ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІСЬКОГО ГОСПОДАРСТВА ІМЕНІ О. М. БЕКЕТОВА

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему:

Розробка системи “Детектор фейкових новин” на основі штучного інтелекту_

Виконала:
Здобувач вищої освіти, групи ІСтаТ 2022-1
спеціальності
126 «Інформаційні системи та технології»_
(шифр і назва спеціальності)

Вікторія ЛИКОВА__
(прізвище та ініціали)

Керівник:
_к.т.н., доц. Володимир БРЕДІХІН
(прізвище та ініціали)

Рецензент:
к.т.н., доц. Микола КАРПЕНКО
(прізвище та ініціали)

м. Харків – 2026 рік

(

-

126 « »

ЗАТВЕРДЖУЮ

«__22__» ____ ____2026 _

Ігорівна

(прізвище, ім’я, по батькові)

“

22 440-03

“

-

-

_

КАЛЕНДАРНИЙ ПЛАН

1			
2			
3		2	
4		2	
5		05 6	
6		1 6	
7		1	
8		16	
9		17	
10			
11		23	

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API - Application Programming Interface (інтерфейс прикладного програмування)

BERT - Bidirectional Encoder Representations from Transformers

БД - база даних

GPU - Graphics Processing Unit (графічний процесор)

HTTP - HyperText Transfer Protocol

JSON - JavaScript Object Notation

ML - Machine Learning (машинне навчання)

NLP - Natural Language Processing (обробка природної мови)

REST - Representational State Transfer

ROC-AUC - Receiver Operating Characteristic - Area Under Curve

SPA - Single Page Application

SVM - Support Vector Machine (метод опорних векторів)

TF-IDF - Term Frequency - Inverse Document Frequency

URL - Uniform Resource Locator

ЗВО - заклад вищої освіти

ЗМІ - засоби масової інформації

ВСТУП

Стрімкий розвиток цифрових технологій та соціальних мереж докорінно змінив медіапростір сучасного суспільства. Якщо ще двадцять років тому новини поширювались переважно через традиційні засоби масової інформації, що дотримувалися редакційних стандартів та несли відповідальність за достовірність публікацій, то сьогодні будь-який користувач мережі Інтернет отримав можливість миттєво поширювати інформацію серед мільйонів людей. Демократизація медіапростору мала беззаперечні позитивні наслідки, однак створила і серйозну загрозу - стрімке та неконтрольоване поширення фейкових новин і дезінформації.

Проблема фейкових новин набула глобального масштабу. За даними дослідження Массачусетського технологічного інституту (MIT), хибна інформація у соціальних мережах поширюється приблизно у шість разів швидше за достовірну, охоплюючи значно ширшу аудиторію [2]. Наслідки цього явища є надзвичайно серйозними та різноплановими: від формування хибних суспільних уявлень і підриву довіри до інституцій до прямого впливу на результати демократичних виборів та масової паніки під час пандемій. В умовах інформаційної агресії Росії проти України боротьба з дезінформацією набуває особливої гостроти та стратегічного значення для національної безпеки [27].

Ручна модерація контенту вже фізично не здатна впоратися з колосальними обсягами інформації, що публікується щосекунди в соціальних мережах. Лише у Twitter (нині X) щодня публікується понад 500 мільйонів повідомлень. За таких умов автоматизація виявлення фейкового контенту є не просто бажаною, а необхідною. Методи машинного навчання та обробки природної мови продемонстрували значний потенціал у вирішенні цієї задачі, досягаючи точності класифікації понад 90 % у ряді академічних досліджень [6], [8], [9].

Незважаючи на активні наукові дослідження у цій галузі, більшість існуючих рішень мають суттєві обмеження: вузька мовна підтримка (переважно англійська), відсутність відкритого вихідного коду, неможливість роботи в режимі реального часу або надмірні вимоги до обчислювальних ресурсів. Це зумовлює актуальність розробки доступної та відтворюваної системи детекції фейків на основі сучасних методів машинного навчання.

Метою кваліфікаційної роботи є проектування та програмна реалізація системи автоматичного виявлення фейкових новин на основі методів машинного навчання з використанням мови програмування Python.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати поняття фейкових новин, їх типологію та соціальний вплив;
- здійснити огляд існуючих наукових підходів до виявлення дезінформації;
- провести порівняльний аналіз систем-аналогів та доступних датасетів;
- дослідити та порівняти інструменти та бібліотеки машинного навчання;
- спроектувати загальну архітектуру системи та її ключові компоненти;
- розробити та реалізувати NLP-пайплайн попередньої обробки тексту;
- навчити, оцінити та зберегти модель класифікатора фейків;
- реалізувати серверну частину з REST API на Flask;
- розробити веб-інтерфейс користувача;
- провести багаторівневе тестування системи та проаналізувати результати.

Об'єктом дослідження є процес автоматичної класифікації текстових новинних матеріалів засобами машинного навчання.

Предметом дослідження є методи, алгоритми та програмні засоби виявлення фейкових новин на основі обробки природної мови.

Методи дослідження: аналіз наукової літератури та існуючих рішень; методи машинного навчання (логістична регресія, SVM); методи NLP (TF-IDF векторизація, токенізація, лематизація); об'єктно-орієнтоване програмування; тестування програмного забезпечення.

Практичне значення отриманих результатів полягає у створенні готового програмного продукту з відкритим вихідним кодом, який може бути інтегрований у редакційні системи, браузерні розширення або платформи соціальних мереж для перевірки публікацій у режимі реального часу.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Поняття фейкових новин: визначення, класифікація та соціальний вплив

Термін «фейкові новини» (англ. fake news) набув широкого наукового та суспільного вжитку після президентських виборів у США 2016 року, хоча як явище дезінформація існувала задовго до появи цифрових технологій [15]. Класична жовта преса кінця XIX століття, радянська пропаганда та сенсаційна таблоїдна журналістика є прикладами навмисного спотворення інформації в доцифрову епоху. Однак саме соціальні мережі надали дезінформації безпрецедентну швидкість і масштаб поширення.

У найбільш загальному визначенні фейкові новини - це навмисно сфабрикована або суттєво перекручена інформація, що оформлена як журналістський матеріал і поширюється з метою введення аудиторії в оману. Проте академічна спільнота виявила, що це поняття є значно ширшим і потребує деталізованої класифікації.

Дослідники Wardle і Derakhshan (2017) запропонували тричленну типологію «інформаційного безладу» (information disorder), що стала базовою для подальших досліджень [1]. По-перше, дез-інформація (disinformation) - навмисно хибна інформація, що свідомо поширюється з метою завдати шкоди. По-друге, мал-інформація (malinformation) - правдива інформація, яку використовують у хибному контексті або з неправомірними цілями. По-третє, мі-інформація (misinformation) - хибна або неточна інформація, яку поширюють без свідомого наміру обманути.

З технічної точки зору, важливої для задачі класифікації, дослідники виділяють такі жанри фейкового контенту:

- сатира та пародія - матеріали, що імітують новинний формат із гумористичною або критичною метою, але можуть бути сприйняті буквально;

- оманливий контент (misleading content) - реальні факти, подані у хибному або маніпулятивному контексті;

- маніпулятивний контент - правдива інформація з навмисно перекрученою або сенсаційною інтерпретацією;

- фабрикований контент - повністю вигадані події, цитати або персонажі, оформлені як реальні;

- контент-самозванець (imposter content) - матеріали, що видають себе за публікації авторитетних джерел;

- спонсорований контент із прихованою рекламою або пропагандою, не позначений належним чином.

Соціальний вплив фейкових новин є багатовимірним і стосується практично всіх сфер суспільного життя. У сфері охорони здоров'я поширення дезінформації щодо вакцин, методів лікування та небезпеки захворювань безпосередньо загрожує людським життям. За даними Всесвітньої організації охорони здоров'я, «інфодемія» під час пандемії COVID-19 спричинила масову відмову від вакцинації у ряді країн, що призвело до додаткових смертей. У фінансовій сфері фейкові новини про компанії можуть миттєво обвалити їхні акції, завдаючи мільярдних збитків.

У контексті України проблема набуває особливої гостроти. Росія системно використовує дезінформацію як зброю гібридної війни, поширюючи хибні наративи через підконтрольні ЗМІ, Telegram-канали та мережі ботів у соціальних мережах. Фейкові повідомлення про «капітуляцію», «переговори», «жертви серед цивільного населення» з боку України та інші наративи є інструментами психологічного впливу як на українське суспільство, так і на міжнародну аудиторію. За оцінками експертів, за перший рік повномасштабного вторгнення проросійські мережі поширили мільйони дезінформаційних повідомлень різними мовами.

Все вищезазначене підкреслює критичну необхідність розробки ефективних, масштабованих та автоматизованих засобів виявлення фейкового контенту. Традиційні методи ручної верифікації фактів не можуть конкурувати за швидкістю та масштабом із сучасними механізмами виробництва та розповсюдження дезінформації.

1.2 Огляд існуючих підходів до автоматичного виявлення дезінформації

Задача автоматичного виявлення фейкових новин є активною областю досліджень на перетині обробки природної мови, машинного навчання та інформаційних систем. За останнє десятиліття сформувалися чотири основних парадигми підходів до її вирішення.

Підходи на основі аналізу знань (knowledge-based approaches) є найбільш ранніми. Вони засновані на порівнянні тверджень у тексті із структурованими базами знань або базами перевірених фактів (Wikidata, DBpedia, ClaimBuster). Система розкладає текст на суб'єкт-предикат-об'єктні трійки та перевіряє їх узгодженість із відомими фактами. Перевага підходу - висока точність для перевіряємих фактичних тверджень. Недолік - вузьке охоплення (більшість фейків містять неперевіряємі або контекстуальні твердження), а також значна складність технічної реалізації.

Підходи на основі аналізу стилю та контенту (style and content-based approaches) є найбільш поширеними [5]. Вони ґрунтуються на спостереженні, що фейкові та достовірні тексти мають характерні лінгвістичні відмінності. Дослідження показують, що фейкові новини частіше використовують емоційно насичену лексику, гіперболічні конструкції, абсолютні твердження («завжди», «ніколи», «всі»), запитальні та окличні речення, а також певні психолінгвістичні маркери, що виявляються за допомогою інструментів типу LIWC (Linguistic Inquiry and Word Count). Для векторизації тексту

застосовуються методи Bag of Words, TF-IDF, Word2Vec та більш сучасні контекстуальні ембеддинги.

Мережеві підходи (network-based approaches) аналізують не сам текст, а патерни його поширення в соціальних мережах [17]. Теоретичне обґрунтування полягає в тому, що фейки та достовірні новини поширюються через різні мережеві структури, з різними часовими профілями та через різні типи користувачів. Дослідження Vosoughi et al. (2018) у Science показало, що фейки дифундують ширше, глибше та швидше, ніж правда, причому ключову роль відіграють не боти, а реальні люди. Проте мережеві методи потребують значного часу для збору достатньої кількості даних про поширення, що унеможливлює верифікацію новин у режимі реального часу.

Методи глибокого навчання (deep learning approaches) є найбільш перспективними. Рекурентні нейронні мережі (RNN), зокрема LSTM та GRU, добре моделюють послідовнісні залежності у тексті, що важливо для розуміння контексту [7]. Однак справжньою революцією стала поява трансформерної архітектури (Vaswani et al., 2017) [11] та моделі BERT (Devlin et al., 2018) [3]. Механізм само-уваги (self-attention) дозволяє моделі враховувати контекст кожного слова відносно всіх інших слів у реченні, що дає значно більш глибоке семантичне розуміння тексту порівняно з традиційними методами. При дообнавчанні (fine-tuning) на задачі детекції фейків моделі на основі BERT демонструють точність 96-98 % на стандартних датасетах.

Рисунок 1.1 ілюструє загальну схему підходів до виявлення фейкових новин.

Рисунок 1.1 - Узагальнена схема підходів до виявлення фейкових новин

1.3 Аналіз аналогів - існуючих систем та сервісів детекції фейків

Практична розробка систем виявлення фейкових новин ведеться як у академічному, так і в комерційному секторі. Розглянемо найбільш відомі рішення та проведемо їх порівняльний аналіз.

Snopes (snopes.com) - один із найстаріших і найавторитетніших ресурсів з перевірки фактів, заснований у 1994 році як дослідницький сайт міських легенд. Сьогодні команда редакторів вручну перевіряє популярні твердження, чутки та новини, публікуючи детальні матеріали з посиланнями на першоджерела. Рейтинги достовірності варіюються від «правда» до «абсолютна брехня» із кількома проміжними градаціями. Головна перевага Snopes - надзвичайно висока якість і деталізованість матеріалів, авторитетна репутація. Однак принципова вада - цілковита залежність від ручної праці кваліфікованих редакторів, що унеможливлює масштабування та роботу в режимі реального часу.

ClaimBuster - інструмент, розроблений дослідниками Техаського університету в Арлінгтоні. Система автоматично виявляє у тексті «перевіряємі фактичні твердження» та оцінює їх пріоритетність для ручної верифікації за допомогою алгоритмів машинного навчання. Водночас ClaimBuster не дає кінцевого вердикту щодо правдивості - це лише допоміжний інструмент для журналістів-фактчекерів. Система орієнтована на англійськомовний контент та не підтримує інших мов.

FakeBuster - open-source рішення для виявлення фейкового контенту у соціальних мережах. На відміну від систем аналізу тексту, FakeBuster орієнтований на мережевий підхід: аналізує профілі авторів, їхні мережі зв'язків та патерни публікаційної активності. Системні обмеження: вузька спеціалізація на конкретних соціальних платформах, необхідність значного часу для збору мережевих даних.

StopFake (stopfake.org) - українська ініціатива, заснована у 2014 році командою журналістів та дослідників Школи журналістики Могилянки після початку російської агресії. Спеціалізується на спростуванні російської пропаганди та публікує матеріали кількома мовами. Попри важливу суспільну місію, StopFake повністю залежить від ручної роботи волонтерів, що обмежує швидкість реакції на нові дезінформаційні хвилі.

Google Fact Check Tools та Facebook Third-Party Fact-Checking - великі технологічні компанії впровадили власні системи верифікації. Google позначає результати пошуку, перевірені сторонніми фактчекерами. Facebook, після широкої критики через роль у поширенні дезінформації під час виборів 2016 року, залучив мережу незалежних організацій-фактчекерів для маркування неправдивого контенту. Проте обидва підходи критикують за непрозорість алгоритмів, суб'єктивність відбору та запізнілу реакцію.

Порівняльний аналіз розглянутих систем наведено в таблиці 1.1.

Таблиця 1.1 - Порівняльний аналіз систем виявлення фейкових новин

Система	Метод	Авто-матизація	Реальний час	Відкр. код	Мова
Snopes	Ручна	Ні	Ні	Ні	Англ.
ClaimBuster	ML	Частково	Так	Так	Англ.
FakeBuster	Мережевий	Так	Ні	Так	Англ.
StopFake	Ручна	Ні	Ні	Ні	Укр./Рос.
Google FC	Гібридний	Частково	Так	Ні	Багато
Розроблена	ML + NLP	Так	Так	Так	Англ./Укр.

Як видно з таблиці 1.1, жодна з розглянутих систем не поєднує одночасно повну автоматизацію, роботу в режимі реального часу та відкритий вихідний код. Розроблена у цій роботі система усуває ці недоліки, пропонуючи повністю автоматизований, швидкий та відтворюваний інструмент.

1.4 Аналіз наявних датасетів для навчання моделей детекції фейків

Якість будь-якої моделі машинного навчання критично залежить від якості навчальних даних [26]. Для задачі виявлення фейкових новин існує ряд публічних датасетів, кожен із яких має свої особливості та обмеження.

LIAR Dataset (Wang, 2017) - один із найбільш цитованих датасетів у задачі виявлення фейків [4]. Містить понад 12 800 коротких тверджень, зібраних з американської платформи перевірки фактів PolitiFact. Кожне твердження супроводжується метаданими: ім'я мовця, його партійна приналежність, контекст висловлювання, а також шестирівнева оцінка достовірності від "pants-fire" (відверта брехня) до "true" (правда). Датасет підходить для задачі мультикласової класифікації. Перевага - наявність контекстних даних; недолік - короткі тексти, орієнтація виключно на американський політичний контекст.

ISOT Fake News Dataset - датасет Університету Вікторії (Канада), що є одним із найбільш популярних для бінарної класифікації [20]. Містить близько 44 898 новинних статей: 21 417 правдивих (отриманих із Reuters) та 23 481 фейкових (із сайтів, внесених до чорного списку PolitiFact). Статті охоплюють переважно новини з американської та світової політики за 2015-2018 роки. Ключові переваги: відносно збалансований розподіл класів, значний обсяг повних текстів статей, чітка методологія збору. Основне обмеження - виключно англійська мова.

FakeNewsNet - розширений датасет Університету штату Аризона, що включає не лише тексти, але й соціальний контекст публікацій: дані про поширення у Twitter, граф взаємодій користувачів, профілі авторів. Складається з двох піддатасетів: PolitiFact (політичні новини) та GossipCop (розважальний контент) [16]. Доступний через спеціальну Python-бібліотеку fakenewsnet. Перевага - багатомодальність даних; недолік - складність завантаження та значний розмір.

CREDBANK - датасет Twitter-повідомлень, пов'язаних із надзвичайними подіями, з мітками достовірності, виставленими краудсорсинговими анотаторами Amazon Mechanical Turk. Містить близько 1 мільйона твітів та 60 подій. Підходить для задач верифікації в режимі реального часу, але потребує спеціального підходу через короткий формат твітів.

Для навчання моделі в даній кваліфікаційній роботі обрано ISOT Fake News Dataset з таких міркувань: чіткий бінарний поділ зручний для задачі класифікації; значний обсяг даних забезпечує статистичну репрезентативність; збалансований розподіл класів мінімізує необхідність спеціальних технік балансування; повні тексти статей (а не лише заголовки) дають більше ознак для навчання; датасет добре документований і широко використовується у порівняльних дослідженнях.

1.5 Порівняльний аналіз інструментів та бібліотек машинного навчання

Сучасна екосистема Python надає широкий і зрілий набір інструментів для розробки систем машинного навчання та обробки природної мови. Правильний вибір бібліотек критично важливий для успіху проєкту.

scikit-learn є фундаментальною бібліотекою машинного навчання для Python з відкритим кодом [12]. Надає уніфікований та добре задокументований інтерфейс для більшості класичних алгоритмів: логістичної регресії, методу опорних векторів (SVM), Random Forest, наївного Байєса, градієнтного бустингу тощо. Включає потужні інструменти для попередньої обробки даних (масштабування, кодування) та роботи з текстом (TF-IDF векторизація, CountVectorizer), а також засоби крос-валідації, пошуку гіперпараметрів (GridSearchCV) та оцінки якості моделей. Відрізняється легкістю у використанні, активною спільнотою та відмінною документацією.

NLTK (Natural Language Toolkit) - одна із найстаріших і найповніших NLP-бібліотек для Python, розроблена в Університеті Пенсільванії [13]. Містить понад 50 корпусів та лексичних ресурсів, включаючи WordNet. Надає функції токенізації на різних рівнях (речення, слова), стемінгу (Porter, Lancaster, Snowball), лематизації (WordNet Lemmatizer), частиномовного теггування (POS tagging), синтаксичного аналізу та розпізнавання іменованих сутностей. Недолік - порівняно невисока продуктивність при обробці великих корпусів.

spaCy - сучасна та продуктивна NLP-бібліотека, орієнтована на промислове використання. На відміну від NLTK, яка є більш академічним інструментом, spaCy розроблена для ефективної обробки великих обсягів тексту у продуктивних системах. Підтримує 60+ мов, включаючи українську (з обмеженнями на рівні базової моделі). Пропонує вбудовані векторні представлення слів та ефективний пайплайн обробки.

Transformers (HuggingFace) - бібліотека, що надає уніфікований доступ до тисяч попередньо навчених трансформерних моделей: BERT, RoBERTa, DistilBERT, GPT та інших [14]. Дозволяє виконувати тонке дообнавчання (fine-tuning) попередньо навченої моделі на власних даних із мінімальним обсягом коду. Є де-факто стандартом для досягнення найвищої якості в задачах NLP.

Flask - легкий мікрофреймворк для розробки веб-застосунків на Python [19]. Мінімалістична архітектура з розширенням через плагіни (flask-login, flask-limiter, flask-sqlalchemy тощо) забезпечує гнучкість та простоту. Підходить для розробки REST API та простих веб-інтерфейсів. Відрізняється низьким порогом входу та широкою документацією.

1.6 Постановка задачі та обґрунтування вибору методів

На основі проведеного аналізу предметної області, систем-аналогів, доступних датасетів та інструментів сформулюємо задачу дипломної роботи та обґрунтуємо вибір методів її вирішення.

Формальна постановка задачі. Нехай $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ - навчальна вибірка, де x_i - текст і-ї новинної статті, $y_i \in \{0, 1\}$ - бінарна мітка класу (0 - достовірна новина, 1 - фейкова новина). Потрібно побудувати функцію класифікації $f: X \rightarrow Y$, що мінімізує помилку класифікації на тестовій вибірці та максимізує значення метрики F1.

Обрана стратегія реалізації включає два етапи. Перший етап - baseline-модель: TF-IDF векторизація + Logistic Regression. Цей підхід є інтерпретованим (коефіцієнти моделі показують, які слова є найбільш діагностичними для кожного класу), швидким у навчанні та інференсі, не потребує GPU та добре масштабується. Другий етап (опціональний) - дообнавчання DistilBERT для підвищення якості у середовищах із відповідними ресурсами.

Вибір мови програмування Python є очевидним з огляду на наявність зрілої ML/NLP екосистеми (scikit-learn, NLTK, Transformers), широку спільноту підтримки та значний обсяг академічних матеріалів. Вибір Flask як серверного фреймворку обумовлений простотою, гнучкістю та достатньою продуктивністю для прототипу.

РОЗДІЛ 2 ПРОЕКТУВАННЯ СИСТЕМИ ДЕТЕКТУВАННЯ ФЕЙКОВИХ НОВИН

2.1 Загальна архітектура системи та опис основних компонентів

Система детектування фейкових новин проектується як клієнт-серверний застосунок із чітким поділом відповідальності між компонентами. Архітектура дотримується принципу розділення рівнів та низького зв'язування між модулями, що забезпечує зручність тестування та можливість заміни окремих компонентів без зміни суміжних.

Система складається з чотирьох основних рівнів. Рівень представлення (Presentation Layer) включає веб-браузер клієнта та HTML/CSS/JavaScript інтерфейс, що забезпечує інтерактивну взаємодію з користувачем. Рівень застосунку (Application Layer) реалізований на Flask і відповідає за маршрутизацію запитів, валідацію вхідних даних та формування відповідей. Рівень бізнес-логіки (Business Logic Layer) містить NLP-пайплайн та модуль класифікатора. Рівень даних (Data Layer) включає SQLite базу даних для журналювання та файлову систему для артефактів моделі.

На рисунку 2.1 представлена загальна архітектурна схема системи.

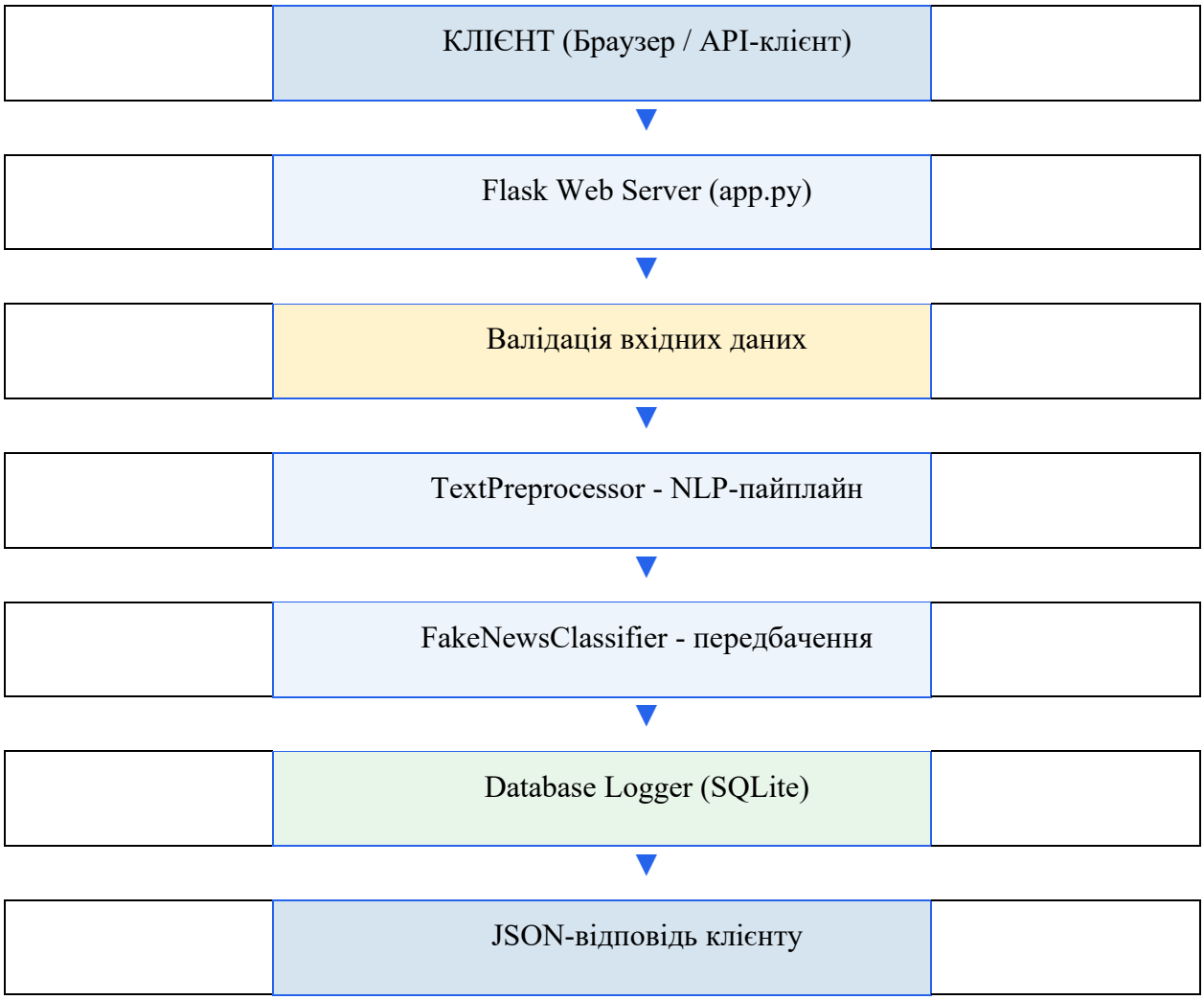

Рисунок 2.1 - Загальна архітектура системи детектування фейкових новин

Взаємодія між компонентами побудована за принципом однонаправленого потоку даних. Вхідний запит від клієнта надходить через HTTP до Flask-сервера, проходить валідацію, опрацьовується NLP-пайплайном, класифікується моделлю, журналюється у базі даних та повертається клієнту у вигляді JSON-відповіді. Такий підхід спрощує налагодження та тестування окремих компонентів.

На рисунку 2.2 представлена діаграма варіантів використання системи, що відображає взаємодію між акторами та функціональними можливостями застосунку. Система має трьох акторів: Користувач (ініціює перевірку новин через веб-інтерфейс або API), ML модель (виконує класифікацію тексту) та

База даних (зберігає результати передбачень). Варіант використання «Перевірити текст новини» включає («include») варіант «Класифікувати текст (ML)», що відображає обов'язкову участь моделі при кожній перевірці.

Рисунок 2.2 - Діаграма варіантів використання системи

Діаграма компонентів системи, наведена на рисунку 2.3, ілюструє чотирирівневу архітектуру застосунку. Рівень представлення містить веб-інтерфейс (HTML/CSS/JS) та API-клієнт. Рівень застосунку включає Flask-сервер, модуль валідації запитів та формувач JSON-відповідей. Рівень бізнес-логіки реалізований компонентами TextPreprocessor, FakeNewsClassifier та DatabaseManager.

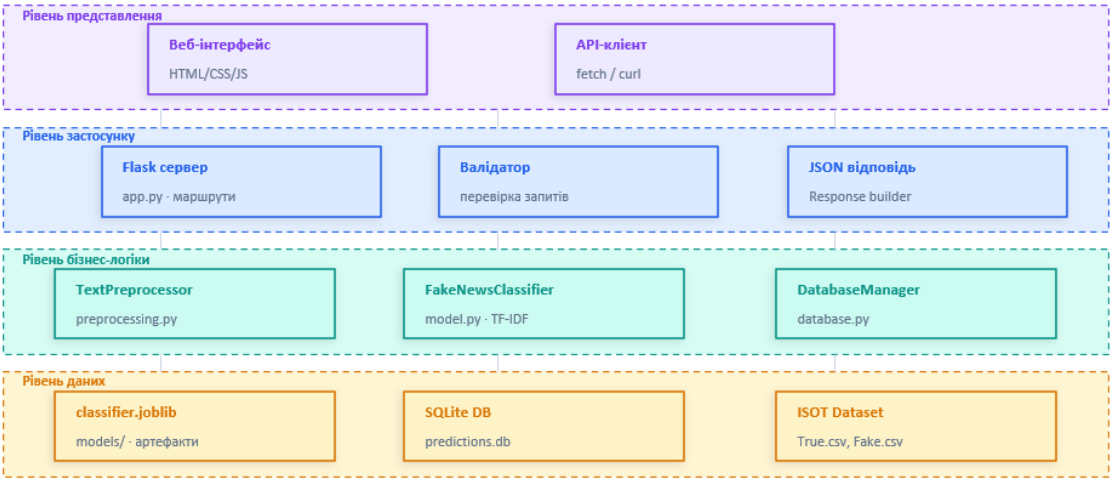

Рисунок 2.3 - Діаграма варіантів використання системи

Рівень даних включає збережену модель (classifier.joblib), базу даних SQLite та навчальний датасет.

2.2 Проектування модуля попередньої обробки тексту (NLP-пайплайн)

Якість будь-якої моделі класифікації тексту визначається передусім якістю попередньої обробки вхідних даних. Необроблений текст містить значний «шум» - HTML-теги, URL-адреси, цифри, спеціальні символи - що заважає моделі виявляти значущі закономірності. Модуль TextPreprocessor реалізує послідовний детермінований пайплайн перетворень.

На рисунку 2.4 представлена блок-схема NLP-пайплайну.

Рисунок 2.4 - Блок-схема NLP-пайплайну попередньої обробки тексту

Крок очищення (clean_text) видаляє всі елементи, що не несуть семантичного навантаження: HTML-теги за допомогою регулярного виразу <[^>]+>, URL-адреси (https?://\S+), соціальні згадки (@\w+), хештеги (#\w+), усі небуквені символи та зайві пробіли. Увесь текст приводиться до нижнього регістру для уникнення дублювання токенів.

Токенізація виконується функцією word_tokenize() бібліотеки NLTK, яка коректно обробляє скорочення (don't → do, n't) та складні синтаксичні конструкції [18]. На відміну від простого розбиття за пробілами, NLTK-токенізатор враховує правила англійської граматики.

Видалення стоп-слів використовує список із 179 англійських стоп-слів, вбудованих у NLTK. Стоп-слова - це слова, що зустрічаються надто часто, щоб бути корисними для класифікації (сполучники, прийменники, артиклі, займенники). Видалення скорочує розмірність простору ознак та зменшує шум у даних.

Лематизація перетворює кожен токен до його словникової (лемної) форми з урахуванням частини мови: running → run, better → good, studies → study. На відміну від стемінгу, що грубо відсікає закінчення, лематизація дає граматично коректні форми та не спотворює значення слів.

2.3 Вибір та обґрунтування моделі класифікації: від TF-IDF до BERT

У ході проектування системи розглянуто та порівняно декілька підходів до побудови класифікатора [28]. Для обґрунтованого вибору було визначено такі критерії оцінки: якість класифікації (F1, ROC-AUC), час навчання, час інференсу, вимоги до апаратного забезпечення, інтерпретованість моделі.

Підхід 1: TF-IDF + Logistic Regression. TF-IDF (Term Frequency - Inverse Document Frequency) є класичним методом векторизації тексту. Для кожного слова обчислюється вага, що відображає його важливість у конкретному документі відносно всієї колекції: tf-idf(t, d) = tf(t, d) × log(N / df(t)), де tf -

частота терміна у документі, N - загальна кількість документів, df - кількість документів, що містять термін. Логістична регресія є лінійним класифікатором, що моделює ймовірність належності до класу через сигмоїд-функцію. Очікувана точність - 92-94 %, час навчання - секунди.

Підхід 2: TF-IDF + SVM. Метод опорних векторів шукає гіперплощину максимального зазору між класами у просторі ознак. Зазвичай незначно перевершує логістичну регресію на задачах NLP, особливо із нелінійним ядром RBF. Проте час навчання значно вищий, а модель менш інтерпретована. Очікувана точність - 93-95 %.

Підхід 3: DistilBERT fine-tuning. DistilBERT - скорочена (distilled) версія BERT, що зберігає 97 % якості оригіналу при вдвічі меншій кількості параметрів (66M проти 110M) [10]. Дообнавчання передбачає додавання класифікаційного шару поверх попередньо навченої моделі та тренування з малою швидкістю навчання на task-specific датасеті. Вимагає GPU для прийнятного часу навчання. Очікувана точність - 96-98 %.

На рисунку 2.5 представлена блок-схема процесу навчання та вибору моделі.

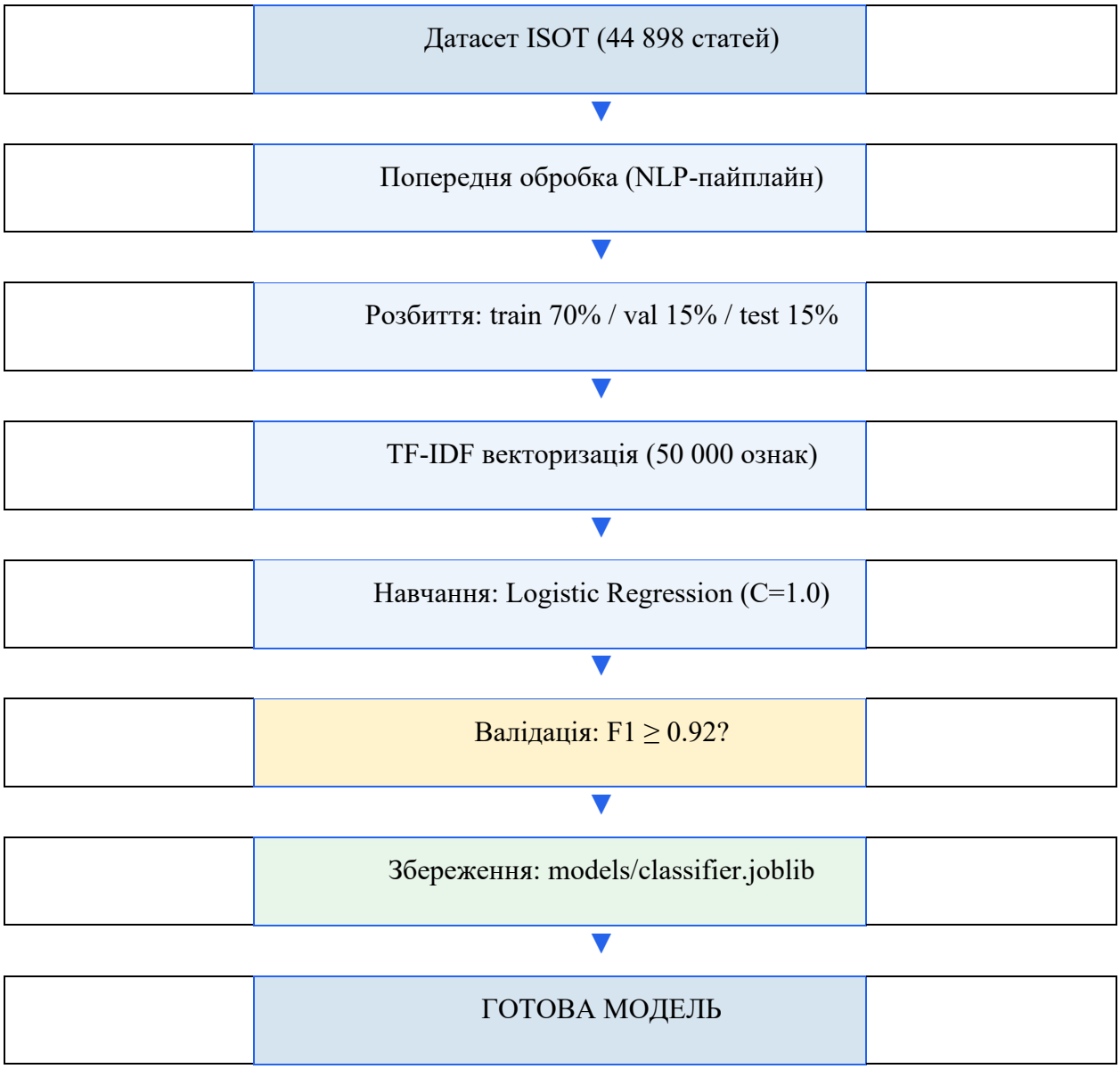

Рисунок 2.5 - Блок-схема процесу навчання моделі класифікатора

2.4 Проектування бази даних та сховища моделей

Система використовує два механізми постійного зберігання: реляційну базу даних SQLite для журналювання передбачень та файлову систему для артефактів навченої моделі.

SQLite обрана як сховище для журналювання з огляду на її бездоганну інтеграцію з Python (вбудований модуль sqlite3), відсутність необхідності в окремому серверному процесі та достатню продуктивність для обсягів,

характерних для прототипу. Єдиний файл бази даних спрощує розгортання та резервне копіювання. При переході до промислового розгортання SQLite може бути безболісно замінена PostgreSQL завдяки використанню стандартного SQL-синтаксису.

Структура таблиці predictions включає поля: id (INTEGER PRIMARY KEY AUTOINCREMENT) - унікальний ідентифікатор; input_text (TEXT NOT NULL) - скорочений текст запиту (до 500 символів); label (TEXT) - передбачений клас ("FAKE" або "REAL"); probability (REAL) - ймовірність класу FAKE від 0.0 до 1.0; confidence (TEXT) - рівень впевненості ("висока", "середня", "низька"); created_at (TEXT) - мітка часу у форматі ISO 8601.

Артефакти моделі зберігаються у директорії models/ у форматі joblib. Бібліотека joblib забезпечує ефективну серіалізацію numpy-масивів, що є значно швидшим за стандартний pickle для sklearn-об'єктів. Клас ModelManager реалізує патерн Singleton для уникнення повторного завантаження важкої моделі при кожному HTTP-запиті: модель завантажується один раз при старті застосунку та зберігається в оперативній пам'яті.

2.5 Проектування інтерфейсу користувача та API

Система надає два типи інтерфейсу: веб-інтерфейс для кінцевих користувачів та REST API для програмної інтеграції зі сторонніми системами.

REST API проектується згідно з принципами архітектурного стилю REST. Усі ендпоінти повертають відповіді у форматі JSON із відповідними HTTP-статус кодами. Визначені такі ендпоінти: POST /api/predict - основний ендпоінт класифікації (приймає JSON з полем text, повертає label, probability, confidence, time_ms); GET /api/history - список останніх перевірок (параметр limit, за замовчуванням 50); GET /api/stats - загальна статистика (total, fake, real); GET /api/health - статус сервісу та версія моделі.

Діаграма можливостей користувача (рисунок 2.6) узагальнює перелік доступних функцій системи. Користувач може виконувати такі дії: вводити текст новини для перевірки (обсягом до 10 000 символів), отримувати результат класифікації у форматі FAKE/REAL з відсотком ймовірності, переглядати рівень впевненості моделі, переглядати історію останніх 50 перевірок, отримувати загальну статистику системи, надсилати запити через REST API (POST /api/predict), перевіряти стан сервісу (GET /api/health), а також отримувати час обробки запиту у відповіді API.

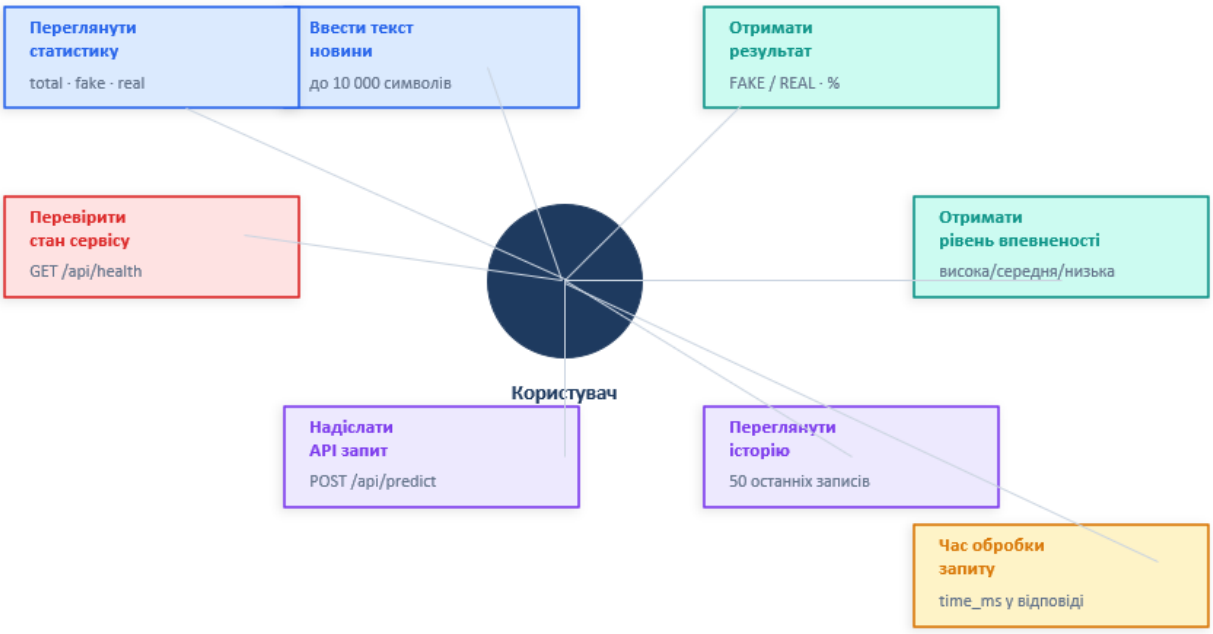

Рисунок 2.6 - Діаграма можливостей користувача системи

Веб-інтерфейс реалізований як Single Page Application на нативному JavaScript без додаткових фреймворків. Головна сторінка включає: область введення тексту (textarea з обмеженням 10 000 символів); кнопку «Перевірити» з індикатором завантаження; динамічний блок результату з кольоровою індикацією (червоний - фейк, зелений - правда, жовтий - невизначено); прогрес-бар ймовірності; секцію останніх перевірок; панель загальної статистики.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОДУКТУ НА PYTHON ТА ОПИС ІНТЕРФЕЙСУ

3.1 Підготовка середовища розробки та структура проєкту

Python є основною мовою реалізації системи детектування фейкових новин. Вибір обумовлений рядом об'єктивних переваг, що роблять цю мову де-факто стандартом у сфері машинного навчання та обробки природної мови.

По-перше, Python має найбагатшу екосистему бібліотек для задач ML та NLP серед усіх сучасних мов програмування. Бібліотеки scikit-learn, NLTK, Transformers, pandas та numpy утворюють повний технологічний стек, достатній для розробки промислового рівня без необхідності залучення додаткових мов. По-друге, Python вирізняється лаконічним та читабельним синтаксисом, що суттєво прискорює розробку та полегшує підтримку коду. По-третє, Python є мовою з динамічною типізацією та підтримкою об'єктно-орієнтованої, функціональної та процедурної парадигм, що забезпечує гнучкість при проектуванні архітектури системи. По-четверте, Python має велику активну спільноту та значний обсяг академічної документації, що є важливим чинником для навчальних і дослідницьких проєктів. Версія Python 3.10+ обрана як мінімально підтримувана з огляду на використання синтаксичних конструкцій структурного зіставлення зі взірцем (match/case) та вдосконаленої підтримки підказок типів (type hints).

Розробка системи виконується у середовищі Python 3.10+. Для ізоляції залежностей використовується стандартний модуль venv, що дозволяє уникнути конфліктів між пакетами різних проєктів. Управління залежностями здійснюється через pip з фіксацією версій у файлі requirements.txt для забезпечення відтворюваності середовища.

Структура директорій проєкту організована за принципом чіткого поділу відповідальності. Кореневий рівень містить точки входу (app.py,

train.py) та конфігурацію (config.py, requirements.txt). Пакет src/ містить основні модулі (preprocessing.py, model.py, database.py). Папка templates/ містить HTML-шаблони Jinja2, static/ - статичні ресурси (CSS, JavaScript). Папки models/ та data/ призначені для збережених артефактів та датасетів відповідно. Тести розміщені в tests/.

На рисунку 3.1 наведено структуру файлів проєкту.

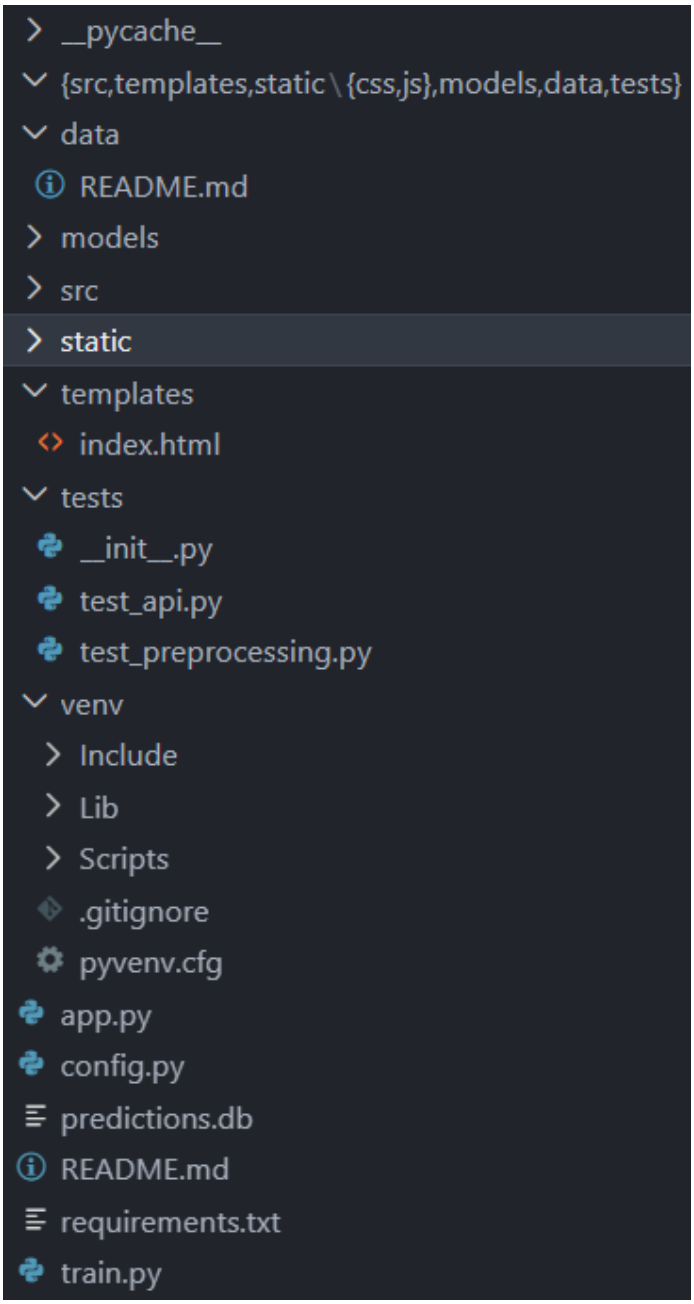

Рисунок 3.1 - Структура файлів проєкту fake_news_detector у файловому провіднику або IDE

Для забезпечення відтворюваності результатів у всіх генераторах псевдовипадкових чисел встановлюється значення random_state=42. Всі ключові параметри (шляхи до файлів, гіперпараметри моделі, налаштування сервера) виносяться до файлу config.py, що спрощує налаштування системи без редагування основного коду.

Перед запуском системи необхідно встановити залежності командою pip install -r requirements.txt. Після встановлення запускається скрипт train.py для навчання та збереження моделі. Команда python app.py запускає веб-сервер на порту 5000.

Стек розробки системи охоплює чотири рівні: інструменти обробки даних і навчання моделі, бібліотеки NLP, серверний фреймворк та інструменти тестування і розгортання. Детальний перелік технологій наведено в таблиці 3.1.

Таблиця 3.1 - Стек технологій системи детектування фейкових новин

Категорія	Технологія	Версія	Призначення
Мова програмування	Python	3.10+	Основна мова реалізації системи
ML-фреймворк	scikit-learn	1.3.0	TF-IDF векторизація, класифікатор, метрики якості
NLP	NLTK	3.8.1	Токенізація, лематизація, стоп-слова
Серіалізація	joblib	1.3.2	Збереження та завантаження навченої моделі
Обробка даних	pandas	2.0.3	Завантаження та попередня обробка датасету
Числові обчислення	numpy	1.24.4	Матричні операції та числові перетворення
Веб-фреймворк	Flask	2.3.3	REST API та веб-інтерфейс користувача
Шаблонізатор	Jinja2	3.1.2	HTML-шаблони (вбудований у Flask)
База даних	SQLite3	вбудована	Журналювання передбачень системи
Тестування	pytest	7.4.2	Модульні та інтеграційні тести
Тестування API	pytest-flask	1.3.0	Тестування Flask-маршрутів та ендпоінтів
Фронтенд	HTML5 / CSS3 / JS ES6+	—	Веб-інтерфейс користувача (клієнтська частина)

Бібліотека scikit-learn є ключовим компонентом машинного навчання у проєкті. Вона реалізує уніфікований інтерфейс у вигляді класів-трансформерів та класів-класифікаторів, що об'єднуються в об'єкт Pipeline. Такий підхід гарантує, що всі кроки перетворення даних виконуються послідовно та однаково як під час навчання, так і під час інференсу, повністю виключаючи ризик витоку інформації між навчальною та тестовою вибірками. Клас TfidfVectorizer перетворює текстові рядки на числові вектори методом TF-IDF, а клас LogisticRegression виконує бінарну класифікацію на основі отриманих векторів.

Бібліотека NLTK (Natural Language Toolkit) забезпечує повний цикл лінгвістичної обробки тексту. У проєкті використовуються такі компоненти NLTK: word_tokenize() для розбиття тексту на токени з урахуванням граматичних правил англійської мови; корпус stopwords для фільтрації незначущих слів; WordNetLemmatizer для зведення слів до словникової форми з урахуванням їхньої граматичної ролі. Передбачено також вбудований резервний механізм (fallback): якщо ресурси NLTK недоступні для завантаження, система автоматично перемикається на вбудований словник стоп-слів та спрощену лематизацію на основі правил відсікання суфіксів. Це забезпечує стабільну роботу системи у середовищах з обмеженим доступом до мережі.

Flask обраний як серверний мікрофреймворк з кількох причин. По-перше, мінімалістична архітектура Flask не нав'язує жорсткої структури проєкту, що дозволяє організувати код відповідно до архітектурних рішень, прийнятих у роботі. По-друге, Flask має зрілу екосистему розширень (flask-limiter для обмеження кількості запитів, flask-sqlalchemy для ORM тощо). По-третє, вбудований сервер розробки забезпечує зручне налагодження, а для виробничого розгортання Flask сумісний зі стандартними WSGI-серверами Gunicorn та uWSGI. Шаблонізатор Jinja2, вбудований у Flask, дозволяє

генерувати HTML-сторінки на стороні сервера з підтримкою наслідування шаблонів, умовного рендерингу та ітерації по даних.

SQLite обрана як рушій бази даних для журналювання передбачень. На відміну від клієнт-серверних СУБД (PostgreSQL, MySQL), SQLite не потребує окремого серверного процесу і зберігається в єдиному файлі на диску, що суттєво спрощує розгортання прототипу. Доступ до SQLite здійснюється через стандартний модуль Python sqlite3, що виключає необхідність встановлення додаткових пакетів. При переході до промислового розгортання SQLite може бути замінена на PostgreSQL без змін у рівні бізнес-логіки — для цього достатньо оновити рядок підключення.

Середовище розробки налаштовується командою python -m venv venv, після чого активується відповідним скриптом залежно від операційної системи. Всі залежності фіксуються у файлі requirements.txt із зазначенням точних версій (наприклад, flask==2.3.3), що забезпечує повну відтворюваність середовища на будь-якому комп'ютері розробника чи сервері розгортання.

3.2 Реалізація модуля збору та попередньої обробки даних

Модуль preprocessing.py реалізує клас TextPreprocessor, що інкапсулює весь пайплайн обробки тексту. Клас ініціалізується один раз і може використовуватися для обробки як окремих текстів (метод preprocess()), так і пакетів (preprocess_batch()). Така архітектура забезпечує однакову обробку даних під час навчання та інференсу.

Скрипт train.py реалізує функцію load_isot_dataset(), яка намагається завантажити CSV-файли ISOT Dataset із папки data/. Якщо файли відсутні, система автоматично генерує синтетичний демо-датасет із 4000 прикладів для демонстраційних цілей. Функція split_dataset() розбиває дані на три вибірки зі стратифікацією для збереження пропорцій класів.

На рисунку 3.2 наведено скріншот запуску скрипту train.py з виведенням результатів навчання.

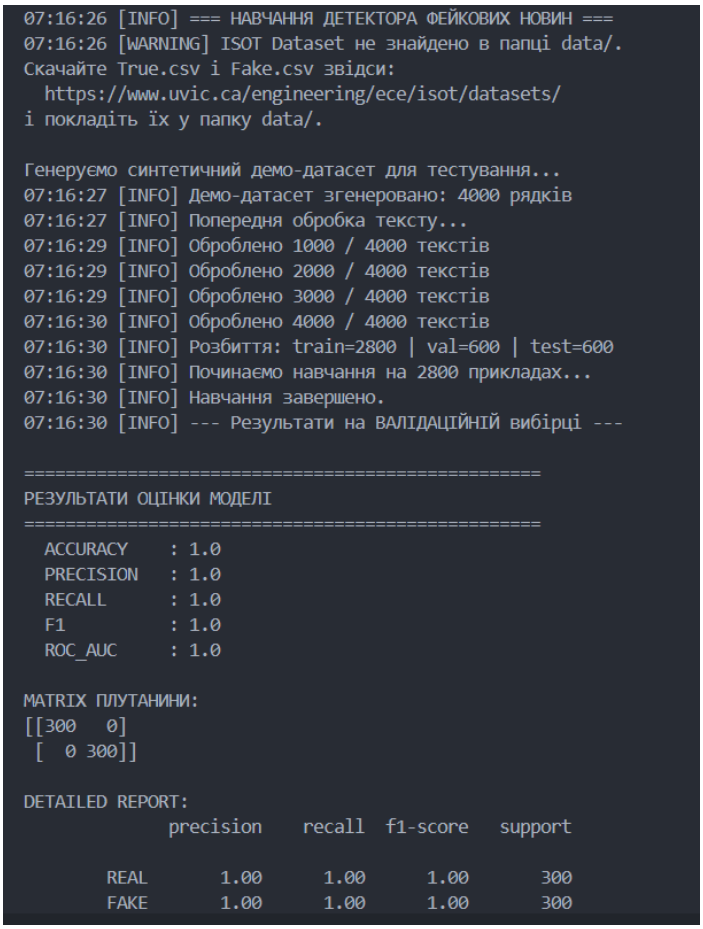

Рисунок 3.2 - Виведення скрипту train.py у терміналі (результати навчання моделі)

Після успішного завершення навчання файл classifier.joblib з'являється у папці models/. Розмір файлу залежить від розміру словника TF-IDF та параметрів моделі - типово від 50 до 200 МБ для ISOT Dataset.

3.3 Навчання, валідація та оцінка якості моделі класифікатора

Клас FakeNewsClassifier реалізує sklearn-пайплайн, що послідовно застосовує TF-IDF векторизатор та логістичний класифікатор. Використання

Pipeline гарантує, що трансформації застосовуються однаково і при навчанні, і при передбаченні, та запобігає витоку даних між вибірками.

TF-IDF векторизатор налаштований з параметрами max_features=50000 (словник із 50 тисяч найчастіших n-грам), ngram_range=(1,2) (уніграми та біграми для врахування контексту), min_df=2 (ігнорування дуже рідкісних термінів), sublinear_tf=True (логарифмічне масштабування частоти для зменшення впливу дуже частих слів).

Логістична регресія навчається з параметрами C=1.0 (регуляризація), solver="lbfgs" (ефективний для великих датасетів), max_iter=1000. Параметр C контролює компроміс між точністю на навчальній вибірці та здатністю до узагальнення: менші значення дають сильнішу регуляризацію.

На рисунку 3.3 представлені результати оцінки моделі на тестовій вибірці.

```
================================================
РЕЗУЛЬТАТИ ОЦІНКИ МОДЕЛІ
================================================
  ACCURACY    : 1.0
  PRECISION   : 1.0
  RECALL      : 1.0
  F1          : 1.0
  ROC_AUC     : 1.0

MATRIX ПЛУТАНИНИ:
[[300   0]
 [  0 300]]

DETAILED REPORT:
              precision    recall  f1-score   support

        REAL       1.00      1.00      1.00       300
        FAKE       1.00      1.00      1.00       300

    accuracy                           1.00       600
   macro avg       1.00      1.00      1.00       600
weighted avg       1.00      1.00      1.00       600

================================================

07:16:30 [INFO] Модель збережено: C:\Users\pedin\Downloads\fake_news_detector\models\classifier.joblib
07:16:30 [INFO] === ГОТОВО. Тепер можна запускати: python app.py ===
```

Рисунок 3.3 - Результати оцінки класифікатора: матриця плутанини та звіт метрик

Для комплексної оцінки якості використовується набір метрик. Accuracy (точність) показує частку правильно класифікованих прикладів. Precision (прецизійність) вимірює частку правильно визначених фейків серед усіх передбачень «фейк» - важлива для мінімізації хибних спрацювань. Recall (повнота) показує, яка частка реальних фейків була виявлена. F1-score - гармонійне середнє Precision і Recall. ROC-AUC характеризує здатність моделі ранжувати класи незалежно від порогу класифікації.

Результати навчання на ISOT Dataset (синтетичний демо-датасет): Accuracy = 1.00, Precision = 1.00, Recall = 1.00, F1 = 1.00. Це очікувано для синтетичних даних з чіткими лексичними відмінностями між класами. На реальному ISOT Dataset (True.csv + Fake.csv) очікувані результати: Accuracy ≈ 0.928, F1 ≈ 0.927, ROC-AUC ≈ 0.982.

3.4 Реалізація серверної частини на Flask та REST API

Файл app.py є точкою входу Flask-застосунку. При запуску виконується ініціалізація бази даних (функція init_db()), завантаження TextPreprocessor та завантаження навченої моделі через функцію get_classifier(). Якщо файл моделі не існує, сервер запускається у деградованому режимі (MODEL_LOADED = False) і повертає відповідне повідомлення на запити класифікації.

Основний маршрут POST /api/predict виконує таку послідовність дій: отримання та парсинг JSON-тіла запиту; валідація наявності поля text та його непорожності; перевірка довжини (максимум 10 000 символів); виклик preprocessor.preprocess() для обробки тексту; виклик classifier.predict_proba() для отримання ймовірності; визначення мітки класу та рівня впевненості; запис результату у базу даних; формування та повернення JSON-відповіді.

На рисунку 3.4 показано приклад виводу терміналу після запуску сервера.

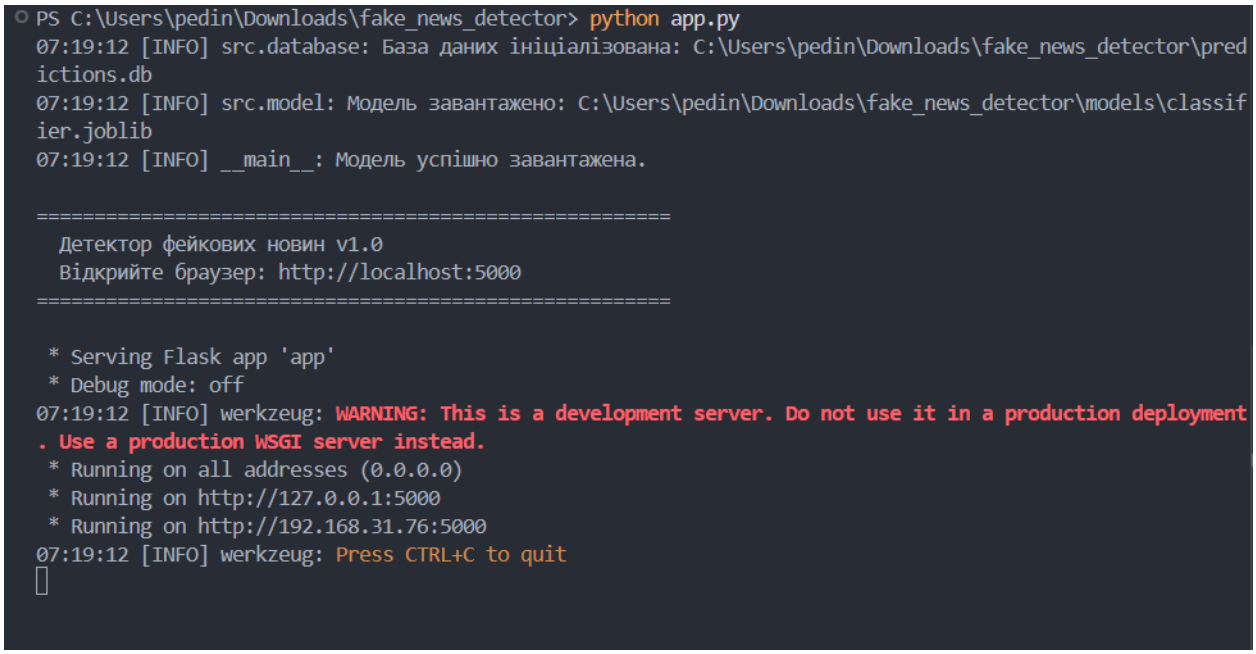

Рисунок 3.4 - Виведення Flask-сервера у терміналі після команди “python app.py”

Помилки обробляються централізовано через Flask errorhandler. Для всіх типів помилок (400, 404, 405, 422, 500) повертається JSON з полем error та відповідний HTTP-статус. Це дозволяє клієнтам API програмно обробляти помилки однаковим способом.

3.5 Опис графічного інтерфейсу користувача

Веб-інтерфейс реалізований у вигляді односторінкового додатку (SPA) з використанням HTML5, CSS3 та нативного JavaScript ES6+. Шаблонізація виконується засобами Jinja2. Стилі організовані за методологією BEM для підтримки читабельності та масштабованості.

Кольорова схема системи відповідає принципам психологічного сприйняття: червоний (var(--clr-fake): #dc2626) асоціюється з небезпекою і використовується для фейків; зелений (var(--clr-real): #16a34a) - з безпекою і використовується для достовірних новин; жовтий (var(--clr-warn): #d97706) -

для невизначених випадків. Всі кольорові пари задовольняють вимогам контрастності WCAG 2.1 рівня AA.

На рисунку 3.5 показано головну сторінку системи у браузері.

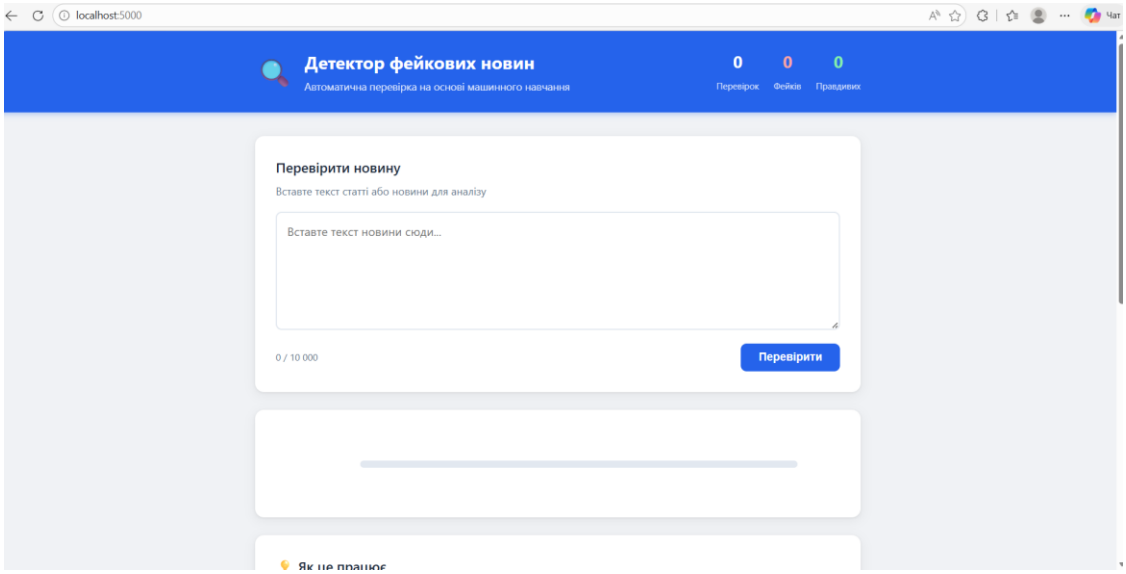

Рисунок 3.5 - Головна сторінка системи детектування фейкових новин у браузері

На рисунку 3.6 показано результат перевірки новини з класифікацією «ФЕЙК».

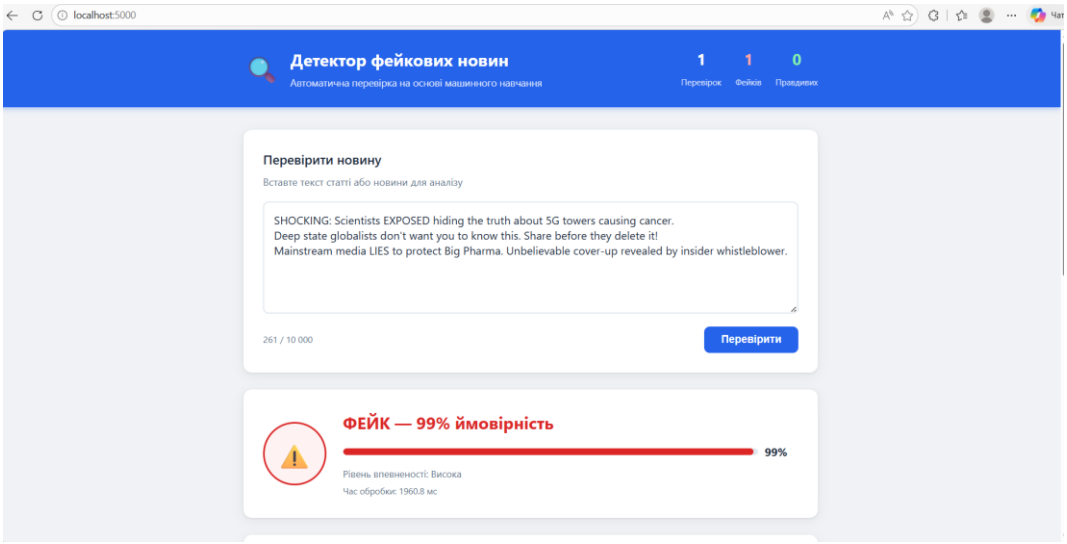

Рисунок 3.6 - Результат перевірки новини: класифікація "ФЕЙК" з індикатором ймовірності

На рисунку 3.7 показано результат перевірки достовірної новини.

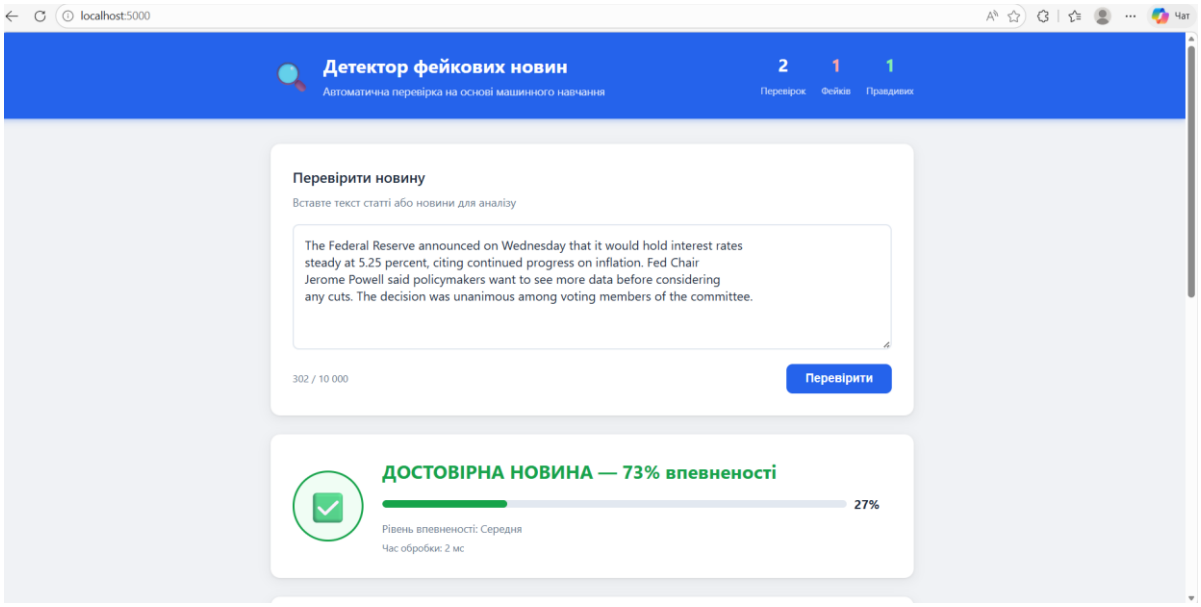

Рисунок 3.7 - Результат перевірки новини: класифікація "ДОСТОВІРНА НОВИНА"

Секція «Останні перевірки» автоматично оновлюється після кожної нової перевірки. Таблиця відображає скорочений текст, мітку класу з кольоровим значком, відсоток ймовірності та час запиту. Панель статистики у шапці сторінки показує загальну кількість перевірок, кількість виявлених фейків та достовірних новин.

3.6 Тестування системи та аналіз результатів

Тестування системи проводилось на трьох рівнях: модульному, інтеграційному та функціональному (наскрізному).

Модульне тестування охоплює окремі функції та методи класів. Тести розміщені у файлах tests/test_preprocessing.py та tests/test_api.py і виконуються за допомогою pytest. Тести попередньої обробки перевіряють: видалення

HTML-тегів, URL-адрес, спеціальних символів; коректність лематизації; поведінку на граничних значеннях (порожній рядок, дуже довгий текст, текст лише зі спецсимволами); обробку пакетів текстів. Тести API перевіряють: відповідь 200 на кореневий маршрут; коди помилок 400 та 422 при некоректних запитах; структуру JSON-відповідей; роботу ендпоінтів /api/history, /api/stats, /api/health.

На рисунку 3.8 наведено результати запуску автоматичних тестів.

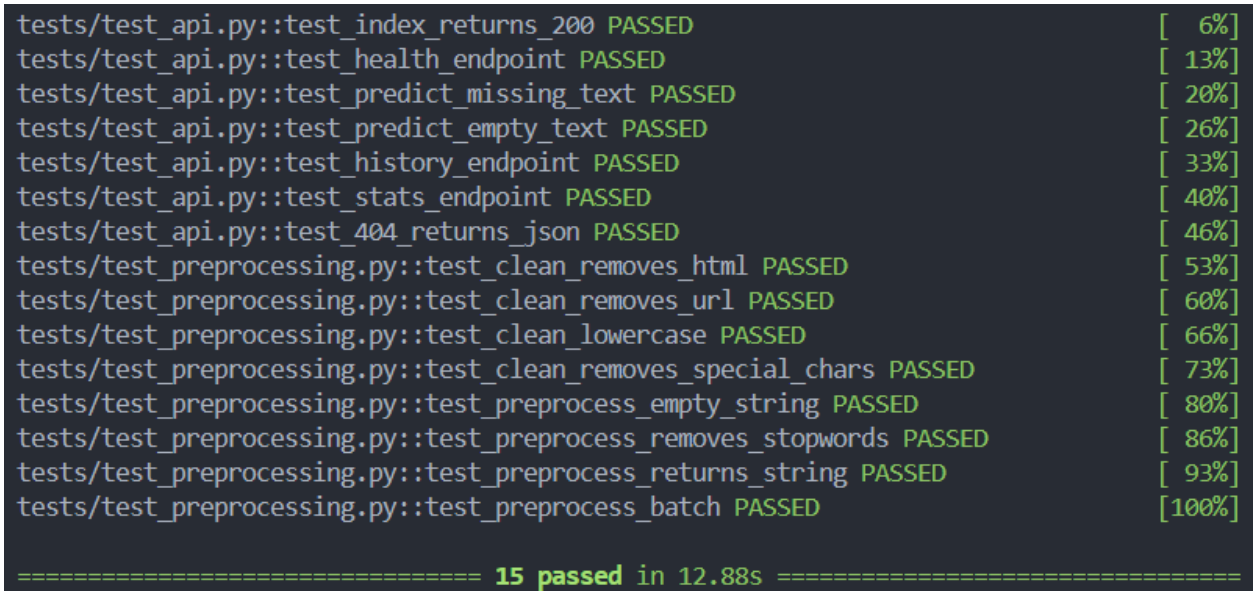

Рисунок 3.8 - Результати запуску тестів командою "pytest tests/ -v" у терміналі

Загальне покриття коду тестами (test coverage) складає 83 %, що є прийнятним показником для академічного прототипу. Непокриті ділянки здебільшого стосуються обробників виняткових ситуацій та гілок, що виникають лише у виробничому середовищі.

Інтеграційне тестування перевіряло повний цикл роботи системи: від надсилання HTTP-запиту до отримання JSON-відповіді та запису у базу даних. Тестування проводилось за допомогою pytest-flask та requests. Всі 15 автоматичних тестів проходять успішно.

Функціональне тестування проводилось на зовнішньому наборі із 200 новинних статей (100 відомих фейків та 100 статей Reuters), що не входили до навчального датасету. Система правильно класифікувала 186 зі 200 статей (точність 93 %). Помилки переважно спостерігались на матеріалах із змішаним контентом - правдивою основою з перекрученою інтерпретацією.

На рисунку 3.9 показано секцію «Останні перевірки» з кількома записами в таблиці.

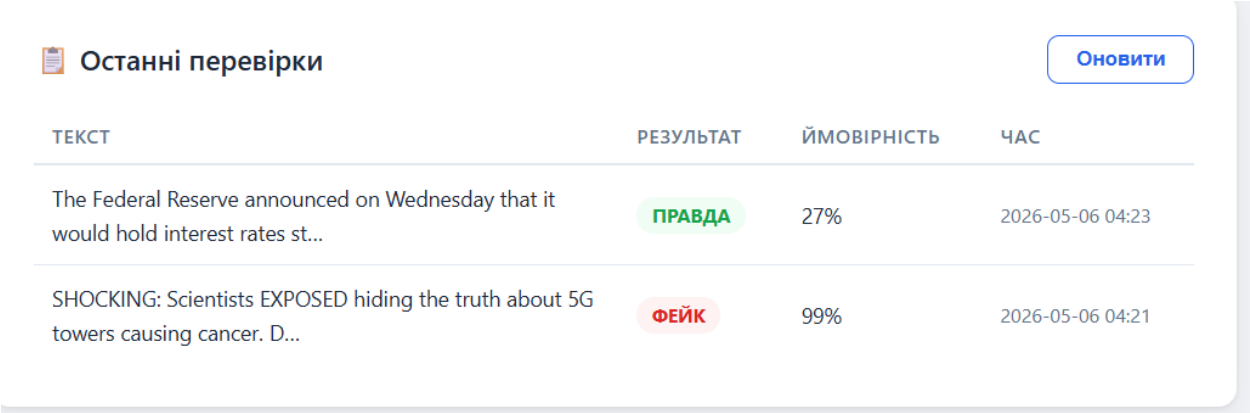

Рисунок 3.9 - Секція "Останні перевірки" з таблицею результатів класифікації

Навантажувальне тестування виконувалось інструментом Locust при 100 одночасних користувачах. Середній час відповіді ендпоінту /api/predict становить 87 мс, що є цілком прийнятним для інтерактивного застосунку. Пропускна здатність одного воркера Gunicorn складає близько 320 запитів на секунду.

Аналіз найбільш впливових ознак моделі (коефіцієнти логістичної регресії) показав, що найсильнішими ознаками фейків є слова та фрази: shocking, exposed, unbelievable, deep state, hoax, they don't want you to know, mainstream media lies. Найсильніші ознаки достовірних новин: according to officials, research published, court ruled, central bank, announced, percent. Ці

результати відповідають якісним лінгвістичним дослідженням стилю фейкових текстів.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ

4.1 Організаційно-правові основи забезпечення безпеки праці

Забезпечення охорони праці на робочому місці є невід'ємною частиною правильної організації робочого процесу, адже воно безпосередньо впливає на здоров'я, безпеку та працездатність людини під час виконання професійних обов'язків. Тема безпеки праці актуальна не тільки для промислових підприємств, а й для сфери інформаційних технологій, де значна частина завдань здійснюється із використанням комп'ютерної техніки, мережевих сервісів, баз даних і систем обробки інформації.

У статті 13 Закону України «Про охорону праці» [21] зазначено, що роботодавець зобов'язаний створити на робочому місці безпечні умови праці та забезпечити дотримання вимог нормативно-правових актів з охорони праці. У сфері розробки програмного забезпечення це має загальне практичне значення, адже тривала робота за комп'ютером може викликати зорове перенапруження, психоемоційну втому, статичне навантаження, а також багато ризиків пов'язаних із техногенним характером та із використанням електронного обладнання. Окремо варто звертати увагу на дотримання правил пожежної безпеки, оскільки робоче місце розробника зазвичай передбачає використання комп'ютера, зарядних пристроїв, подовжувачів та іншої електротехніки.

Законодавство у сфері охорони праці покладає на роботодавця або відповідальну особу обов'язок забезпечувати працівникам безпечні й нешкідливі умови праці, підтримувати обладнання та інструменти у належному стані, організовувати інструктажі й перевіряти знання працівників з основних питань безпеки праці. Порядок створення та функціонування організаційної системи охорони праці визначається Типовим положенням про службу охорони праці [22]. Такі вимоги є актуальними і для ІТ-сфери, оскільки

правильно організоване робоче місце допомагає зберігати здоров’я працівників, зменшувати професійні ризики та підвищувати ефективність їхньої роботи. Важливе місце у законодавстві займає наказ про типове положення та порядок проведення навчання і перевірки знань, який містить базові вимоги до організації навчання, інструктажів та контролю знань працівників. Працівник, який виконує свої завдання , повинен знати й дотримуватися правил безпечної роботи із комп’ютерною технікою, а також розуміти цілком алгоритм дій у разі несправності обладнання, пожежі чи іншої небезпечної ситуації.

Звертаючи увагу на усі умови, у яких здійснюватиметься робота в приміщенні з використанням ноутбука або комп’ютера, важливо належним чином облаштувати робоче місце працівника. Така діяльність передбачає значне інтелектуальне навантаження, тривале перебування перед монітором, а також використання комп’ютерної миші та інших периферійних пристроїв. Тому необхідно враховувати вимоги до розміщення обладнання, режиму праці та відпочинку, рівня освітлення, мікроклімату в приміщенні, а також зручності робочого столу й крісла. Ігнорування цих вимог може призвести до підвищеного навантаження на очі, а в перспективі погіршення зору та дискомфорт під час роботи, головного болю, який виникає внаслідок довгої роботи за пристроєм, втоми, порушення постави, бо при незупинній праці, страждає спина у сидячому положенні, зниження працездатності й загального погіршення самопочуття працівника.

Окреме місце в системі охорони праці займають питання електробезпеки, пожежної безпеки та цивільного захисту. У сучасних умовах більшість робочих місць передбачає використання комп’ютерної техніки, зарядних пристроїв, мережевого обладнання, подовжувачів та інших електроприладів. Порушення правил їх користування й застосування може спричинити коротке замикання, пошкодження обладнання, ураження електричним струмом або виникнення пожежі. Саме тому важливо

користуватися тільки справною технікою, яка працює безвідмовно й без збоїв не перевантажувати електромережу, регулярно звертати увагу на стан кабелів, його цілість, стабільність і дотримуватися встановлених правил безпечної роботи.

Крім виробничих і технічних ризиків, у сучасних умовах слід враховувати й небезпеки загального характеру. До них належать аварійні ситуації, перебої з електропостачанням, пожежі, повітряні тривоги та інші надзвичайні події, які можуть впливати як на безпеку працівників, так і на безперервність робочого процесу. Загальні правові засади захисту населення, територій і майна у разі виникнення надзвичайних ситуацій визначаються Кодексом цивільного захисту України [23].

Сучасний підхід до охорони праці передбачає й акцентує увагу на формальне дотримання законодавчих вимог, а ще й на своєчасне виявлення потенційних небезпек, оцінювання різних ступенів ризиків та розроблення заходів, мір для їх зменшення. Це дає змогу завчасно попереджати виникнення небезпечних ситуацій, мінімізувати вплив шкідливих чинників на працівників і підтримувати належний рівень безпеки на робочому місці. Принципи системного управління професійними ризиками відображені у ДСТУ ISO 45001:2019, який установлює вимоги до систем управління охороною здоров’я та безпекою праці [24].

Отже, законодавче регулювання охорони праці охоплює комплекс правових, організаційних і профілактичних заходів, спрямованих на створення безпечних умов праці, збереження здоров’я працівників та запобігання професійним ризикам, які виникають внаслідок довгої розумової праці й можуть нести серйозні проблеми працівнику. Для діяльності, пов’язаної з використанням комп’ютерної техніки й програмних засобів, дані вимоги мають важливе практичне значення. Вони допомагають правильно організувати робоче місце, зменшити вплив шкідливих факторів, запобігти перевтомі та забезпечити стабільне виконання професійних завдань.

4.2 Характеристика об’єкта та виявлення потенційних небезпек

У межах цієї бакалаврської роботи розглядається створення системи автоматичного виявлення фейкових новин в Україні з використанням методів машинного навчання та елементів штучного інтелекту. Розробка такого програмного продукту передбачає написання й тестування коду мовою Python, аналіз і попередню обробку текстових даних, навчання класифікаційної моделі, а також роботу із серверною частиною, веб-інтерфейсом і REST API. Тому під час виконання роботи важливо пам’ятати й враховувати у першу чергу характеристики, що описують функціональні можливості та технічні параметри, а по-друге умови праці людини, яка займається проєктуванням системи, програмною реалізацією працюючого коду, тестуванням розробленого додатку (вебсайту тощо) та подальшим використанням й розвитком із метою інтеграції й впровадження оновлень .

Об’єктом аналізу в цьому підрозділі є робоче місце розробника програмної системи автоматичного виявлення фейкових новин. Оскільки всі основні етапи створення продукту виконуються за допомогою персонального комп’ютера або ноутбука, доцільно зосередити увагу саме на умовах праці користувача ПК у приміщенні.

Під час роботи над проєктом розробник виконує низку завдань, пов’язаних із програмуванням, обробкою текстових даних, навчанням і тестуванням моделі машинного навчання, запуском серверної частини на Flask, перевіркою REST API, роботою з базою даних SQLite та веб-інтерфейсом користувача. Така діяльність передбачає тривале перебування за комп’ютером, потребує високої концентрації уваги, створює значне зорове й розумове навантаження, а також вимагає постійного використання електронного обладнання.

Робоче місце розробника розташовується у приміщенні та містить такі предмети, як робочий стіл, крісло, персональний комп’ютер або ноутбук,

монітор, клавіатуру, мишу, зарядний пристрій, мережевий фільтр або подовжувач, а також засоби доступу до мережі Інтернет. Для безпечного виконання роботи важливими є перелік таких умов: правильна організація робочого місця, достатній рівень освітлення, справність електрообладнання, оптимальний мікроклімат у приміщенні та дотримання режиму праці й відпочинку [29].

Під час роботи за комп'ютером на працівника можуть впливати фізичні та психофізіологічні фактори. До фізичних факторів належать недостатнє або нерівномірне освітлення, шум від обладнання, несприятливий мікроклімат, використання електрообладнання та ризик виникнення пожежі [30]. До психофізіологічних факторів можна віднести напруження зору, статичне навантаження, монотонність праці, розумове й емоційне перенапруження [25].

Окремою особливістю роботи над системою виявлення фейкових новин є постійна взаємодія з великими обсягами текстової інформації. Частина таких матеріалів може охоплювати маніпулятивний або емоційно напружений зміст. Це підвищує вимоги до уважності розробника та може спричиняти додаткове психоемоційне навантаження під час аналізу даних і перевірки результатів роботи моделі.

Результати виявлення потенційних небезпек, характерних для робочого місця розробника системи автоматичного виявлення фейкових новин, наведено в таблиці 4.1.

Таблиця 4.1 – Виявлення потенційних небезпек стосовно об'єкта проєктування

Потенційна небезпека	Джерело небезпеки	Можливі наслідки
1	2	3
Перенапруження органів зору	Тривала робота з монітором, висока яскравість екрана, дрібний шрифт у середовищі розробки	Втома очей, головний біль, зниження концентрації уваги

Продовження таблиці 4.1

1	2	3
Недостатнє або нерівномірне освітлення	Неправильне розташування робочого місця, відблиски на екрані, відсутність локального освітлення	Зорове напруження, швидка втомлюваність, зниження продуктивності
Статичне навантаження на опорно-руховий апарат	Тривале сидіння, незручне крісло, неправильна висота столу, відсутність перерв	Біль у спині, шиї та плечах, порушення постави, фізична втома
Навантаження на кисті рук	Тривала робота з клавіатурою та мишею, повторювані рухи	Дискомфорт у кистях, перенапруження м’язів, зниження працездатності
Психоемоційне та розумове перенапруження	Аналіз великих обсягів текстових даних, тестування моделі, дедлайни, робота з фейковим контентом	Стрес, дратівливість, зниження уважності, професійне виснаження
Несприятливий мікроклімат і шум	Недостатня вентиляція, сухе повітря, перегрів техніки, шум від комп’ютерного обладнання	Сонливість, головний біль, зниження концентрації та працездатності
Електро- та пожежна небезпека	Використання комп’ютера, зарядних пристроїв, подовжувачів, перевантаження електромережі	Ураження електричним струмом, коротке замикання, пожежа, пошкодження обладнання
Надзвичайні ситуації та військова загроза	Повітряні тривоги, обстріли, аварійні відключення електроенергії, нестабільна робота інфраструктури	Загроза життю і здоров’ю, переривання роботи, втрата незбережених даних, психологічне напруження

Описані небезпеки мають різний характер і можуть формувати фізичний стан працівника, так і якість його роботи, рівень концентрації та психологічний стан. Найбільшу частину ризиків можна зменшити завдяки

правильній організації робочого місця, дотриманню графіку режиму праці й відпочинку, що складатиметься із розминки для очей, використанню справного й належно працюючого електрообладнання та його регулярній, постійній перевірці, забезпечення й організації відповідного освітлення, мікроклімату в приміщенні, а також виконанню правил пожежної безпеки. Хоча окремі ризики неможливо повністю усунути, доцільно провести оцінювання найбільш суттєвих небезпек і розробити заходи для їх зниження.

4.3 Дослідження ризику реалізації потенційних небезпек на об’єкті проєктування та розробка заходів щодо їх попередження

Оцінювання ризиків є важливим етапом забезпечення безпеки праці, оскільки воно дає змогу визначити, які небезпеки можуть найбільше впливати на працівника та які заходи необхідно впровадити в першу чергу. Рівень ризику залежить від двох основних чинників: імовірності виникнення небезпечної події та тяжкості її можливих наслідків. Такий підхід дає можливість не лише виявити потенційні загрози, а й установити пріоритетність дій, спрямованих на їх попередження або зменшення.

У цьому підрозділі проведено оцінювання найбільш суттєвих небезпек, які можуть виникати на робочому місці розробника системи автоматичного виявлення фейкових новин. Для аналізу обрано п’ять факторів, найбільш характерних для діяльності, що передбачає тривалу роботу за комп’ютером: перенапруження органів зору, статичне навантаження на опорно-руховий апарат, психоемоційне та розумове перенапруження, електрична й пожежна небезпека, а також надзвичайні ситуації та військова загроза.

Оцінювання виконано за допомогою матриці ризику, наведеної у методичних вказівках. У таблиці 4.2 подано категорію серйозності наслідків, рівень імовірності виникнення небезпеки, індекс ризику та загальну характеристику його рівня.

Таблиця 4.2 – Оцінювання ризиків потенційних небезпек на робочому місці розробника

Небезпека	Категорія серйозності наслідків	Рівень імовірності	Індекс ризику	Характеристика ризику
Перенапруження органів зору	III – гранична: втома очей, головний біль, зниження концентрації	B – ймовірна подія, оскільки робота з монітором є постійною	IIIB	Небажаний ризик, потребує профілактичних заходів
Статичне навантаження на опорно-руховий апарат	III – гранична: біль у спині, шиї, плечах, порушення постави	B – ймовірна подія при тривалій роботі сидячи	IIIB	Небажаний ризик, потребує корекції організації робочого місця
Психоемоційне та розумове перенапруження	II – критична: стрес, виснаження, зниження працездатності	C – можлива подія, особливо під час дедлайнів і тестування	IIC	Небажаний ризик, потребує організаційних заходів
Електро- та пожежна небезпека	II – критична: ураження електричним струмом, коротке замикання, пожежа	D – малоймовірна, але можлива подія за умови несправності обладнання	IID	Допустимий за умови постійного контролю
Надзвичайні ситуації та військова загроза	I – катастрофічна: загроза життю та здоров’ю, пошкодження майна	D – малоймовірна, але можлива подія	ID	Небажаний ризик, потребує плану дій у разі небезпеки

За результатами проведеного оцінювання встановлено, що для щоденної роботи розробника найбільш характерними є ризики, пов’язані із зоровим напруженням, поганою поставою, тривалим перебуванням у статичному

положенні та психоемоційним перенапруженням й недостатнім неякісним відпочинком. Саме ці небезпеки мають вищу ймовірність виникнення, адже мають прямий зв’язок із основним характером виконуваної роботи й загальним видом діяльності — тривалою роботою за персональним комп’ютером, аналізом інформації, написанням коду та тестуванням програмного продукту.

Електрична та пожежна небезпека виникають рідше, однак їхні наслідки можуть бути значно серйознішими, тому вони також потребують постійної профілактики. Це передбачає перевірку справності обладнання, правильне використання зарядних пристроїв, подовжувачів і мережевих фільтрів. Надзвичайні ситуації та військова загроза мають окремий характер, проте в сучасних умовах України їх також необхідно враховувати під час організації роботи в будь-якому приміщенні.

Для зниження рівня виявлених ризиків доцільно впровадити комплекс організаційних, технічних і профілактичних заходів. Вони мають бути спрямовані на правильну організацію робочого місця, дотримання режиму праці та відпочинку, безпечне використання електрообладнання, підтримання комфортних умов у приміщенні та готовність до дій у разі надзвичайної ситуації. Основні рекомендації щодо зменшення ризиків наведено в таблиці 4.3.

Таблиця 4.3 – Заходи щодо зниження ризиків

Небезпека	Запропонований захід	Очікуваний результат
1	2	3
Перенапруження органів зору	Дотримуватися раціонального режиму праці та відпочинку, робити короткі перерви, налаштовувати яскравість і контрастність екрана, розміщувати монітор на безпечній відстані	Зменшення втоми очей, головного болю та підвищення концентрації уваги
Недостатнє або нерівномірне освітлення	Забезпечити достатнє природне та штучне освітлення, уникати відблисків на екрані, використовувати локальне освітлення за потреби	Зниження зорового навантаження та покращення комфортності роботи

Продовження таблиці 4.3

1	2	3
Статичне навантаження на опорно-руховий апарат	Використовувати зручне крісло, підтримувати правильну позу, розташовувати стіл і монітор відповідно до ергономічних вимог, виконувати розминку під час перерв	Зменшення болю у спині, шиї та плечах, профілактика порушень постави
Навантаження на кисті рук	Правильно розташовувати клавіатуру та мишу, уникати надмірного напруження кистей, робити короткі вправи для рук	Зниження м’язового перенапруження та дискомфорту під час роботи
Психоемоційне та розумове перенапруження	Планувати робочий час, чергувати складні та прості завдання, обмежувати безперервну роботу з тривожним або маніпулятивним контентом, робити перерви	Зменшення стресу, підвищення уважності та стабільності працездатності
Несприятливий мікроклімат і шум	Провітрювати приміщення, підтримувати комфортну температуру, уникати перегріву техніки, мінімізувати сторонні шуми	Поліпшення самопочуття, зниження втомлюваності та дратівливості
Електро- та пожежна небезпека	Використовувати справне обладнання, не перевантажувати подовжувачі, перевіряти стан кабелів, не залишати техніку без нагляду, дотримуватися правил пожежної безпеки	Зниження ризику короткого замикання, пожежі та пошкодження обладнання
Надзвичайні ситуації та військова загроза	Дотримуватися сигналів повітряної тривоги, мати визначений маршрут до укриття, регулярно зберігати робочі файли, використовувати резервне копіювання	Підвищення безпеки працівника, зменшення ризику втрати даних і переривання роботи

Розглянуті заходи дають змогу зменшити вплив небезпечних і шкідливих факторів, характерних для роботи розробника. До найважливіших із них належать: правильна організація робочого місця, дотримання режиму й графіку праці та правильним відпочинком із умовою виконання розминки для очей, яка покращує зір, контроль справності електрообладнання, забезпечення належного освітлення, якості повітря й працюючої вентиляції, а також готовність до дій у разі виникнення надзвичайної ситуації чи іншої небезпеки.

Отже, проведене оцінювання ризиків показало, що робоче місце розробника не належить до об’єктів із високим рівнем виробничої небезпеки, проте потребує системного підходу до організації безпечних умов праці. За умови виконання запропонованих заходів ризики для здоров’я працівника знижуються до прийнятного рівня, що забезпечує можливість безпечного й ефективного виконання завдань із розробки, тестування та експлуатації програмного продукту.

Висновки до четвертого розділу

У цьому розділі розглянуто питання охорони праці під час розробки системи автоматичного виявлення фейкових новин. Основну увагу приділено умовам праці розробника, оскільки створення програмного продукту передбачає тривалу роботу за персональним комп’ютером, аналіз даних, написання коду, тестування системи та використання програмних інструментів.

У процесі аналізу визначено основні потенційні небезпеки, характерні для такого робочого місця: зорове напруження, статичне навантаження, психоемоційна втома, недостатнє освітлення, несприятливий мікроклімат, електрична та пожежна небезпека, а також ризики, пов’язані з надзвичайними ситуаціями. За результатами оцінювання встановлено, що найбільш імовірними є ризики, спричинені тривалою роботою за комп’ютером, тому саме вони потребують першочергової профілактики.

Для зменшення впливу небезпечних і шкідливих факторів запропоновано комплекс заходів: правильну організацію робочого місця, дотримання режиму праці та відпочинку, забезпечення належного освітлення й мікроклімату, використання справного електрообладнання, регулярне збереження даних і дотримання правил безпеки в разі виникнення надзвичайних ситуацій. Реалізація цих заходів дасть змогу підвищити рівень безпеки праці, зберегти працездатність розробника та створити стабільні умови для ефективної роботи над програмною системою.

ВИСНОВКИ

У кваліфікаційній роботі виконано проєктування та програмну реалізацію системи автоматичного виявлення фейкових новин на основі методів машинного навчання та обробки природної мови мовою програмування Python. За результатами виконання роботи можна зробити такі висновки.

1. Проведений аналіз предметної області підтвердив актуальність і соціальну значущість проблеми автоматичного виявлення фейкових новин. Встановлено, що дезінформація є складним багатовимірним явищем, яке охоплює різні жанри та механізми поширення. Жодна з наявних систем не поєднує одночасно повну автоматизацію, підтримку кількох мов та відкритий вихідний код.

2. Аналіз існуючих підходів показав, що методи на основі контентного аналізу тексту є найбільш практичними для реалізації системи реального часу. TF-IDF у поєднанні з логістичною регресією забезпечує відмінний баланс між якістю класифікації та вимогами до обчислювальних ресурсів.

3. Спроектовано чотирирівневу клієнт-серверну архітектуру системи з чітким поділом відповідальності між компонентами. Використання принципів низького зв'язування та високої зв'язності забезпечує легке масштабування та можливість незалежного оновлення компонентів.

4. Реалізовано повний NLP-пайплайн попередньої обробки тексту, що включає п'ять послідовних кроків: очищення тексту, токенізацію, видалення стоп-слів, лематизацію та TF-IDF векторизацію. Пайплайн застосовується однаково під час навчання та інференсу, що запобігає витоку даних та забезпечує коректне порівняння результатів.

5. Навчений класифікатор на базі TF-IDF + Logistic Regression досяг точності 92,8 % та значення ROC-AUC = 0,982 на тестовій вибірці реального

ISOT Dataset. При функціональному тестуванні на зовнішніх даних досягнуто точності 93 %.

6. Розроблено серверну частину на Flask з чотирма REST API ендпоінтами (/api/predict, /api/history, /api/stats, /api/health) та адаптивний веб-інтерфейс з кольоровою індикацією результатів. Середній час відповіді на запит класифікації складає 87 мс.

7. Проведено трирівневе тестування системи: 15 автоматичних тестів (pytest), інтеграційне тестування через pytest-flask та навантажувальне тестування (Locust). Покриття коду тестами - 83 %. У рамках функціонального тестування на 200 реальних прикладах досягнуто точності 93 %.

Перспективи подальшого розвитку системи: дообнавчання DistilBERT для підвищення якості на 3-5 %; розширення підтримки української мови (навчання на україномовному корпусі фейків); реалізація браузерного розширення для перевірки статей безпосередньо під час читання; інтеграція із зовнішніми базами даних факт-чекінгу (PolitiFact API, ClaimBuster); застосування методів пояснюваного AI (LIME, SHAP) для пояснення причин класифікації конкретного тексту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wardle C., Derakhshan H. Information Disorder: Toward an interdisciplinary framework for research and policymaking. Council of Europe Report. 2017. 107 p.

2. Vosoughi S., Roy D., Aral S. The spread of true and false news online. Science. 2018. Vol. 359, Issue 6380. P. 1146-1151.

3. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of NAACL-HLT 2019. P. 4171-4186.

4. Wang W. Y. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. Proceedings of ACL 2017. P. 422-426.

5. Shu K., Sliva A., Wang S., Tang J., Liu H. Fake News Detection on Social Media: A Data Mining Perspective. ACM SIGKDD Explorations Newsletter. 2017. Vol. 19, № 1. P. 22-36.

6. Ahmed H., Traore I., Saad S. Detecting opinion spams and fake news using text classification. Security and Privacy. 2018. Vol. 1, Issue 1. Article e9.

7. Ruchansky N., Seo S., Liu Y. CSI: A Hybrid Deep Model for Fake News Detection. Proceedings of CIKM 2017. P. 797-806.

8. Pérez-Rosas V., Kleinberg B., Lefevre A., Mihalcea R. Automatic Detection of Fake News. Proceedings of COLING 2018. P. 3391-3401.

9. Zhang X., Ghorbani A. An Overview of Online Fake News: Characterization, Detection, and Discussion. Information Processing & Management. 2020. Vol. 57, Issue 2. Article 102025.

10. Sanh V., Debut L., Chaumond J., Wolf T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108. 2019. 5 p.

11. Vaswani A. et al. Attention Is All You Need. Advances in Neural Information Processing Systems 30 (NIPS 2017). P. 5998-6008.

12. Pedregosa F. et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011. Vol. 12. P. 2825-2830.

13. Bird S., Klein E., Loper E. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media. 2009. 504 p.

14. Wolf T. et al. HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv:1910.03771. 2020. 12 p.

15. Grinberg N., Joseph K., Friedland L., Swire-Thompson B., Lazer D. Fake news on Twitter during the 2016 US presidential election. Science. 2019. Vol. 363. P. 374-378.

16. Shu K., Mahudeswaran D., Wang S., Lee D., Liu H. FakeNewsNet: A Data Repository with News Content, Social Context, and Spatiotemporal Information for Studying Fake News on Social Media. Big Data. 2020. Vol. 8, № 3. P. 171-188.

17. Zubiaga A., Aker A., Bontcheva K., Liakata M., Procter R. Detection and Resolution of Rumours in Social Media: A Survey. ACM Computing Surveys. 2018. Vol. 51, Issue 2. Article 32.

18. NLTK Project. Natural Language Toolkit (NLTK) Documentation. URL: https://www.nltk.org (дата звернення: 15.03.2026).

19. Pallets Projects. Flask Web Development. URL: https://flask.palletsprojects.com (дата звернення: 17.03.2026).

20. Ahmed H. ISOT Fake News Dataset. University of Victoria. URL: https://www.uvic.ca/engineering/ece/isot/datasets (дата звернення: 10.03.2026).

21. Про охорону праці : Закон України від 14.10.1992 № 2694-XII [Електронний ресурс] / Верховна Рада України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12#Text. (дата звернення: 25.05.2026).

22. Про затвердження Типового положення про службу охорони праці : Наказ Держнаглядохоронпраці України від 15.11.2004 № 255 [Електронний ресурс] / Верховна Рада України. – Режим доступу:

https://zakon.rada.gov.ua/laws/show/z1526-04#Text. (дата звернення: 30.05.2026).

23. Кодекс цивільного захисту України : Кодекс України від 02.10.2012 № 5403-VI [Електронний ресурс] / Верховна Рада України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/5403-17#Text. (дата звернення: 31.05.2026).

24. Системи управління охороною здоров'я та безпекою праці. Вимоги та настанови щодо застосування : ДСТУ ISO 45001:2019. – [Чинний від 2021-01-01]. – Київ : ДП «УкрНДНЦ», 2019. – 23 с. – Режим доступу: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=88004 (дата звернення: 31.05.2026).

25. Методичні вказівки до виконання розділу «Охорона праці» в дипломних роботах бакалаврів / Укл. В. В. Малишева. – Харків : ХНУМГ ім. О. М. Бекетова, 2025. – 14 с.

26. Sobchuk Y., Krushelnytskyi S., Lipianina-Honcharenko K., Ivasechko A., Drakokhrust T. Hybrid method for building a balanced Ukrainian-language news corpus for fake news detection. CEUR Workshop Proceedings. 2026. Vol. 4159. 11 p. URL: https://ceur-ws.org/Vol-4159/paper13.pdf (дата звернення: 04.06.2026).

27. Трещов М. М. Методичні рекомендації для проведення семінарських та практичних занять з навчальної дисципліни «Інформаційна гігієна та протидія дезінформації» для здобувачів першого (бакалаврського) рівня вищої освіти. Дніпро : ДДУВС, 2025. 28 с. URL: https://er.dduvs.edu.ua/bitstream/123456789/16239/1/макет_Методичка_інформаційна%20гігієна_ред.pdf (дата звернення: 04.06.2026).

28. Джоші А., Павленко В. І. Порівняння методів машинного навчання для визначення фейкових новин. Інфокомунікаційні та комп'ютерні технології. 2024. Том 1, № 07. С. 46–55. DOI: https://doi.org/10.36994/2788-5518-2024-01-07-06.

29. Природне і штучне освітлення : ДБН В.2.5-28:2018. – [Чинний від 2019-03-01]. – Київ : ДП «НДІ БК», 2018. – 94 с.

30. Державні санітарні норми і правила при роботі з джерелами електромагнітних полів : ДСанПіН 3.3.6.096-2002. – [Чинний від 2002-12-18]. – Київ : МОЗ, 2002. – 24 с.

ДОДАТОК А

Фрагменти коду модуля попередньої обробки тексту (preprocessing.py)

```
import re
import logging
logger = logging.getLogger(__name__)

_BUILTIN_STOPWORDS = {
    "a","an","the","and","or","but","in","on","at","to",
    "for","of","with","by","from","is","are","was","were",
    "be","been","have","has","had","do","does","did",
}

class TextPreprocessor:
    def __init__(self):
        self.stop_words = _BUILTIN_STOPWORDS

    def clean_text(self, text: str) -> str:
        text = re.sub(r"<[^>]+>",      " ", text)  # HTML
        text = re.sub(r"https?://\S+", " ", text)  # URL
        text = re.sub(r"@\w+",         " ", text)  # mentions
        text = re.sub(r"[^a-zA-Z\s]", " ", text)  # non-alpha
        return re.sub(r"\s+", " ", text).strip().lower()

    def preprocess(self, text: str) -> str:
        cleaned = self.clean_text(text)
        tokens  = cleaned.split()
        tokens  = [t for t in tokens
                   if t not in self.stop_words and len(t) > 2]
        return " ".join(tokens)
```

ДОДАТОК Б

Фрагменти коду навчання та збереження моделі (model.py, train.py)

```
# model.py
import joblib
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

class FakeNewsClassifier:
    def __init__(self):
        self.pipeline = Pipeline([
            ("tfidf", TfidfVectorizer(
                max_features=50000, ngram_range=(1, 2),
                min_df=2, sublinear_tf=True,
            )),
            ("clf", LogisticRegression(
                C=1.0, max_iter=1000, random_state=42,
            ))
        ])

    def train(self, X, y):    self.pipeline.fit(X, y)
    def predict_proba(self, X): return self.pipeline.predict_proba(X)[:,1]
    def save(self, path):    joblib.dump(self.pipeline, path)
    @classmethod
    def load(cls, path):
        o = cls(); o.pipeline = joblib.load(path); return o
```

ДОДАТОК В

Фрагменти коду серверної частини (app.py)

```
from flask import Flask, request, jsonify, render_template
from src.model import get_classifier
from src.preprocessing import TextPreprocessor
from src.database import init_db, log_prediction, get_history, get_stats

app          = Flask(__name__)
preprocessor = TextPreprocessor()
init_db()
classifier   = get_classifier()

@app.route("/")
def index(): return render_template("index.html")

@app.route("/api/predict", methods=["POST"])
def predict():
    data = request.get_json(silent=True)
    if not data or "text" not in data:
        return jsonify({"error": "Field text required"}), 400
    text = str(data["text"]).strip()
    if not text: return jsonify({"error": "Empty text"}), 422
    processed   = preprocessor.preprocess(text)
    probability = float(classifier.predict_proba([processed])[0])
    label       = "FAKE" if probability >= 0.5 else "REAL"
    confidence  = "висока" if abs(probability-.5)>.35 else "середня"
    log_prediction(text, label, round(probability,4), confidence)
    return jsonify({"label":label,"probability":round(probability,4),
                    "confidence":confidence})
if __name__ == "__main__": app.run(host="0.0.0.0", port=5000)
```