickle.HIGHEST_PROTOCOL)

    def unpickle_model(self, file_path):
        with open(file_path, 'rb') as f:
            self.model = pickle.load(f)
        self.set_labels(self.model)

    def set_labels(self, model):
        if self.interface_mode.get() == 'advanced':
            self.accuracy.set(round(model.score * 100, 2))
            self.stats.set(model.matrix)
            if self.model_name.get() == '':
                self.model_name.set(self.selected_model.get())
            self.current_model.set(self.model_name.get())
            self.current_model_lb['foreground'] = ''
```

```
            self.spin_test.set(model.test_size)
            self.spin_train.set(model.train_size)
            self.selected_classifier.set(str(model))

            if isinstance(model, fnd.LinearDetector):
                # enable max iter and early stopping
                # have to send classes in; else cant check for both class and
                # isntance
                self.disable_widgets(model.__class__)
                self.spin_iter.set(model.max_iter)
                self.spin_stopping.set(model.early_stopping)
            else:
                # disable max iter and early stop
                self.disable_widgets(model.__class__)

    def show_about(self, root):
        win_about = Toplevel(root)
        win_about.title('Про програму')
        win_about.columnconfigure(0, weight=1)
        win_about.rowconfigure(0, weight=1)
        frame_about = ttk.Frame(win_about, padding='3 3 12 12')
        frame_about.grid(column=0, row=0, sticky=(N, W, E, S))
        ttk.Label(
            frame_about,
            text='Розробник: Віталій Волосенко 2ПІ-20б').grid(
            column=0,
            row=0, columnspan=2)
        ttk.Label(
            frame_about,
            text='Створено з: Python 3.10.4, Tk 8.6').grid(
            column=0,
            row=1, columnspan=2)

        ttk.Button(
            frame_about,
            text='Закрити',
            command=win_about.destroy,
            state='active').grid(
            column=1,
            row=4,
            sticky=(E, S))
        frame_about.columnconfigure((0, 1), weight=1)
        frame_about.rowconfigure((0, 1, 2, 3), weight=1)
        win_about.bind('<Return>', lambda e: win_about.destroy())

        for child in frame_about.winfo_children():
            child.grid_configure(padx=5, pady=5)

if __name__ == '__main__':
    if not os.path.exists(
            path_to_check := './models'):
```

```
    os.makedirs(path_to_check)
root = Tk()
FakeDetector(root)
root.mainloop()
```

Додаток Г – Ілюстративна частина

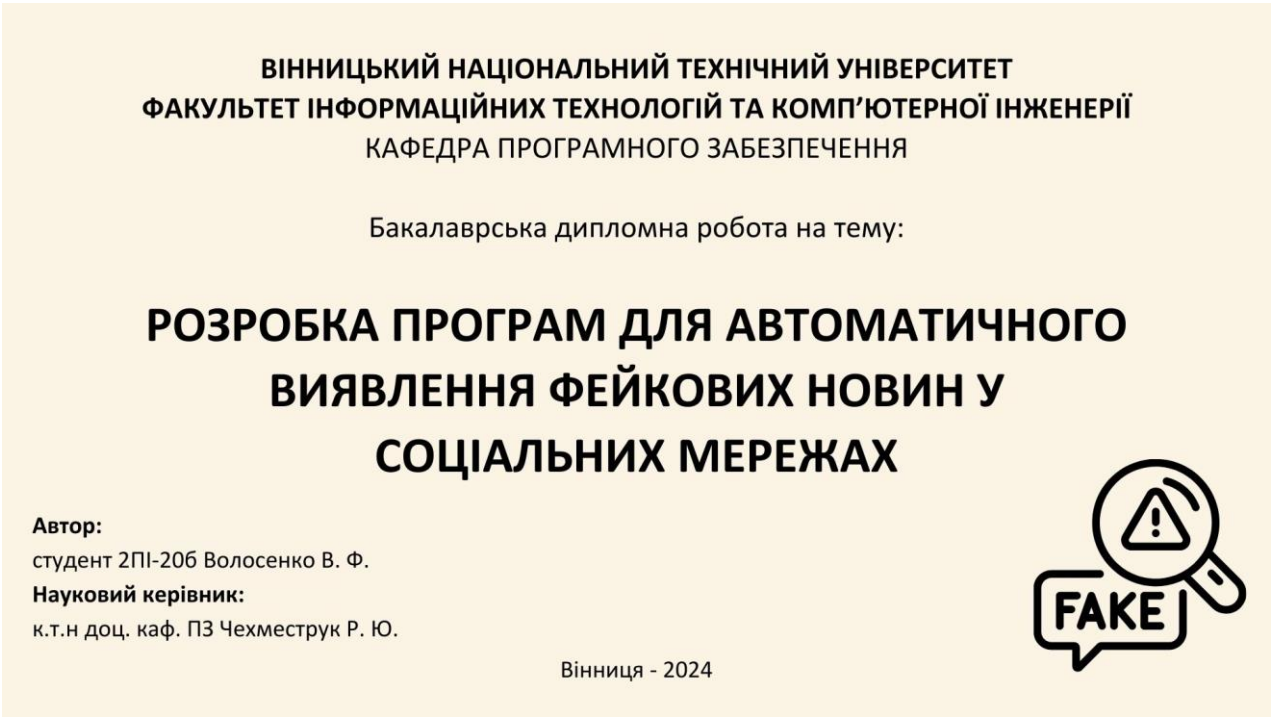

Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Актуальність роботи

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – створення програмного забезпечення, яке здатне автоматично виявляти фейкові новини на основі різноманітних аналітичних методів та алгоритмів машинного навчання.

Об'єкт дослідження – процес виявлення та фільтрації фейкових новин в інтернеті та соціальних мережах.

Предмет дослідження – розробка засобів для виявлення фейкових новин, які базуються на аналізі тексту та застосуванні методів машинного навчання.

Рисунок Г.3 – Мета, об'єкт та предмет дослідження

ЗАГАЛЬНИЙ ОГЛЯД ПРОБЛЕМИ

Зі зростанням впливу соціальних мереж як основного джерела інформації для багатьох людей, поширення фейкових новин стає серйозною загрозою для суспільства та демократичних процесів. **Потенційно шкідлива інформація може впливати на громадську думку, політичні вибори, економічні ринки та інші аспекти суспільного життя.** Існує необхідність в розробці та впровадженні ефективних інструментів для автоматичного виявлення фейкових новин у соціальних мережах. Це важлива задача, **оскільки потрібно забезпечити достовірність інформації**, яка доходить до користувачів, та **підтримати надійність соціальних мереж як засобу комунікації.**

Рисунок Г.4 – Загальний огляд проблеми

АНАЛІЗ СУЧАСНИХ ПІДХОДІВ

Критерії	Fake news debunker	Facebook Fact-Checking	Бот "Перевірка" в Телеграмі	Розроблювана програма
Машинне навчання	-	-	-	+
Аналіз тексту	-	+	+	+
Доступність для користувачів	-	-	+	+
Швидкість аналізу	+	-	+	+
Розширені можливості	+	-	-	+
Загальна оцінка	40%	20%	60%	100%

Рисунок Г.5 – Аналіз сучасних підходів

ПОСТАНОВКА ЗАДАЧІ

Після визначення проблематики та аналізу існуючих рішень було виділено наступні задачі:

- провести аналіз стану технологій аналізу інформації;
- розробити інтерфейс програмного застосунку;
- розробити алгоритми для ефективної обробки великого обсягу даних з соціальних мереж;
- розробити модуль класифікації тексту;
- створити модель машинного навчання для класифікації новин на реальні та фейкові заснованої на їх характеристиках;
- провести тестування розробленого програмного забезпечення.

Рисунок Г.6 – Постановка задачі

ЗАСОБИ І ТЕХНОЛОГІЇ

Під час розробки програмного застосунку були використані наступні технології: мова програмування **Python**, бібліотека для машинного навчання **scikit-learn**, а також бібліотека для створення графічного інтерфейсу **tkinter**.

Рисунок Г.7 – Засоби і технології

ПРАКТИЧНА ЦІННІСТЬ

Практична цінність отриманих результатів полягає у створенні ефективного програмного забезпечення для автоматичного виявлення фейкових новин у соціальних мережах. **Розроблені алгоритми та модулі можуть бути інтегровані в існуючі платформи для моніторингу та фільтрації контенту, що дозволить значно знизити поширення дезінформації.** Це, у свою чергу, сприятиме підвищенню якості інформації, доступної користувачам, і зменшенню негативного впливу фейкових новин на суспільство.

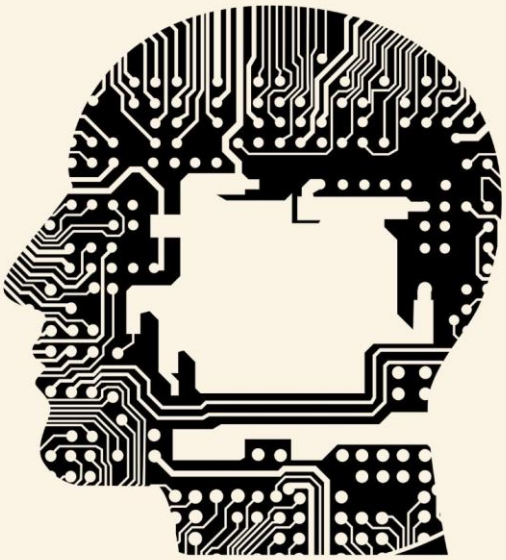

Рисунок Г.8 – Практична цінність

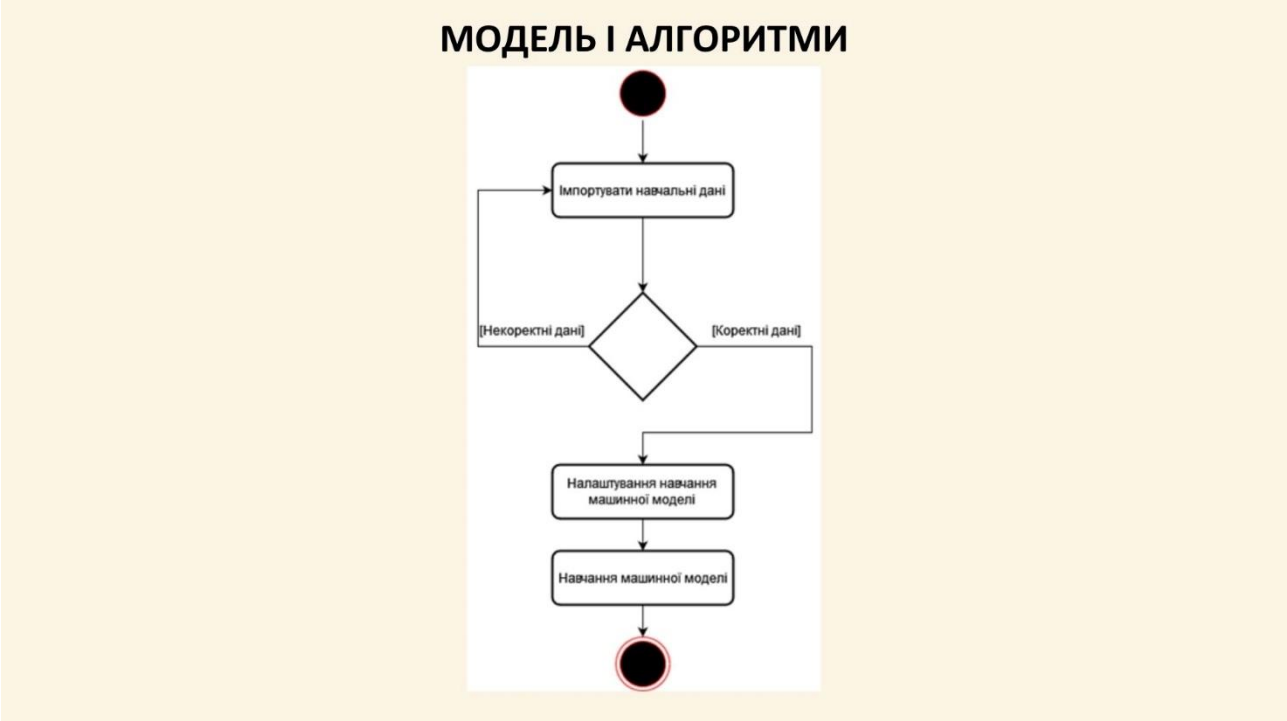

Рисунок Г.9 – Модель і алгоритми

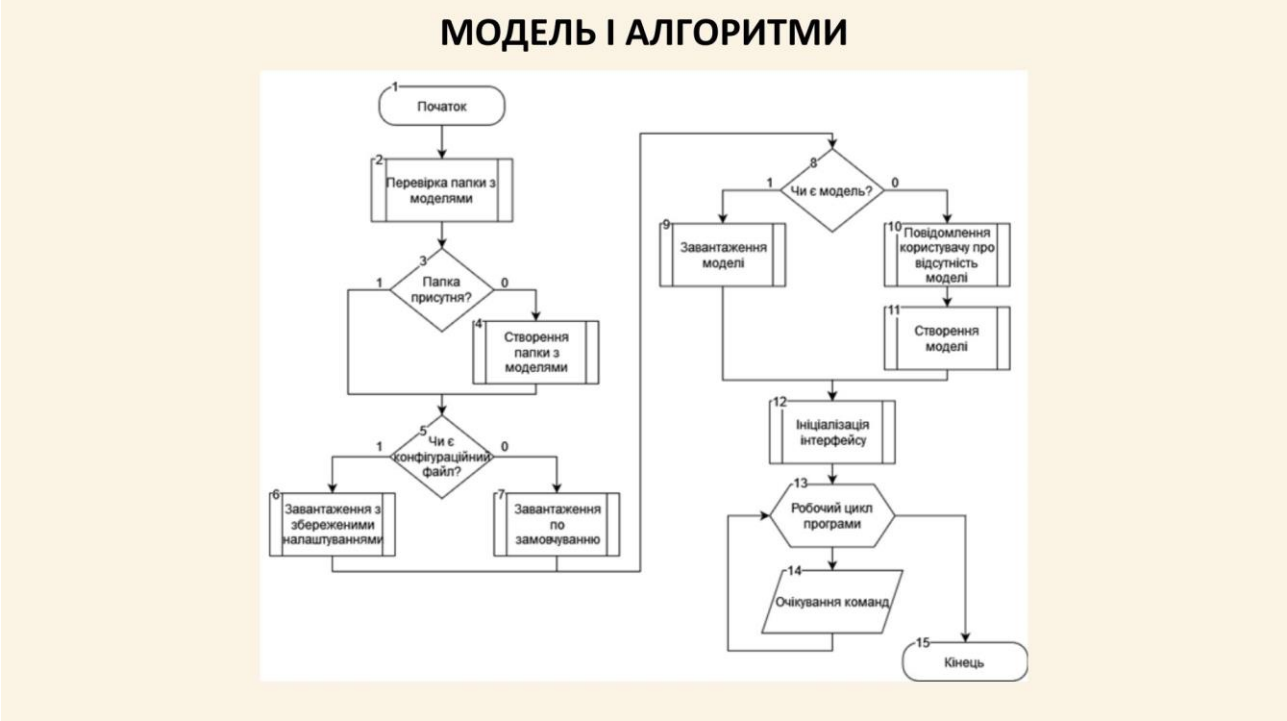

Рисунок Г.10 – Загальний алгоритм

ПРИНЦИП РОБОТИ ПРОГРАМНОГО МОДУЛЯ

ПРОЦЕС ЗАПУСКУ ПРОГРАМИ:

- Перевірка наявності папки з моделями;
- Ініціалізація інтерфейсу;
- Робочий цикл програми;
- Програма оновлює графічний інтерфейс і перевіряє наявність команд користувача;
- Якщо команд немає, цикл повторюється.

Рисунок Г.11 – Принцип роботи програмного модуля

ІНТЕРФЕЙС КОРИСТУВАЧА

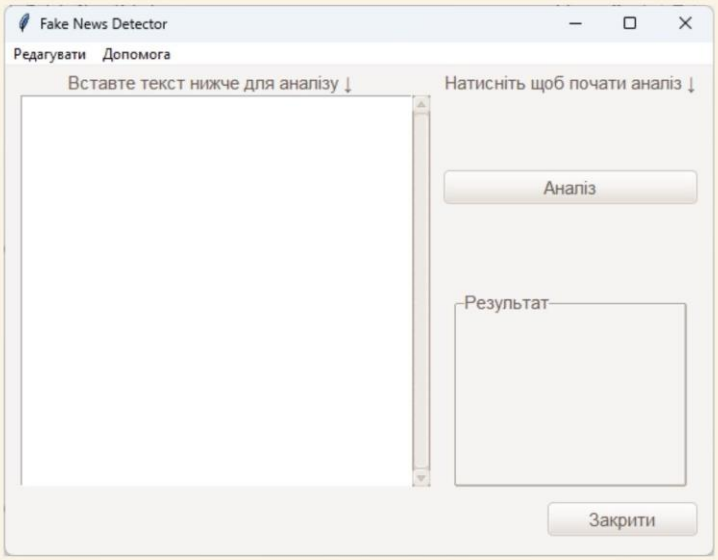

Рисунок Г.12 – Інтерфейс користувача

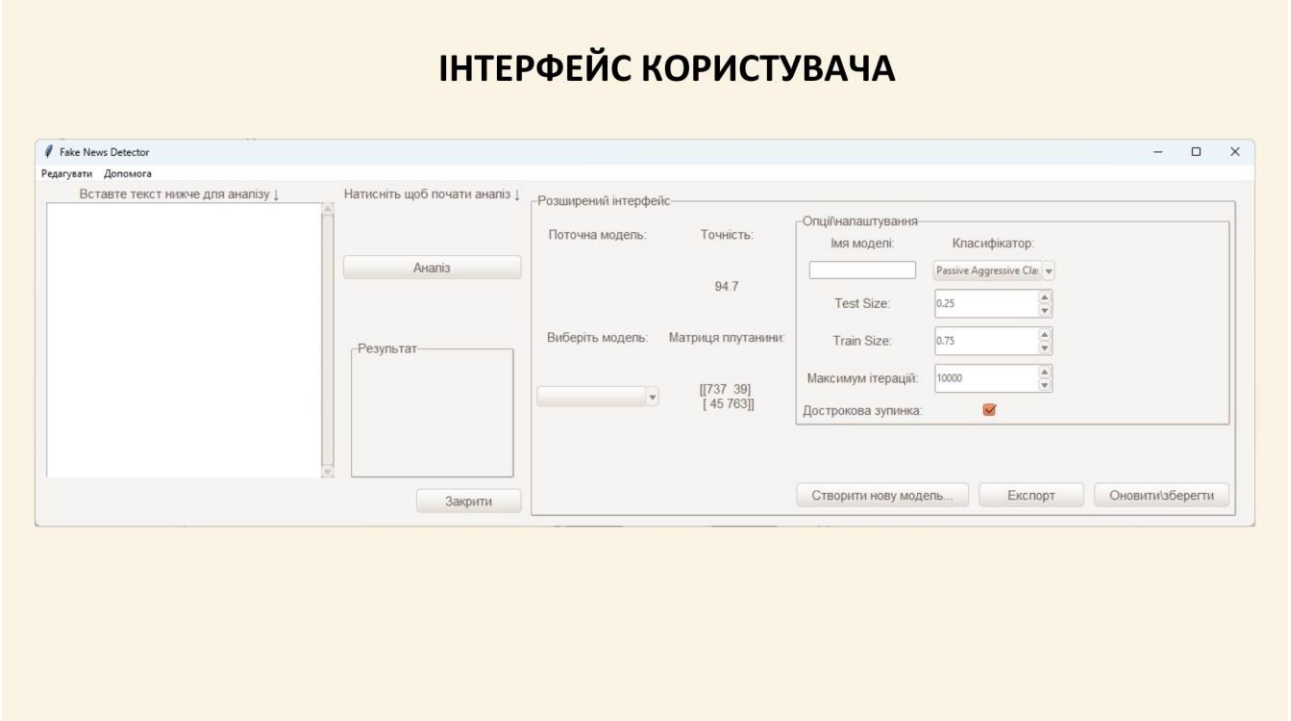

Рисунок Г.13 – Розширений інтерфейс користувача

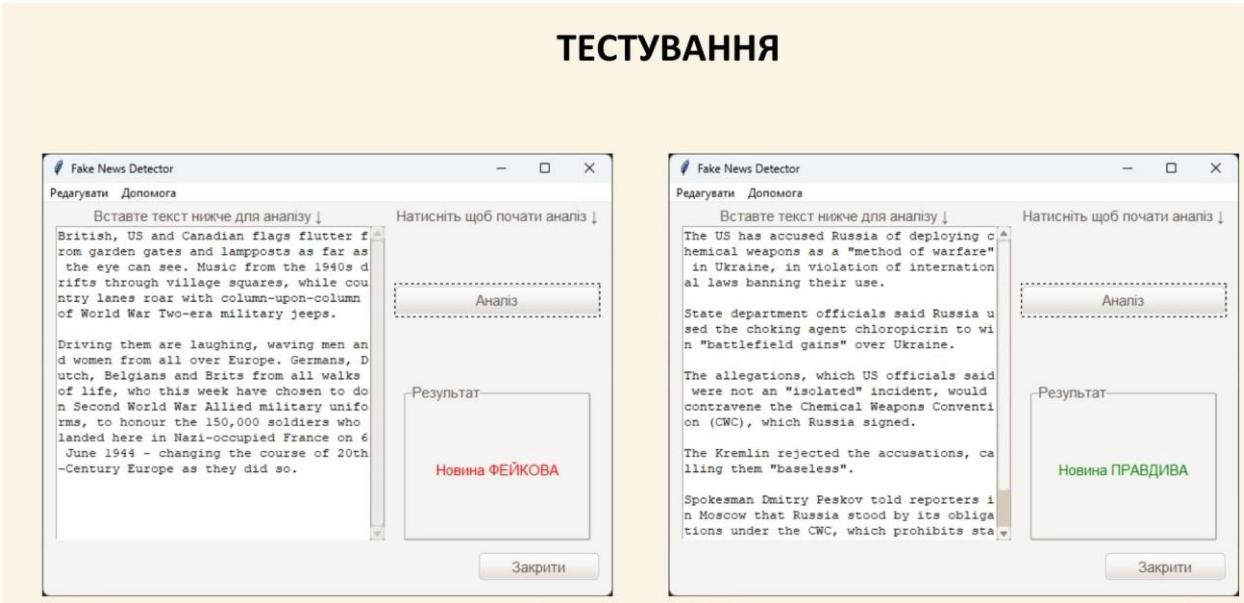

Рисунок Г.14 – Тестування

АПРОБАЦІЯ ТА ПУБЛІКАЦІЇ РЕЗУЛЬТАТІВ РОБОТИ

здійснені на конференції XXIV Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів (2024) і опубліковані в тезах доповіді цієї конференції

1. **Волосенко В. Ф., Чехместрук Р. Ю. Розробка програм для автоматичного виявлення фейкових новин у соціальних мережах** // Стан, досягнення та перспективи інформаційних систем і технологій: матеріали XXIV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів, Одеса, 18-19 квітня 2024 р. Одеса: Видавництво ОНТУ, 2024. 498 с.

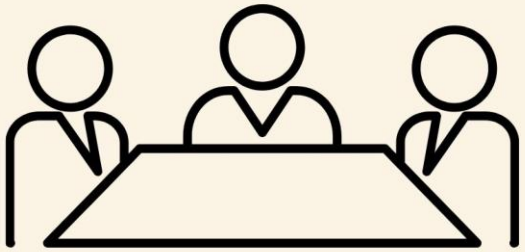

Рисунок Г.15 – Апробація та публікації результатів роботи

ВИСНОВКИ

В результаті було розроблено **програму автоматичного виявлення фейкових новин в соціальних мережах.**
Було досліджено предметну область виявлення фейкових новин в соціальних мережах та розглянуто аналоги програмного застосунку. Під час аналізу технологій та засобів розробки було обґрунтовано вибір в користь бібліотек **Tkinter, Scikit-learn** та інших.

У бакалаврській дипломній роботі вирішено наступні задачі:

- проведено аналіз стану технологій аналізу інформації;
- розроблено інтерфейс програмного застосунку;
- розроблено алгоритми для ефективної обробки великого обсягу даних з соціальних мереж;
- розроблено модуль класифікації тексту;
- створено модель машинного навчання для автоматичної класифікації новин на реальні та фейкові заснованої на їх характеристиках;
- проведено тестування розробленого програмного забезпечення.

✔ **Програма готова до роботи і має потенціал для подальшого розвитку та інтеграції з різними сервісами**

Рисунок Г.16 – Висновки