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

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.912

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-науковою програмою
«Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»
зі спеціальності 121 Інженерія програмного забезпечення
на тему: «Програмний метод аналізу текстів новин на неправдивість»

Виконав:
студент ІІ курсу, групи КП-21мн
Мельничук Олексій Геннадійович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н.,
Олещенко Любов Михайлівна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри ІСТ, к.т.н., доцент,
Полторак Вадим Петрович

_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність (спеціалізація) – 121 «Інженерія програмного забезпечення»

Освітньо-наукова програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Євгенія СУЛЕМА

«___»_____2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Мельничуку Олексію Геннадійовичу

1. Тема дисертації «Програмний метод аналізу текстів новин на неправдивість», науковий керівник дисертації доцент кафедри ПЗКС, к.т.н., доцент Олещенко Любов Михайлівна, затверджені наказом по університету від 27 березня 2024 р. №1478-С.

2. Термін подання студентом дисертації «13» червня 2024 р.

3. Об'єкт дослідження: процес аналізу текстів новинних статей на предмет невідповідності встановленим фактам.

4. Предмет дослідження: методи програмної інженерії для розроблення програмних систем аналізу текстів новин на неправдивість з використанням алгоритмів машинного навчання.

5. Перелік завдань, які потрібно розробити:
- провести дослідження наукових матеріалів та алгоритмів для поставленої задачі;
- визначити набір алгоритмів та методів, які будуть використовуватись для розробки;
- розробити алгоритмічний метод аналізу тексту;
- порівняти результати алгоритму з відомими аналогами;
- створити сервіс для тестування алгоритму;
- спростувати чи підтвердити гіпотези

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
- схема алгоритму попередньої обробки текстових даних;
- діаграма вебсервісу;
- схема роботи ансамблевих класифікаторів;
- схема алгоритму циклічного тренування класифікаторів;

7. Орієнтовний перелік публікацій:
- Тези доповіді «Програмний метод аналізу текстів новин на неправдивість»;
- Стаття «Застосування ансамблевих методів машинного навчання для виявлення неправдивого тексту».

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент кафедри ПЗКС		

9. Дата видачі завдання «11» жовтня 2022 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ознайомлення з предметною галуззю	23.12.2022	
2.	Розробка та узгодження технічного завдання; визначення структури магістерської дисертації; вивчення та пошук літератури	15.03.2023	
3.	Проведення наукового дослідження; підготовка матеріалів першого розділу	10.05.2023	
4.	Розробка основного модулю проєкту, налаштування алгоритмів; тестування алгоритмів	29.05.2023	
5.	Підготовка матеріалів другого розділу; підготовка доповіді на ПМК-2023	28.10.2023	
6.	Рефакторинг та оптимізація коду; розробка вебсервісу	23.12.2023	
7.	Підготовка матеріалів третього розділу	13.02.2024	
8.	Підготовка матеріалів четвертого розділу	17.03.2024	
9.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу; підготовка наукової статті	01.04.2024	
10.	Оформлення текстової і графічної частини магістерської дисертації	08.05.2024	

Студент _____ Олексій МЕЛЬНИЧУК

Науковий керівник _____ Любов ОЛЕЩЕНКО

РЕФЕРАТ

Актуальність теми. З розвитком інформаційних технологій цифровізація новин була лише питанням часу. Завдяки доступності Інтернету та соціальних мереж, інформація почала поширюватися зі швидкістю, до якої людство не було готовим. З цією швидкістю стало складно дотримуватись нейтралітету та правдивості новинних текстів, особливо в соціальних мережах.

Об'єктом дослідження є процес аналізу текстів новинних статей на предмет невідповідності встановленим фактам.

Предметом дослідження є методи програмної інженерії для розроблення програмних систем аналізу текстів новин на неправдивість з використанням алгоритмів машинного навчання.

Мета роботи: підвищення точності виявлення достовірності аналізу текстів новин для спрощення ідентифікування неправдивої інформації.

Методи дослідження: в роботі використовуються програмні методи попередньої обробки та перетворення великих текстових даних, програмні методи навчання та тренування моделей класифікації, ансамблеві методи.

Наукова новизна: розроблено удосконалений програмний метод аналізу текстів новинних статей, характерними рисами якого є використання алгоритму попередньої комбінованої обробки та аналізу текстів новин різних категорій ансамблевими методами класифікації, що дає змогу підвищити точність розпізнавання у середньому до 4.18% порівняно з класичними методами.

Практична цінність роботи полягає у програмній реалізації запропонованого вдосконаленого методу обробки та аналізу текстових даних на неправдивість у вигляді вебзастосунку, що є конкурентоспроможним та готовим до розгортання у вигляді комерційного продукту.

Апробація роботи. Основні положення і результати даної роботи були представлені на науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг ПМК-2023» (Київ, 28 – 30 листопада 2023 р.) та у статті «Застосування ансамблевих методів машинного навчання для виявлення неправдивого тексту» в науковому журналі категорії Б «Вісник Херсонського національного технічного університету» (2024, № 1(88). С.258-263.).

Структура та обсяг роботи. Магістерська дисертація складається зі вступу, чотирьох розділів, висновків та додатків.

У вступі виконано оцінку стану проблеми теми роботи, обґрунтовано актуальність дослідження, сформульовано мету, ідеї та задачі досліджень.

У першому розділі був проведений аналіз загальних методик аналізу текстів, зроблено огляд існуючих комерційних рішень, їх особливості та характеристики.

У другому розділі описані алгоритмічні методи аналізу текстів, використані класифікатори, векторизатори, комбіновані ансамблеві алгоритми, наведені аргументи для їх використання в дослідженні.

У третьому розділі наведені вимоги для програмного забезпечення, обрано технології програмування, які забезпечують обробку великого обсягу даних. Зроблено огляд використаних інструментів, їх переваг та недоліків.

У четвертому розділі представлені результати роботи запропонованих алгоритмів обробки та аналізу текстових даних, зроблено порівняння всіх комбінацій алгоритмів з векторизаторами та категоріями недостовірності.

У висновках проаналізовані основні результати роботи.

Робота виконана на 83 аркушах, містить 3 додатки та посилання на список використаних літературних джерел з 74 найменувань. У роботі наведено 37 рисунків, 1 таблиця, 7 лістингів коду.

Ключові слова: програмний метод, ансамблеві методи, алгоритми класифікації, аналіз текстів на недостовірність, TF-IDF, Python.

ABSTRACT

Relevance of the topic. With the development of information technologies, the digitization of news was only a matter of time. Thanks to the availability of the Internet and social media, information began to spread at a speed to which humanity was not prepared. With this speed, it became difficult to maintain neutrality and truthfulness of news texts, especially on social media.

The object of the research is the process of analyzing news articles for inconsistencies with established facts.

The subject of the research are software engineering methods for the development of software systems for analyzing news texts for falsity using machine learning algorithms.

The aim of the work is development of an ensemble classification method for analyzing news texts for falsehood, satire and hate speech and creating a software application that will increase the accuracy of detecting the authenticity of news texts to simplify the identification of false information.

Research methods: the work uses software methods of preprocessing and transformation of large text data, software methods of learning and training classification models, ensemble methods.

The scientific novelty: an improved software method for analyzing the texts of news articles has been developed, characterized by the use of an algorithm for preliminary combined processing and analysis of news texts of different categories by ensemble classification methods, which makes it possible to increase the recognition accuracy by an average of 4.18% compared to classical methods.

The practical value of the work lies in the software implementation of the proposed improved method of processing and analyzing textual data for falsity in the form of a web application that is competitive and ready for deployment as a commercial product.

Validation of the work. The main provisions and results of this work were presented at the scientific conference of master's and graduate students "Applied Mathematics and Computing - 2023" (Kyiv, November 28-30, 2023) and the article "Machine learning ensemble methods implementation for deceptive text detection" in the category B scientific journal "Visnyk of Kherson National Technical University" (2024, No. 1(88). P.258-263.).

Structure and scope of the work. The master's thesis consists of an introduction, four chapters, conclusions, and appendices.

In the introduction, an assessment of the state of the problem of the research topic is made, the relevance of the research is substantiated, and the purpose, ideas, and tasks of the research are formulated.

The first chapter analyzes general methods of text analysis, provides an overview of existing commercial solutions, their features, and characteristics.

The second chapter describes algorithmic machine text analysis methods, classifiers used, vectorizers, combined ensemble algorithms, and arguments for their use in the research.

The third chapter presents software requirements, selected technologies for application that best meet the project's needs and ensure efficiency and speed of processing large volumes of data. An overview of the tools used, their qualities, and shortcomings is provided.

The fourth chapter presents the results of the algorithms' work, compares all combinations of algorithms with vectorizers and metrics.

The conclusions analyze the main results of the work.

The work consists of 83 pages, including 3 appendices and references to a list of 74 literary sources. It contains 37 figures, 1 table, and 7 listings.

Keywords: software method, ensemble machine learning methods, classification algorithms, falsehood text analysis, TF-IDF, Python.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення, програмний застосунок;

Датасет (dataset) – набір даних, в контексті машинного навчання – класифікована таблиця даних, яка використовується для тренування предиктивних моделей;

Документ – одиниця тексту, яка підлягає аналізу. В контексті наукової роботи це новина стаття;

Токен – результат розбиття речення, одиниця, з якою можуть працювати моделі аналізу мовлення;

AI (Artificial Intelligence) – ШІ/штучний інтелект;

NLP (Natural Language Processing) – обробка природної мови (тексту);

DL (Deep Learning) – глибоке навчання;

NB – класифікатор Naive Bayes;

PA – пасивно-агресивний класифікатор;

SVM – лінійний класифікатор опорних векторів;

LR – класифікатор логістичної регресії;

KNN – класифікатор k-найближчих сусідів;

FOREST – класифікатор випадкового лісу;

Back End – серверна частина, яка виконує бізнес логіку та є пластом доступу до даних;

Front End – клієнтська частина, абстракція над пластом Back End, яка надає доступ до розташованого за ним функціоналу через інтерфейс, зручний для користувача;

Краудсорсинг (crowdsourcing) – процес отримання послуг, ідей або фінансування від маси людей, зазвичай через Інтернет;

BOW (bag of words) – простий метод векторизації тексту, де кожне унікальне слово в документі представляється як одиниця і обчислюється його відносна частота в цьому документі.

ВСТУП

Неправдиві новини стали поширеною проблемою цифрової епохи, поширюючись через платформи соціальних медіа та онлайн-видання новин. Швидке поширення недостовірної або оманливої інформації становить серйозну загрозу суспільному дискурсу, політичним процесам і довірі суспільства. Традиційні методи перевірки текстів новин часто не відповідають масштабам і складності сучасних кампаній недостовірної інформації. Отже, існує нагальна потреба в передових обчислювальних техніках для боротьби з поширенням недостовірних новин та збереження цілісності інформаційних каналів.

Широка доступність Інтернету та платформ соціальних мереж прискорила поширення інформації безпрецедентними темпами. Таке швидке поширення створило проблеми для підтримки неупередженості та точності новинного контенту, особливо в середовищі соціальних мереж. Різні онлайн-платформи використовують різноманітні стратегії для класифікації та оцінки схожого вмісту. У той час як традиційні інформаційні агентства покладаються на процеси перевірки вручну, платформи соціальних мереж, такі як Facebook, використовують автоматизовані системи для виявлення ненависті та ворожнечі в коментарях. Такі платформи, як X, використовують «нотатки спільноти», спільні анотації, які додаються до публікацій, пропонуючи додатковий контекст та іноді розвінчують неправдиву інформацію. Аналізуючи величезну кількість текстових даних, моделі машинного навчання можуть робити прогнози та намагатися розпізнати правдивість нової інформації.

Метою даної магістерської дисертації є розробка ансамблевого методу класифікації для аналізу новинних текстів на неправду, сатиру та мову ворожнечі. Основна ідея – проаналізувати новинний текст без сторонньої інформації (дата публікації). Текст аналізується окремо за трьома показниками: правдивість новини, сатира чи мова ворожнечі.

1. АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Аналіз програмних методів перевірки текстів новин

Аналіз новин на недостовірність частіше за все проводиться вручну. Новини аналізуються візуально людьми, порівнюючи різні посилання, стиль тексту, шукаючи різні веббібліотеки, соціальні мережі тощо. З набутого досвіду було класифіковано та виведено кілька груп підходів до такого аналізу. Ці підходи можна поєднувати та застосовувати під час автоматизації процесу.

Мовні методи [1] спрямовані на використання лінгвістичних методів для розкриття недостовірних інформаційних матеріалів. Авторів недостовірної інформації можна розпізнати за властивим стилем мовлення. Ці методи враховують всі аспекти мови, включаючи граматичну структуру, послідовність слів та семантику. Особлива увага приділяється аналізу слів та їх значень. "Мішок слів" та семантичний аналіз – це лише декілька прикладів таких мовних методів.

У методі "мішка слів" [2] кожне слово у тексті розглядається окремо, незалежно одне від одного. Для цього використовується частота вживання кожного слова, а також n-грами, тобто послідовностей слів. Разом ці метрики можуть допомогти виявити шаблони недостовірної інформації. Недоліком цього підходу є ігнорування контексту і порядку слів.

Семантичний аналіз [3] передбачає визначення правдивості шляхом порівняння особистого досвіду з профілем на цю тему, отриманим із схожих джерел. Автори роблять схожі висновки щодо обговорюваної теми.

В методі глибокого синтаксису використовуються ймовірнісні контекстно-вільні граматики [4] для реалізації аналізу текстів. Ці граматики виконують глибокий синтаксичний аналіз через дерева розбору, що дозволяє аналізувати структури, що не обмежені контекстом. Ймовірнісна безконтекстна граматика є розширенням безконтекстної граматики, де речення перетворюються за допомогою правил, що дозволяють аналізувати

різні синтаксичні структури. Цей метод дозволяє порівнювати синтаксис з відомими шаблонами недостовірної інформації та визначати різницю між недостовірними та справжніми новинами.

Тематично-агностичні підходи [5] до виявлення недостовірних новин концентруються на особливостях самої теми, а не на змісті статей. Цей підхід використовує лінгвістичні ознаки та характеристики вебресурсів для виявлення недостовірних інформаційних матеріалів. Деякі з таких ознак, що не залежать від конкретної теми, включають велику кількість рекламних блоків, довгі заголовки та емоційне наповнення текстів.

Алгоритми машинного навчання можуть бути застосовані для виявлення недостовірних новин шляхом аналізу різноманітних навчальних даних. Цей процес полягає в використанні різних наборів даних для вдосконалення алгоритмів, що дозволяє розробникам створювати нові підходи та методи машинного навчання. Набори даних грають ключову роль у тренуванні алгоритмів для виявлення недостовірної інформації у новинах.

Підхід, що базується на знаннях, ставить за мету використовувати зовнішні джерела для перевірки достовірності новин та ідентифікації їх, ще до того, як вони розповсюдяться надто швидко [6]. Підхід, що базується на знаннях, також ще називають перевіркою фактів.

Існують три різні методи перевірки фактів:

- експертний;
- машинний;
- публічний.

Експертна перевірка [7] фактів вимагає докладного аналізу та вивчення даних і документів. Цей метод передбачає, що фахівці ретельно перевіряють точність новин шляхом проведення досліджень та оцінки конкретних заяв. Суть перевірки фактів полягає в тому, щоб надати достовірність певному елементу, порівнюючи точність тексту з іншим, попередньо перевіреним експертами рукописом.

Мета машинної перевірки фактів – забезпечити користувачам автоматизований процес визначення достовірності конкретної новини [8]. Такий процес використовує різноманітні методи, зокрема, графи знань і відкриті вебджерела, що базуються на практичних даних, для відокремлення правдивих інформаційних матеріалів від недостовірних. Наприклад, інструмент ClaimBuster є яскравим прикладом того, як перевірка фактів може автоматично виявляти недостовірні новини [9], він використовує методи машинного навчання та обробки природної мови, а також звертається до різних джерел даних для аналізу контексту у соціальних мережах, інтерв'ю та публічних виступах у реальному часі, з метою визначення "фактів" та їх порівняння з підтвердженими даними.

Публічний метод, або «краудсорсинг», дає можливість групі людей приймати колективне рішення шляхом перевірки точності новин [10]. Точність новин повністю ґрунтується на мудрості натовпу. Kiskkit є прикладом платформи, яку можна використовувати для краудсорсингу, де платформа дозволяє групі людей оцінювати частини новинної статті. Після того, як один матеріал було оцінено, натовп переходить до наступного матеріалу для оцінки, доки не буде оцінено всю статтю новин і її точність не буде визначено мудрістю натовпу.

Недавні дослідження вказують на необхідність поєднання машинного навчання з попередніми підходами для виявлення недостовірних новин. У соціальних мережах новини часто поширюються з неконтрольованою швидкістю. Щоб протидіяти цьому, використовуються гібридні методи аналізу текстів.

Недавнє дослідження [11] виокремлює три основних елементи, що характеризують недостовірні новини: текст статті, реакція на цю статтю та джерело, що її мотивує. В результаті проведеного аналізу [12] виникла інноваційна гібридна модель, яка використовує комбінацію людського та машинного навчання для виявлення недостовірних новин у соціальних мережах. Дослідження показує, що люди мають лише 4% шансів вгадати,

чи є новина правдивою, і можуть ідентифікувати Недостовірні новини лише у 54% випадків. Однак гібридна модель, заснована на поєднанні людських та машинних ресурсів, значно підвищує цей показник. Для досягнення ефективності ця модель поєднує аналіз новин у соціальних мережах з методами машинного навчання та мережевого підходу, мета якої – оцінка ймовірності того, що конкретна новина є недостовірною.

Прикладом гібридної моделі є також розробка, відома як CSI (захоплення, оцінка, інтеграція) [11]. Вона базується на трьох основних етапах:

- етап захоплення включає процес вилучення репрезентацій статей за допомогою повторюваної нейронної мережі (RNN);
- етап оцінки та векторного представлення статей;
- етап інтеграції, де результати захоплення та оцінки інтегруються у вектор, який використовується для класифікації новин.

1.2. Аналіз існуючих комерційних рішень

Новинний портал та мобільний додаток The Factual оцінює вміст новин на основі «обсягу та якості його джерел», «експерту журналіста», «упевненого характеру використовуваної мови» та «історичної репутації зір» [12]. Вебсервіс оцінює вміст за шкалою від 0 до 100, щоб оцінити якість вмісту. Інструмент оцінює якість новин на основі різноманітності джерел, досвіду автора, мови, що використовується, тощо. Вебсервіс також визначає джерела вищої якості та джерела з іншого боку політичного спектру. The Factual надає інформаційний бюлетень, додаток, розширення Chrome і вебсайт для користувачів, які хочуть отримувати інформацію про достовірність конкретних історій. The Factual працює на основі алгоритму, який щодня оцінює достовірність понад 10 000 новин. Фактори, які враховуються, включають історію джерел сайту, послужний список автора та різноманітність джерел у статті новин.

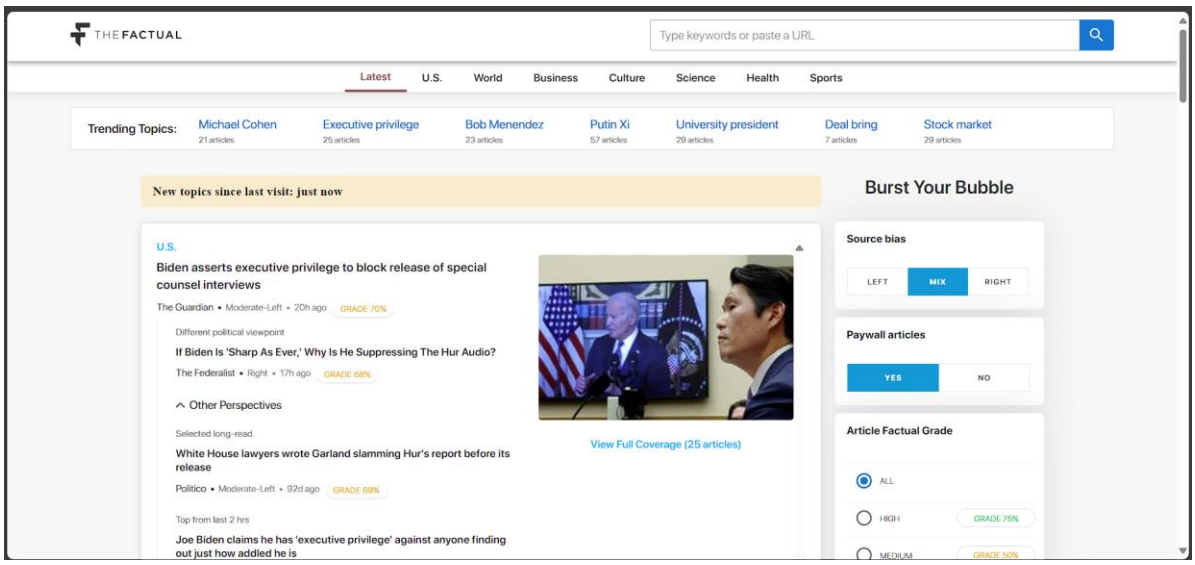


Рис. 1.1. Інтерфейс сервісу The Factual [12]

У 2019 році некомерційна організація Meedan запустила власний проєкт перевірки фактів [13] у поєднанні з WhatsApp і Facebook. Платформа Meedan Check увімкнула збір порад із відкритим кодом через WhatsApp щодо недостовірної інформації в Африці, Індії та Бразилії. Check включає такі функції, як прямий обмін повідомленнями між порадниками та перевіряючими факти. Алгоритми ідентифікують вміст на основі його схожості з позначеною недостовірною інформацією та можуть визначати пріоритетність певних типів інформації на основі вподобань користувача.

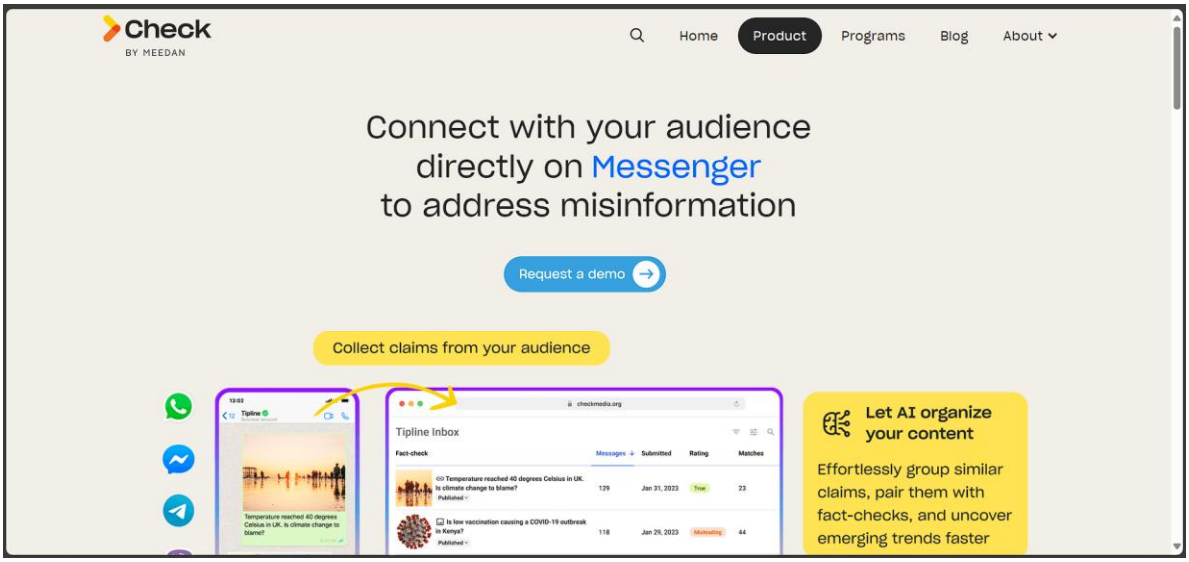


Рис. 1.2. Сторінка сервісу Meedan Check [13]

Logically, створений в 2017 році, є безкоштовним мобільним додатком і розширенням для браузера [14]. Сервіс надає послуги перевірки фактів і зображень. Logically використовує штучний інтелект як частину функції автоматичного помічника пошуку. Logically також покладається на експертів з перевірки фактів. Штучний інтелект сервісу розроблено для аналізу заяв, думок і подій. Він відстежує понад один мільйон вебдоменів і платформ соціальних медіа в режимі реального часу, використовуючи зібрану інформацію для оцінки достовірності інформації та тверджень в Інтернеті.

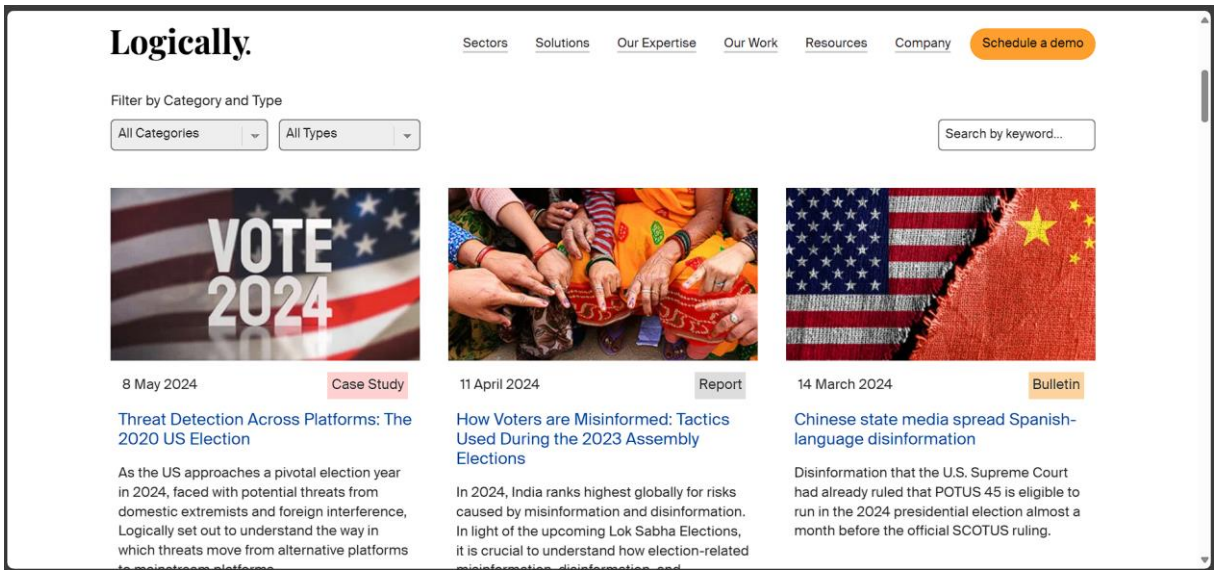


Рис. 1.3. Сторінка сервісу Logically [14]

Медіакомпанія та однойменний сервіс Full Fact, засновані в 2009 році, пропонують кілька інструментів перевірки фактів, у тому числі автоматизованих за допомогою штучного інтелекту [15]. Сервіс став переможцем конкурсу Google AI Impact Challenge 2019. Full Fact створює інструменти штучного інтелекту, щоб допомогти тим, хто перевіряє факти, зрозуміти, яка інформація дня є найважливішою та вартою перевірки. Сервіс також спрямований на розробку алгоритму, який може визначити, коли хтось свідомо повторює те, що, як їм відомо, є недостовірністю.

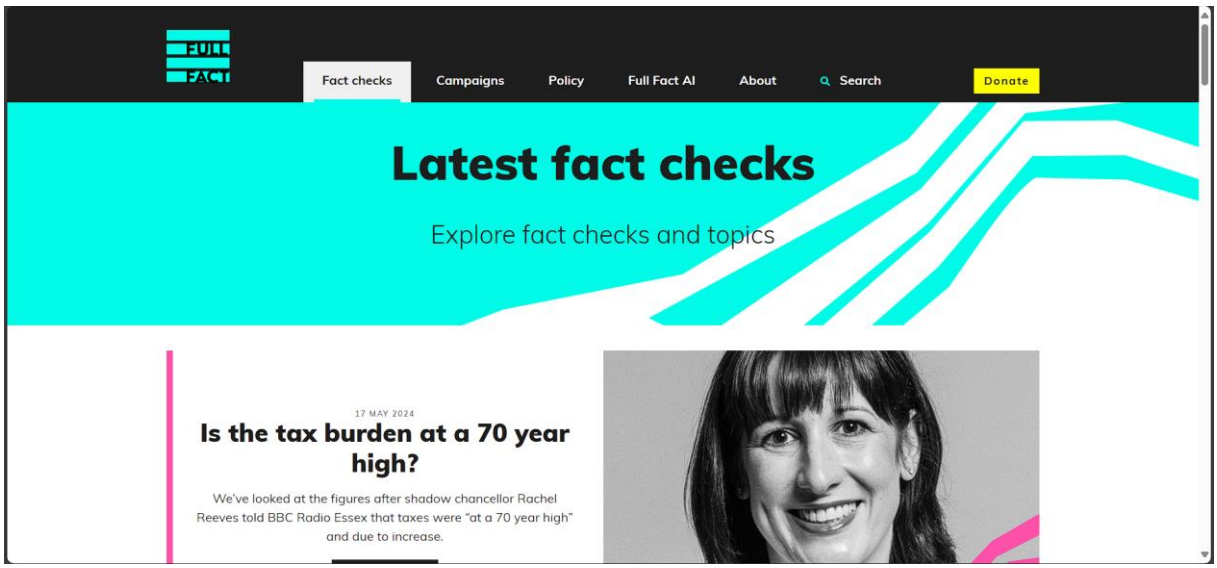


Рис. 1.4. Сторінка сервісу Full Fact [15]

У 2019 році компанія X Corp (Twitter Inc.) придбала стартап Fabula AI. Стартап [16] фокусувався на глибокому навчанні і його алгоритми зосереджені на унікальних моделях поширення недостовірної інформації в мережі Інтернет. Під час придбання Fabula компанія Twitter планувала посилити можливості глибокого навчання графіків. Імовірно, команда Fabula продовжуватиме робити ключові внески в поточні алгоритми Twitter, пов’язані з недостовірними новинами.

Також у 2019 дослідниками з Вашингтонського університету була представлена розробка Grover – модель штучного інтелекту для виявлення недостовірних новин [17]. Алгоритм використовує мову конкретних публікацій, щоб точніше виявляти недостовірну інформацію. Модель Grover довела здатність ефективно генерувати контент, що робить її ефективною у виявленні недостовірної інформації, створеної ШІ. Хоча дехто висловлював занепокоєння щодо потенціалу Grover створювати переконливі Недостовірні новини, основний задум полягає в тому, щоб використовувати сервіс для боротьби з недостовірною інформацією, а не сприяти її поширенню.

Рис. 1.5. Сторінка сервісу Grover [17]

Вище згаданий сервіс ClaimBuster є онлайн-інструментом для миттєвої перевірки фактів, заснований у 2017 році [9]. Використовуючи API Google Fact-Check Explorer, ClaimBuster дозволяє користувачам перевіряти правдивість власного тексту. Він збирає результати пошуку, подібні до їхньої письмової заяви, а також визначення відносної правдивості цих тверджень щодо хибності. ClaimBuster також стежить за політичними дебатами, покладаючись на штучний інтелект, щоб висвітлити заяви, які, на його думку, заслуговують на перевірку фактів.

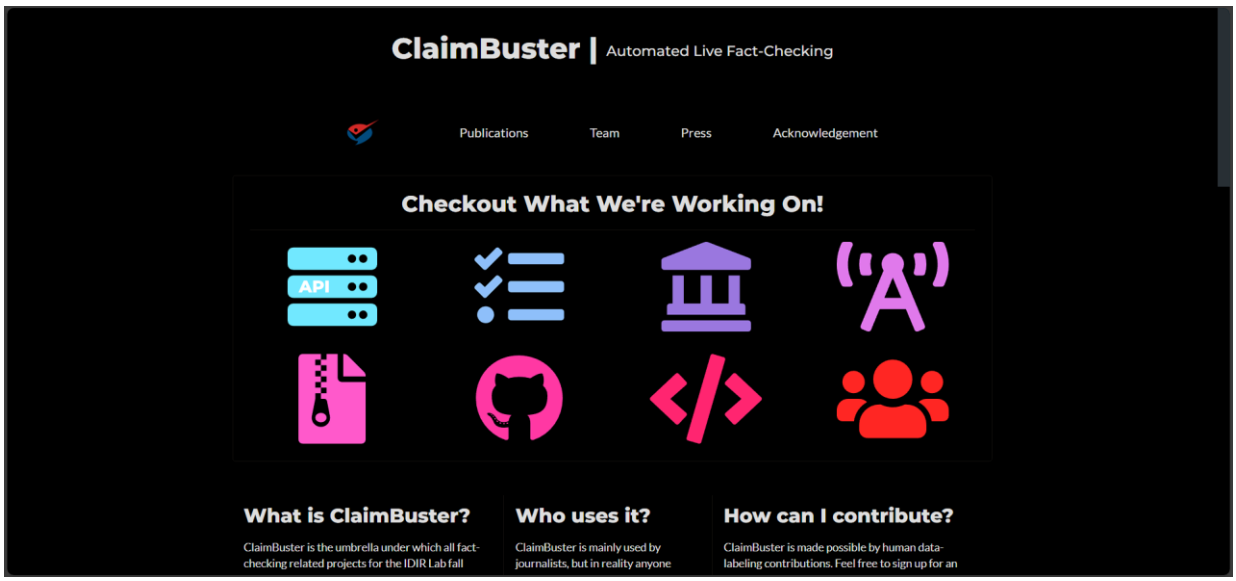


Рис. 1.6. Сторінка сервісу Claimbuster [9]

Схожою розробкою, але для виявлення фальшивої реклами, є Adverif.ai [18]. Фальшива реклама може бути троянським конем для зловмисного програмного забезпечення або може бути шкідливою з низки інших причин. Adverif.ai поєднує свій алгоритм FakeRank із рецензентами, щоб ідентифікувати та сповіщати користувачів про потенційно шкідливий вміст, будь то посилання зі зловмисним програмним забезпеченням або мініатюра з потенційно відвертим вмістом.

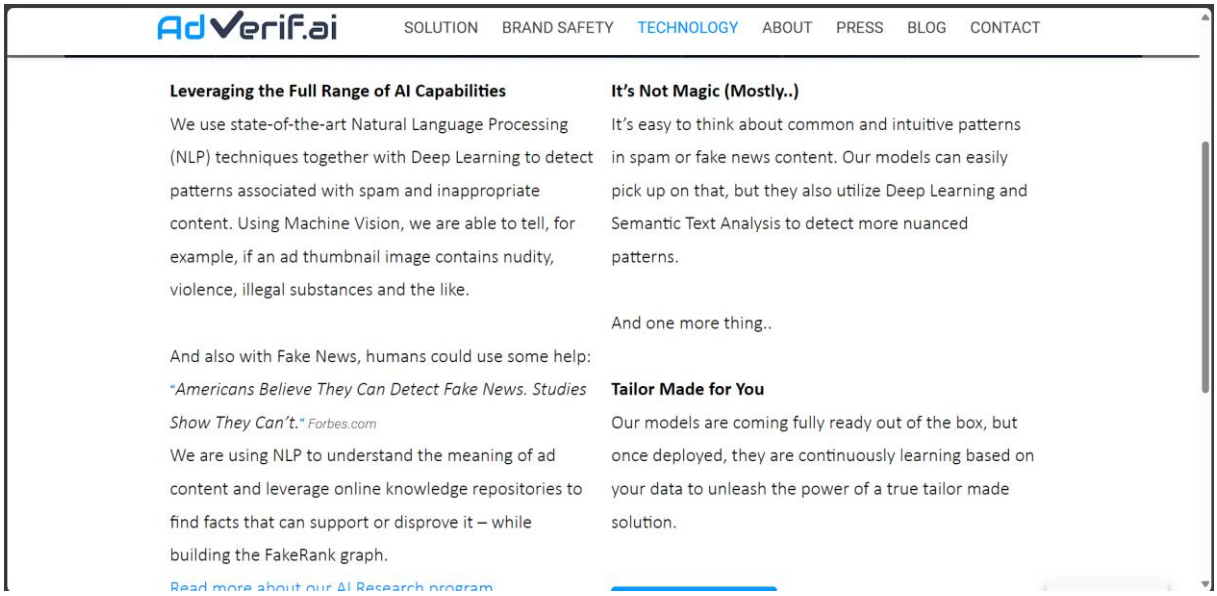


Рис. 1.7. Сторінка сервісу Adverif.ai [18]

Більш загальним та універсальним проєктом є Alto Intelligence. Компанія використовує штучний інтелект, щоб надати практичну інформацію про ризики [19]. Інструменти попередження ризиків дозволяють користувачам відображати та групувати інформацію навколо певних тем. Інструменти Alto можуть аналізувати інформацію понад 50 мовами та графічне представлення, дозволяючи користувачеві краще зрозуміти, на що орієнтувати свою рекламу чи комунікацію, або приймати інші стратегічні рішення.

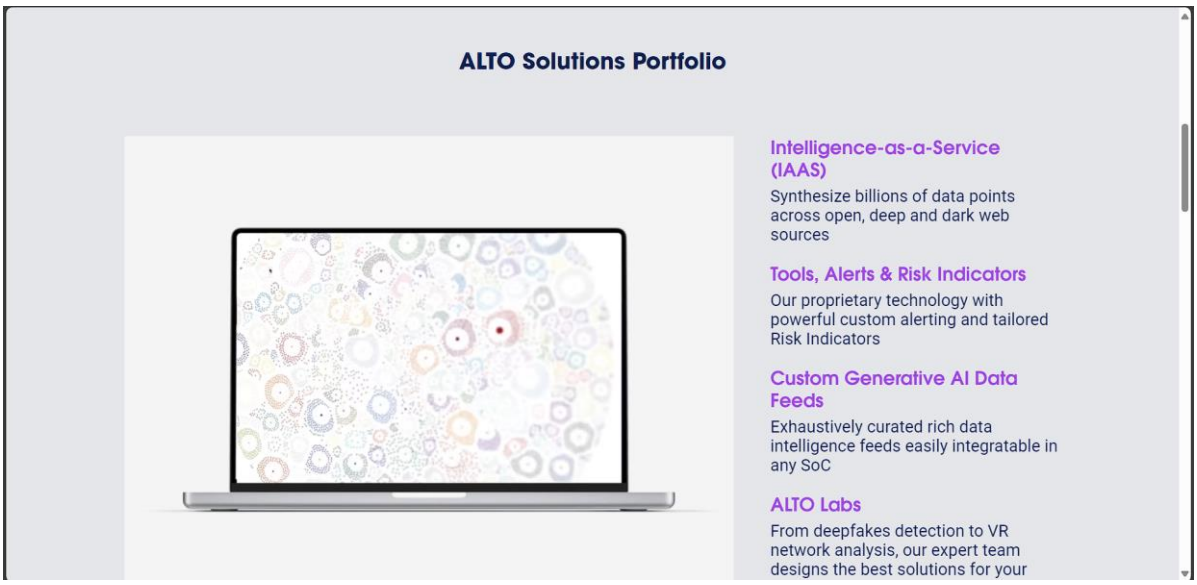


Рис. 1.8. Сторінка сервісу Alto Intelligence [19]

Blackbird AI називає себе «виявленням обману для інформаційної ери» [20]. Використовуючи алгоритми та інструменти на базі штучного інтелекту, сервіс виявляє розмови, які мають ознаки «змагального конфлікту». Інструмент може бути спеціально спрямований на придушення негативного сприйняття окремих організацій. Заснований у 2014 році Blackbird AI націлений на уряди, соціальні медіа-платформи, корпоративні організації, PR-компанії та медіа-організації.

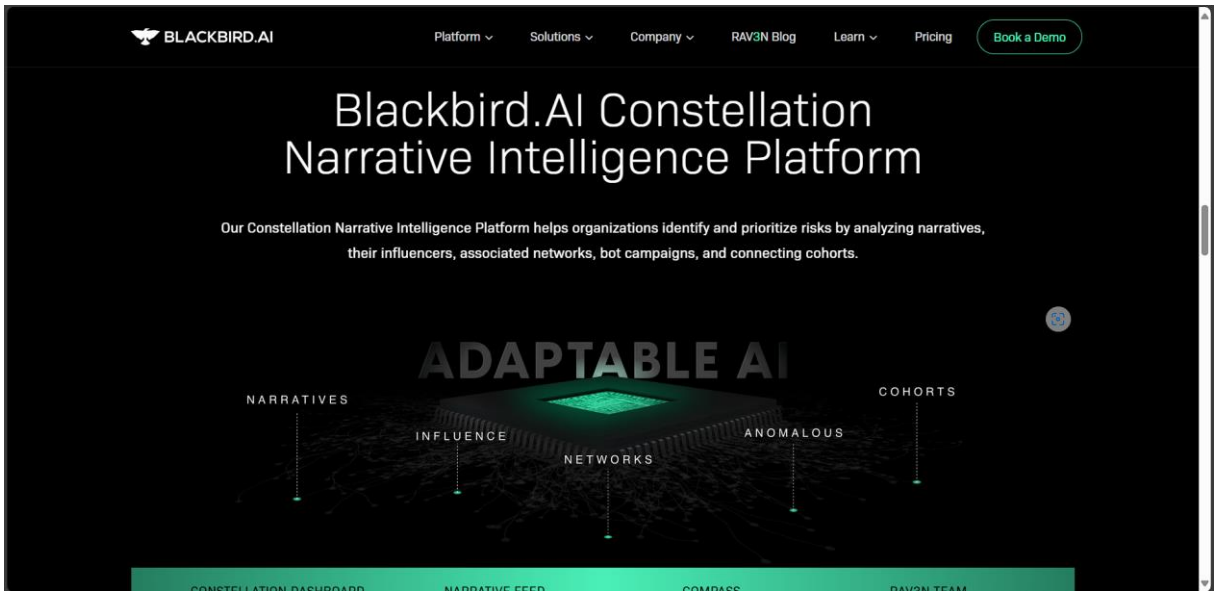


Рис. 1.9. Сторінка сервісу Blackbird [20]

Стартап Defudger працює з візуальними даними [21]. Defudger має здатність виявляти маніпуляції з відео та зображеннями. Це означає показ зображень, відредагованих за допомогою таких програм, як Photoshop, а також відео підробок. Defudger також перевіряє оригінальний візуальний вміст за допомогою технології блокчейн. База даних вмісту Defudger містить лише візуальний вміст, автентичність якого підтверджено за допомогою технології блокчейн. Це запобігає передачі дублікатів або зміненого вмісту як автентичного.

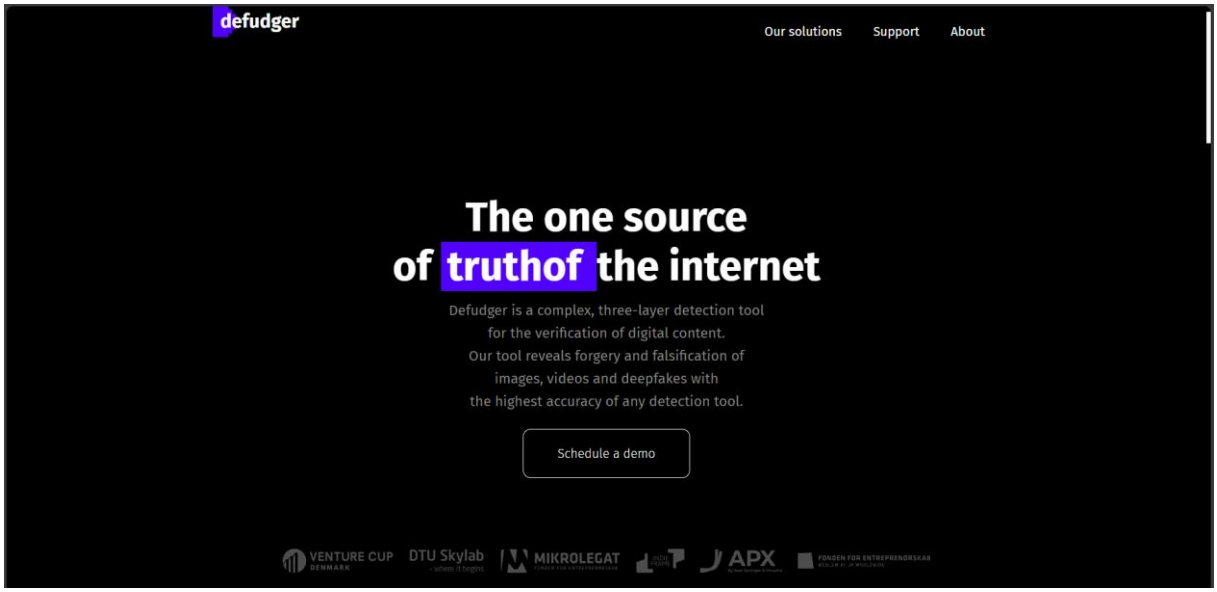


Рис. 1.10. Сторінка сервісу Defudger [21]

Новинний портал Snopes зосереджений на ручній ідентифікації недостовірних новин та використовує ряд піктограм для класифікації вмісту: Правда, Переважно правда, Змішаний вміст, Переважно хибний, Недостовірний, Непідтверджений, Застарілий, Неправильне зазначення, Правильне зазначення авторства, Неправильне приписування, Сканування та Легенда [22].

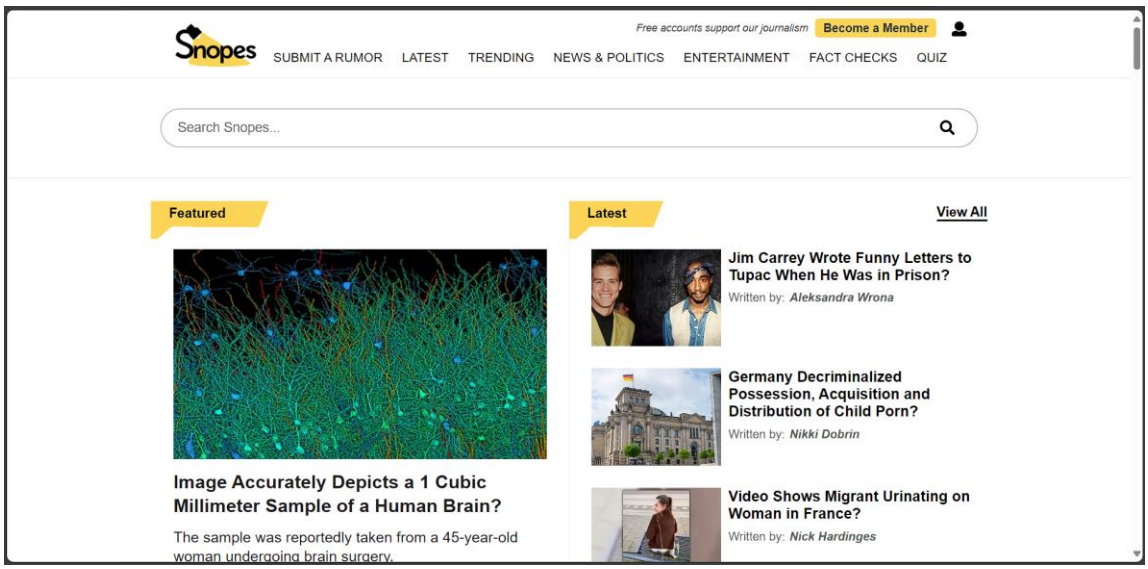


Рис. 1.11. Сторінка сервісу Snopes [22]

Краудсорсинговий проєкт CaptainFact надає користувачам широкий інструментарій для перевірки новин [23]. Проєкт містить розширення для вебпереглядача, яке забезпечує накладання відео на інтернет-відео з джерелами та контекстною інформацією, а також піктограми, що показують достовірність на основі голосів користувачів. Він також має «дебатну платформу», яка дозволяє обговорювати конкретні питання. Хоча наразі вони зосереджені на відео, вони розробляють інструмент для надання подібного накладення на статті.

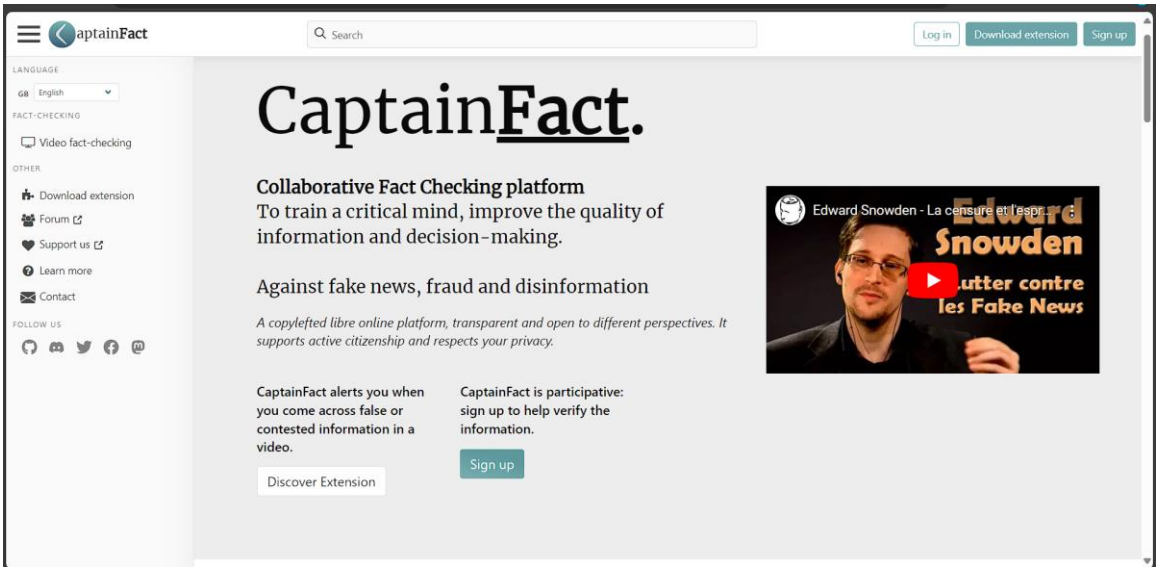


Рис. 1.12. Сторінка сервісу CaptainFact [23]

З метою аналізу політичних медіакампаній, у 2015 році був створений вебсайт MB Factchecking, та платний API для нього [24]. Вебсайт оцінює політичну прихильність та фактичну звітність різних медіа на основі чотирьох основних категорій: використання слів та заголовків, перевірка фактів та джерела, вибір історій та політична приналежність. MBFC також враховує кілька підкатегорій, таких як приховування упередження, вибір джерел та використання обтяжливої мови. Фактична звітність джерела оцінюється за семибальною шкалою від "Дуже високої" до "Дуже низької", а рейтинги політичного упередження класифікуються як "екстремально ліві", "ліві", "ліво-центристські", "найменші упередження", "право-центристські", "праві" і "екстремально праві". Крім того, MBFC використовує термін "Про-наука" для позначення науково обґрунтованих або законних наукових висновків та присвоює попереджувальні категорії, такі як "Змова / Псевдонаука", "Сумнівні джерела" та "Сатира" певним джерелам. Факт-чеки проводяться незалежними рецензентами, які працюють у Міжнародній мережі факт-чекінгу (IFCN), та дотримуються Принципів перевірки фактів IFCN, розроблених Інститутом Пойнтер.

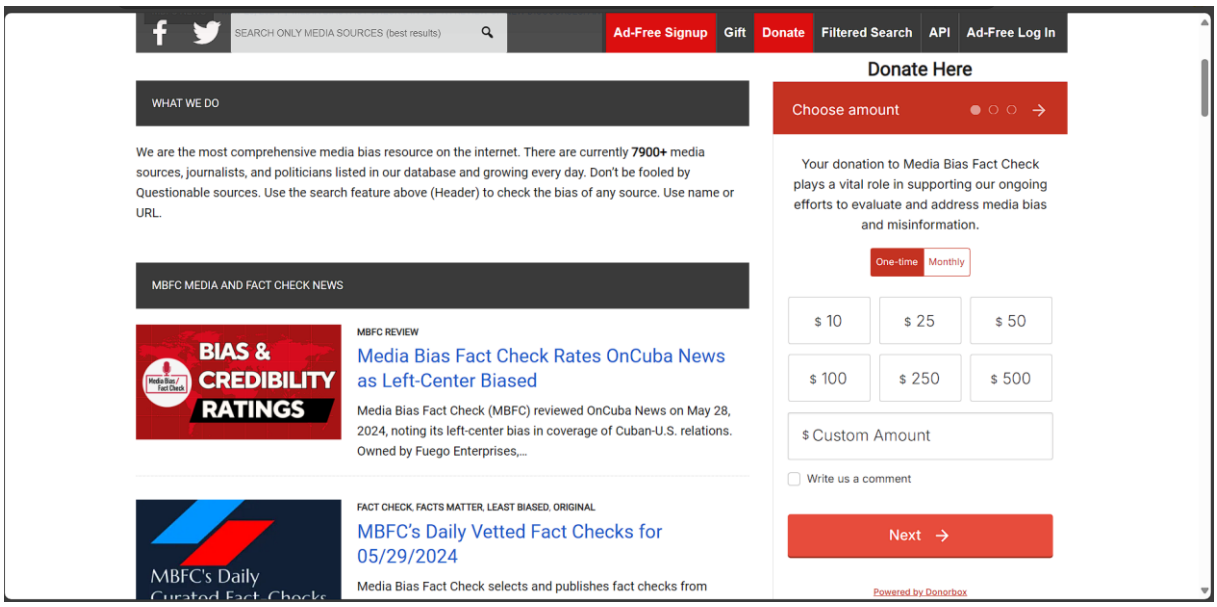


Рис. 1.13. Сторінка сервісу MBFC [24]

Болючою темою є агресивна кампанія поширення недостовірної інформації злочинною владою російської федерації. На протидію цьому була створена низка сервісів, серед яких окремо можна виділити Hamilton 2.0 [25]. Вебінструмент спрямований на моніторинг та аналіз російської пропаганди та дезінформації в Інтернеті, а також відстеження сотень російських акаунтів у Twitter, які прагнуть впливати на інформацію в Сполучених Штатах та Європі. Аналізуючи повідомлення та теми, що просуваються російською владою та державними медіа на різних платформах, включаючи Twitter, YouTube, телебачення та державні новинні вебсайти, "Hamilton 2.0" прагне підвищити усвідомленість про активні дії державної інформаційної операції. Ці операції можуть включати поширення хибної чи зманіпульованої інформації з метою виявлення тенденцій та тактик, використовуваних для впливу на західну аудиторію в соціальних мережах. Інструмент фінансується Альянсом забезпечення демократії та Фондом Маршала Німеччини Сполучених Штатів, що свідчить про його важливість у загальних зусиллях боротьби з дезінформацією та просуванням правдивої і прозорої інформації в цифровому просторі.

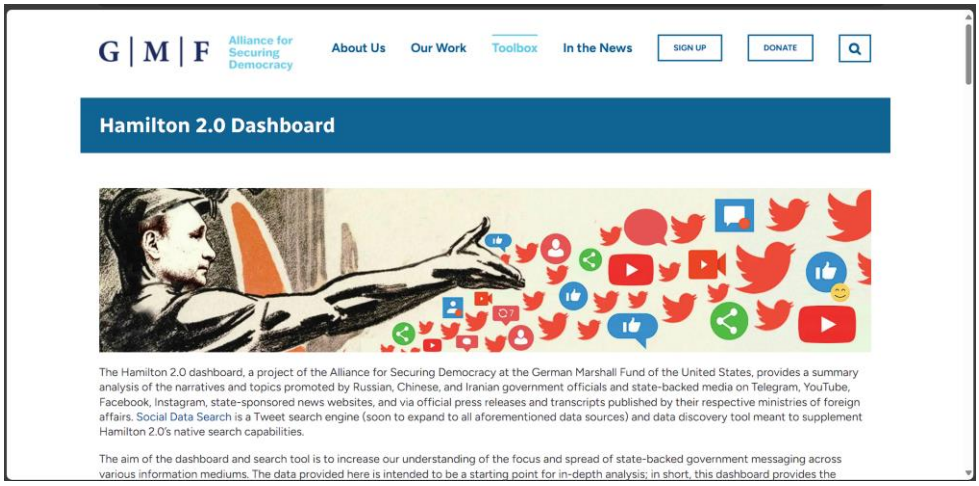


Рис. 1.14. Сторінка сервісу Hamilton 2.0 [25]

Iffy Quotient [26] – це вебінструмент, який використовує алгоритм NewsWhip [27] для запитів у Facebook і Twitter і ідентифікації URL-адрес,

які, як відомо, є упередженими або часто повідомляють недостовірну інформацію. Інструмент розраховує відсоток URL-адрес на кожному сайті, які є "невідомими" або відомими тим, що повідомляють недостовірну чи оманливу інформацію.

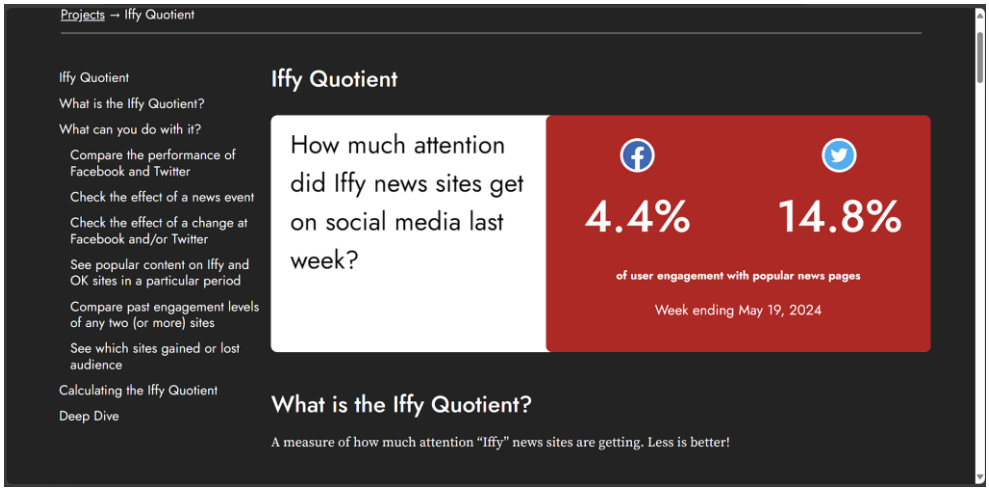


Рис. 1.15. Сторінка сервісу Iffy [26]

Politifact [28] – це вебсайт, який перевіряє факти «вагих і значущих» заяв і оцінює ці заяви як «Правдиві», «Здебільшого правдиві», «Наполовину правдиві», «Здебільшого Недостовірні», «Недостовірні» та «Штани вогню». Процес передбачає перегляд інших джерел перевірки фактів, пошук у Google, онлайн-пошук баз даних, консультації експертів та інші огляди літератури.

Рис. 1.16. Сторінка сервісу Politifact [28]

Polygraph.info – це вебплатформа для перевірки фактів, яка покладається на журналістів для вивчення урядових заяв і звітів, а також заяв високопоставлених осіб. Він класифікує звіти на основі міри достовірності та надає додатковий контекст [29].

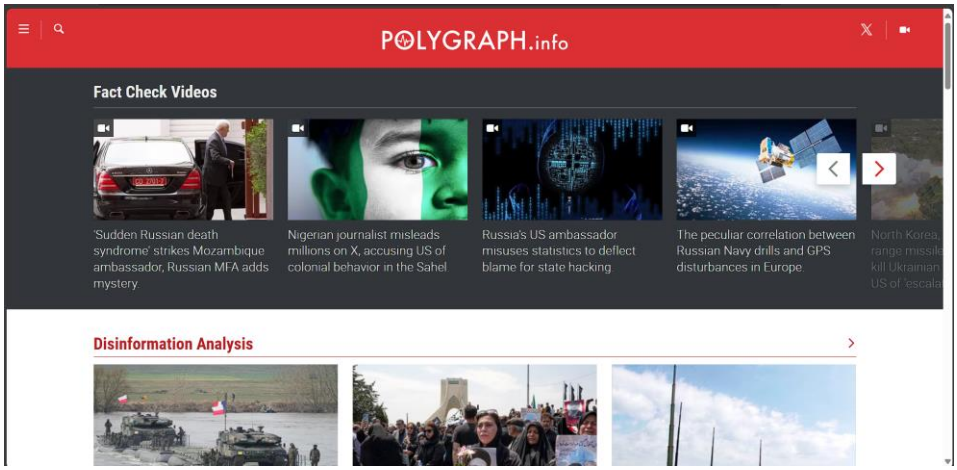


Рис. 1.17. Сторінка сервісу Polygraph [29]

Factcheck.org – це вебсайт, пов'язаний з Анненбурзьким центром громадської політики Університету Пенсільванії, який проводить перевірку фактів. Метою вебсайту є «моніторинг фактичної точності», яка присутня в американській політиці. Вебсайт зосереджений на заявах американських політиків. У групі є кілька журналістів, які досліджують матеріал [30].

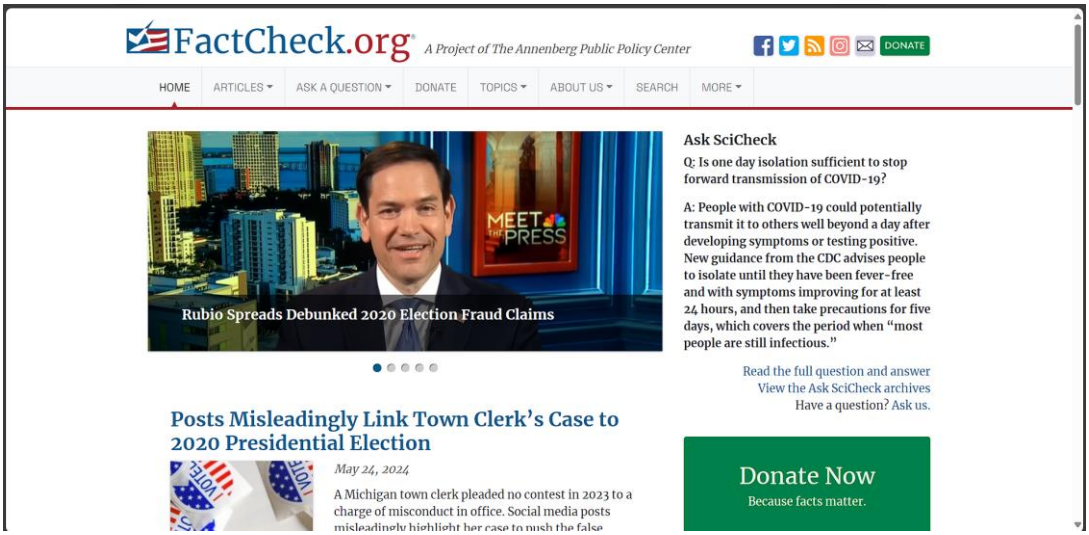


Рис. 1.18. Сторінка сервісу Factcheck.org [30]

1.3. Аналіз останніх наукових досліджень

У дослідженні [31] вивчається, як технології призвели до переміщення споживання новин з друкованих ЗМІ на платформи соціальних мереж, що створило проблему поширення недостовірних новин. Для вирішення цієї проблеми були використані алгоритми машинного навчання, такі як KNN, SVM, дерево рішень, Naïve B та логістична регресія, щоб відрізнити недостовірні новини від справжніх у наборі даних. Методи попередньої обробки даних використовувалися для підвищення ефективності алгоритму, особливо при обробці незбалансованих даних. Результати свідчать про високу точність: 89,98% для KNN, 90,46% для логістичної регресії, 86,89% для наївного Байєса, 73,33% для дерева рішень і 89,33% для SVM.

З часом обсяг текстових даних різко зріс, паралельно з розвитком штучного інтелекту та обчислювальних потужностей. Недостовірні новини стали глобальною проблемою з далекосяжними наслідками. У статті [32] досліджується, як обробка природної мови та методи машинного навчання можуть вирішити цю проблему. Для навчання даних за п'ятьма класифікаторами використовуються такі методи, як мішок слів, n-грами та векторизатори підрахунку, а також метод «частота терміна – частота документа, обернена до частоти терміна» (TF-IDF). Для оцінки ефективності кожної моделі для маркованих новинних повідомлень використовуються такі метрики оцінки, як точність, пригадування та оцінка F1. Бібліотека Python Scikit-learn пропонує широкий спектр сучасних алгоритмів машинного навчання для середньомасштабних контрольованих і неконтрольованих завдань. Робота [33] має на меті зробити машинне навчання доступним для неспеціалістів за допомогою зручної мови високого рівня. Основна увага приділяється продуктивності, ретельній документації та узгодженим API.

Мова Python має мало залежностей і є вільно доступною під спрощеною ліцензією BSD, що сприяє її використанню як в академічному, так і в комерційному середовищі. Ансамблеві методи об'єднують декілька

класифікаторів для прогнозування шляхом агрегування їхніх результатів. Вони включають такі методи, як усереднення Байєса, кодування вихідних даних з виправленням помилок, пакування та бустінг. У статті [34] розглядаються ці методи і обговорюється, чому ансамблі часто перевершують індивідуальні класифікатори. У ній розглядаються попередні дослідження, що порівнюють ансамблеві методи, і наводяться нові експерименти, які допомагають зрозуміти, чому Adaboost не може швидко переналаштовуватися. Останніми роками проводяться широкі дослідження в галузі обробки природної мови (NLP), зосереджені на етапах обробки тексту від підготовки до розуміння. Моделювання векторного простору має на меті представити слова в мовному корпусі у вигляді векторів, групуючи схожі значення близько один до одного. Два найпоширеніші підходи до векторизації – слово-орієнтований, що враховує весь мовний контекст, і документ-орієнтований, що фокусується на корпусі аналізованих текстів.

У статті [35] порівнюються різні методи векторизації тексту в NLP, зокрема в Text Mining, з використанням простих методів класифікації, таких як NBC і KNN. Оцінюючи точність класифікації, дослідження закладає основу для подальших досліджень в галузі автоматизованого аналізу текстів. У статті [36] обговорюється зростання кількості недостовірних новин у соціальних мережах, відзначаючи їхній потенційний негативний вплив. У ній висвітлюються труднощі у виявленні недостовірних новин через навмисне оманливий контент і складний характер залучення користувачів. В огляді представлено огляд існуючих досліджень щодо виявлення недостовірних новин, які охоплюють психологічні та соціальні теорії, алгоритми інтелектуального аналізу даних, метрики оцінювання та набори даних. Він також визначає відкриті проблеми та майбутні напрямки досліджень у цій сфері.

Дослідження [37] заглиблюється у виявлення недостовірних новин, їхніх авторів та суб'єктів в онлайн-соціальних мережах, оцінювання їхньої ефективності. Автори розглядають проблеми, що виникають через

різноманітну природу недостовірних новин і взаємозв'язок між статтями, авторами та суб'єктами. Завдяки ретельному аналізу даних у статті представлено FakeDetector – інноваційну модель для автоматичного визначення достовірності недостовірних новин. FakeDetector використовує явні та приховані ознаки текстової інформації для побудови моделі глибокої дифузної мережі, одночасно вивчаючи репрезентації новинних статей, авторів та суб'єктів. Широкі експерименти на реальних наборах даних недостовірних новин порівнюють FakeDetector з кількома найсучаснішими моделями, демонструючи його ефективність.

У статті [38] досліджується сатира як унікальна форма обману, що протиставляється легітимним новинам. У ній представлено алгоритм на основі SVM, збагачений п'ятьма прогностичними ознаками, і протестовано їхні комбінації на 360 новинних статтях. Найкраща комбінація ознак (абсурдність, граматика та пунктуація) досягає 90% точності та 84% відтворюваності у виявленні сатиричних новин. Алгоритмічна ідентифікація сатири може допомогти пом'якшити її оманливий вплив.

У дослідженні [39] розглядаються методи виявлення недостовірних новин у соціальних мережах шляхом аналізу різних ознак, витягнутих з новинних історій, включаючи джерела та пости в соціальних мережах. У роботі представлено новий набір ознак та оцінено ефективність існуючих підходів для автоматичного виявлення недостовірних новин. Результати дають уявлення про ефективність різних ознак у виявленні недостовірних новин. Стаття завершується обговоренням практичного застосування методів виявлення недостовірних новин, окреслюючи виклики та можливості.

У статті [40] представлено підхід машинного навчання для виявлення недостовірних новин із використанням нового алгоритму класифікації на основі мереж Байєса. Застосовуючи цей алгоритм до двох наборів даних новинних статей, автори досягають високої точності, перевершуючи інші існуючі моделі виявлення недостовірних новин. Наголошуючи на

зростаючій проблемі недостовірних новин та їхньому згубному впливі на суспільство, автори статті закликають продовжувати дослідження для розробки більш ефективних методів виявлення та стримування їхнього поширення.

У статті [41] автори запропонували систему, яка покращує виявлення недостовірних новин шляхом інтеграції двох методів вилучення текстових ознак і двох алгоритмів машинного навчання для класифікації тексту. Були використані базові методи попередньої обробки, а вилучення ознак проводилося за допомогою TF-IDF з Word2Vector. Вони об'єднали алгоритми KNN та Naïve Bayes в KNN Bayes, що перевершує традиційні методи. На відміну від багатьох існуючих програмних систем, їхня система поєднує в собі два методи вилучення ознак. Метрики оцінки, включаючи точність, достовірність, пригадування та оцінку F1, продемонстрували вищу продуктивність KNN Байєса над KNN. Крім того, аналіз «Площа під кривою-характеристики оператора» (AUC-ROC) виявив вищу AUC для KNN Байєса ROC-кривої порівняно з KNN, що ще більше підтверджує їхню модель.

У дослідженні [42] представлено керовану систему машинного навчання, спрямовану на виявлення недостовірних новин. Вона вирішує задачу ідентифікації прогностичних ознак для покращення продуктивності класифікатора, використовуючи як ознаки, що базуються на контенті, так і на користувачеві. Оцінюється важливість цих ознак та аналізується їхній вплив на продуктивність. Детально описано процес відбору остаточного набору ознак. Сім керованих класифікаторів, включаючи Naive Bayes, SVM, KNN, логістичну регресію, випадковий ліс, максимальну ентропію та умовний випадковий ліс, застосовуються та оцінюються з використанням набору даних PHEME. Проведено аналіз ознак, порівнюючи ефективність ознак, що базуються на контенті, та ознак, що базуються на користувачеві, у визначенні достовірності. Виявлено, що Random Forest демонструє найвищу ефективність при використанні виключно користувацьких ознак,

досягаючи точності 82,2%. Поєднання контент-орієнтованих та користувацьких ознак дає найвищу точність 83,4% при використанні класифікатора Random Forest.

У статті [43] представлено систему категоризації заголовків новин з використанням алгоритмів машинного навчання. Після великого збору даних слідують етапи навчання і тестування, що включають завдання попередньої обробки, такі як токенізація, видалення цифр і виключення стоп-слів. Ручне створення набору стоп-слів підвищує продуктивність системи, а видалення стоп-слів відіграє вирішальну роль у виборі ознак. Оптимізація досягається за допомогою генетичних алгоритмів, що зменшує розмір елементів. Проведено порівняння між оптимізованими та неоптимізованими процесами. Набір даних складається із заголовків новин з різних бенгальських новинних порталів і сайтів, що демонструє ефективну ефективність категоризації.

У дослідженні [44] представлено модель для ефективного виявлення та класифікації оманливих новинних статей у соціальних мережах. Ознаки були вилучені з набору даних PHEME за допомогою ентропійного відбору, а потім нормалізовані за допомогою методів Min-Max нормалізації. Прогностична модель виявлення недостовірних новин була побудована як стековий ансамбль з трьох алгоритмів. Оцінка ефективності порівняно з існуючою моделлю використовувала метрики точності виявлення, чутливості та прецизійності, виявивши вищу на 17,25% точність виявлення та на 15,78% чутливість, хоча і з нижчою на 0,2% прецизійністю. Запропонована модель підвищує точність виявлення недостовірних новин, зменшуючи кількість недостовірних повідомлень.

У дослідженні [45] представлено нову модель машинного навчання, що використовує методи НЛП для виявлення «недостовірних новин» шляхом включення контентних і соціальних ознак. Модель демонструє вражаючу продуктивність, досягаючи середньої точності 90,62% і показника F1 Score 90,33% на стандартному наборі даних.

У дослідженні [46] наголошується на необхідності комплексного підходу, який поєднує аналіз моделей машинного навчання з міждисциплінарними дослідженнями в галузі людсько-машинної комунікації, для того, щоб зменшити поширення недостовірних новин за межами того, чого можна досягти лише методами, що базуються на даних. Цей підхід спрямований на вдосконалення систем виявлення недостовірних новин для покращення співпраці та взаєморозуміння між людиною і машиною. Дослідження підкреслює важливість розробки зрозумілих систем виявлення недостовірних новин, зосереджуючись як на структурній інтерпретованості моделей для з'ясування їхніх операційних принципів, так і на їхній поведінці, в тому числі на наданні причин для їхніх прогнозів. Запропоноване рішення – це система комунікації на основі людино-машинної теорії, яка використовує інтерактивну пояснювальну систему штучного інтелекту, що пропонує шлях до більш прозорого та ефективного виявлення недостовірних новин.

У статті [47] представлено новий метод класифікації тексту на основі нормалізованої відстані стиснення NCD з використанням популярного стискача gzip. Метод досягає точності, порівнянної з ненавченими нейромережевими класифікаторами на наборах даних з розподілом, і перевершує як навчені, так і ненавчені моделі на наборах даних без розподілу. Подальші дослідження включають розширення методу на нейронні компресори для тексту та вивчення його продуктивності на різних завданнях. Обмеженнями методу є складність обчислень KNN та зосередженість на класичній версії DNN. Етичні міркування підкреслюють вплив на навколишнє середовище та інклюзивність для мов з низьким рівнем ресурсів.

Стаття [48] заглиблюється в складну сферу виявлення неправдивих новин, використовуючи підхід, заснований на ансамблевих методах машинного навчання. Основна мета цього дослідження полягала в тому, щоб використати властивості тексту і методи ансамблів для підвищення

ефективності виявлення недостовірних новинних статей у різноманітних наборах даних. Завдяки стратегічному використанню таких алгоритмів, як логістична регресія, машини опорних векторів (SVM), багатошаровий персептрон (MLP), k-найближчих сусідів (KNN), випадковий ліс і бустинг, було ретельно вивчено продуктивність ансамблів, що навчаються, порівняно з індивідуальними моделями, акцентуючи увагу на таких показниках, як точність, достовірність, пригадування і оцінка F1, щоб оцінити ефективність механізмів виявлення. Наслідки дослідження виходять за рамки поточних висновків, натякаючи на потенційні майбутні напрямки досліджень, включаючи вивчення виявлення неправдивих новин в режимі реального часу в динамічних форматах, таких як відео, і стратегічне визначення ключових джерел, що впливають на поширення дезінформації в цифровому ландшафті.

Дослідження [49] представляє новий метод виявлення неправдивих новин за допомогою складної стратегії відбору ознак, яка поєднує кластеризацію за методом K-середніх та класифікацію за допомогою машини опорних векторів (SVM). Цей підхід спрямований на оптимізацію виявлення фейкових новин шляхом стратегічного зменшення складності набору даних за допомогою відбору релевантних ознак. Шляхом ретельного аналізу схожості ознак, кластеризації їх у значущі групи та відбору найбільш дискримінативних ознак метод підвищує ефективність і точність класифікації неправдивих новин, особливо при використанні потужного SVM-класифікатора. Комплексна оцінка цього методу демонструє його значні переваги над існуючими методами, підкреслюючи його здатність точно виявляти фейкові новини в різноманітних наборах даних, у тому числі отриманих із соціальних мереж.

У роботі [50] досліджується процес використання алгоритму Naive Bayes для боротьби з поширенням неправдивих новин. Дослідження проливає світло на проблеми, пов'язані з необхідністю використання великих наборів даних для навчання гібридних моделей, підкреслюючи

критичну роль точності даних у класифікації новин. Сегментуючи дані на тестові та навчальні набори, система використовує алгоритм Naive Bayes для ретельного аналізу ваги слів, розрізняючи важливу та тривіальну інформацію, щоб встановити достовірність новинного контенту. Завдяки стратегічному застосуванню методів машинного навчання система намагається захистити від поширення дезінформації та оманливих наративів, що в кінцевому підсумку має на меті підтримати цілісність інформації, якою обмінюються на цифрових платформах.

Дослідження [51] заглибилося у сферу аналізу настроїв у Твіттері, зокрема, пов'язаних з «Турецькою кризою 2018 року», за допомогою згорткових нейронних мереж та наївного класифікатора Байєса. Автори дослідження виконали низку методологічних кроків, що охоплюють пошук даних, попередню обробку, розробку моделей, навчання, тестування та візуалізацію, щоб ретельно оцінити та порівняти ефективність цих алгоритмів у розпізнаванні патернів настроїв. Емпіричні результати виявили помітну розбіжність у результативності: модель CNN продемонструвала похвальний показник точності 88%, перевершивши точність моделі NBC у 78% у завданнях аналізу настроїв. Ця розбіжність підкреслює чудові аналітичні можливості нейронної мережі CNN у розшифровці нюансів настроїв у даних Twitter, що свідчить про зростаючий потенціал методологій глибокого навчання в розгадці складної динаміки соціальних мереж. Перспективні напрямки подальших досліджень авторів передбачають збільшення обсягу даних твітів, вивчення різноманітних алгоритмів машинного навчання для всебічного порівняльного аналізу, а також потенційне втілення результатів цих досліджень у практичні програми, пристосовані для аналізу настроїв у динамічній сфері взаємодії в соціальних мережах.

У дослідницькій роботі [52] зроблено огляд онлайн пасивно-агресивних алгоритмів (PA) у трьох варіантах: PA-I, PA-II та PA-III. Експериментальні результати з алгоритмами PA-I і PA-II заглиблюються в

нюанси того, як різні значення параметрів впливають на рівень помилок, проливаючи світло на адаптивність алгоритмів та їхню продуктивність за різних умов. У багатокласових експериментах алгоритми PA демонструють свою ефективність на різноманітних наборах даних, таких як USPS і MNIST, ілюструючи їхню здатність ефективно вирішувати складні завдання класифікації. Отримання оновлень у PA-I та PA-II з використанням виразів закритої форми для навчання вагових векторів підкреслює аналітичну основу алгоритмів та їхню обчислювальну ефективність. Порівняння оновлень на основі прогнозу та максимальних втрат дає змогу зрозуміти сильні та слабкі сторони кожного підходу, пропонуючи особливості поведінки алгоритмів у різних контекстах та сприяючи глибшому розумінню їхніх стратегій оптимізації.

Стаття [53] заглиблюється у складну сферу семантичного аналізу, проливаючи світло на його ключову роль у точному класифікуванні тексту за допомогою асоціацій та словосполучень. Вона підкреслює важливість семантичного аналізу для навігації в таких складних мовах, як англійська, з її великою кількістю значень і синонімів. Імітуючи когнітивні процеси людського мозку, семантичний аналіз розглядає текст крізь призму правил і зв'язків, віддзеркалюючи нюанси способу обробки інформації. Дослідження підкреслює особливості аналізу настроїв, розкриваючи тонкощі точної класифікації даних, пов'язаних з думками, та заглиблюється в тонкощі узгодження ознак і описів – завдання, пов'язане зі складнощами, і складний процес асоціювання дескрипторів з отриманими властивостями, що може перешкоджати точності класифікації. У статті також розглядаються особливості порівняльних та об'єктивних речень, точність класифікації та сарказм, що підкреслює багатогранність моделей аналізу настрою та перешкоди, з якими вони стикаються на шляху до ефективного розшифрування нюансів настрою.

У дослідженні [54] розглядаються виклики, пов'язані з поширенням неправдивих новин у соціальних мережах, і підкреслюється гостра потреба

в надійних механізмах їх виявлення. Використовуючи передові технології навчання та інформацію про коментарі користувачів, дослідження має на меті розробити інноваційні рішення для виявлення та протидії поширенню дезінформації. Завдяки комплексному аналізу взаємодії користувачів, особливостей контенту та мережевих характеристик дослідження проливає світло на складну динаміку поширення неправдивих новин і підкреслює важливість вжиття превентивних заходів для захисту цілісності інформації, якою обмінюються в Інтернеті. Дослідження підкреслює потенціал використання технологій машинного навчання та штучного інтелекту для боротьби з неправдивими новинами та підвищення довіри до онлайн-контенту.

Дослідження [55] заглиблюється у сферу виявлення неправдивих новин через призму методологій машинного навчання з головною метою – підвищити точність виявлення дезінформації. Дослідження передбачає попередню обробку набору даних під назвою «Liar-dataset», ретельно анотованого мітками, що позначають достовірність новинних статей. Завдяки використанню спектру алгоритмів машинного навчання, таких як дерево рішень, випадковий ліс, SVM, Naive Bayes, KNN та XGBoost, набір даних піддається всебічному аналізу, щоб виявити патерни та характеристики, які вказують на неправдиві новини. Результати показали, що метод XGBoost виявився найкращим, продемонструвавши вражаючий показник точності, що перевищує 75%, перевершивши показники SVM та випадкового лісу та які коливаються навколо позначки 73% точності. Ефективність моделі у виявленні неправдивих новин значною мірою пояснюється інноваційною інтеграцією текстового аналізу повідомлень та доповненням різноманітних функцій, спрямованих на підвищення точності та надійності механізму виявлення.

Проаналізовані роботи в сукупності підкреслюють багатогранність проблеми виявлення неправдивих новин і різноманітність стратегій, що застосовуються для її вирішення. В умовах стрімкого поширення

недостовірної інформації на цифрових платформах дослідники почали розробляти складні моделі машинного навчання, здатні розрізняти достовірні та недостовірні новинні статті. Ці зусилля відображають глибоке занепокоєння соціальними наслідками неправдивих новин, визнаючи їхній потенціал сіяти розбрат, маніпулювати громадською думкою та підривати демократичні процеси.

Нагальність цього виклику ще більше посилюється широким розповсюдженням соціальних мереж як основного джерела споживання новин, де інформація може поширюватися з безпрецедентною швидкістю і в безпрецедентних масштабах.

У відповідь на це дослідники застосували низку алгоритмів класифікації, кожен з яких має свої сильні та слабкі сторони, від традиційних підходів, таких як Naive Bayes і логістична регресія, до більш просунутих методів, таких як ансамблеві методи і нейронні мережі. Мета полягає в тому, щоб використовувати прогностичну силу машинного навчання для виявлення тонких сигналів і шаблонів, що вказують на недостовірні новини. Виділення ознак є критично важливим компонентом цієї роботи, і дослідники вивчають безліч методів для вилучення значущої інформації з необроблених текстових даних. Аналізуючи лінгвістичні характеристики, семантичні зв'язки та показники соціальної активності, ці методи мають на меті зафіксувати тонкі нюанси, які відрізняють справжні новини від оманливих наративів.

Етапи попередньої обробки даних відіграють ключову роль у підвищенні якості виокремлених ознак. Такі методи, як токенізація, стеммінг і видалення стоп-слів, застосовуються для підвищення співвідношення сигнал/шум і поліпшення здатності моделі розрізняти релевантну і нерелевантну інформацію.

Використання штучного інтелекту (ШІ) стало ключовим у боротьбі з поширенням недостовірних новин завдяки його здатності швидко аналізувати величезні обсяги даних [56-57].

1.4. Висновки до розділу 1

Комерційні проєкти часто поєднують багато методів аналізу даних разом. Методи ручної перевірки найчастіше звертаються до тематично-агностичного підходу, мовного підходу, експертного/«бібліотечного» підходу, опирається на краудсорсинг/волонтерів, експертів, соціальні мережі, інформацію у відкритому доступі. Автоматизовані алгоритми звертаються до «мішку зі словами», глибокого синтаксу, семантичного аналізу, використовують спеціально створені, але також публічно доступні набори даних.

Вебсервіси оцінки неправдивих новин можуть використовувати різні підходи, включаючи ручну експертну перевірку та аналіз за допомогою алгоритмів, які фокусуються лише на виявленні недостовірних стверджень. Велика частина описаних вебсервісів перевірки новин спирається на ручну перевірку даних, частіше експертами. Це суттєво подовжує час аналізу новинних текстів, а особисті переконання експертів можуть вплинути на результат оцінки. Ручна експертна перевірка може бути обмеженою також через витратність за часом і ресурсами, а також обмежений спектр розглянутих методів обману.

Автоматизований аналіз позбавляється проблем часу та упередженості, проте не повністю, і має деякі недоліки. Точність та упередженість аналізу напряму залежить від характера даних, на яких модель для аналізу була побудована. Обмежені дані, обрані для визначення лише одного типу недостовірних новин, може пропустити інші форми обману, не враховувати контекст і не бути ефективним у виявленні суб'єктивної недостовірності. Такий підхід може недооцінювати складність та різноманітність джерел недостовірної інформації, що може призвести до недостатньо об'єктивної оцінки новин.

Незважаючи на прогрес у методах аналізу новинних текстів на предмет виявлення фальсифікацій, у цій сфері залишається кілька невирішених питань. Одна з ключових проблем пов'язана з динамічним

характером недостовірних новин, що вимагає постійної адаптації та вдосконалення методів їх виявлення. Крім того, притаманна мові складність і постійно мінливий ландшафт онлайн-комунікації створюють труднощі з точним розрізненням оманливого контенту від справжніх новин. Крім того, такі проблеми, як упередженість навчальних даних, швидке поширення недостовірної інформації в соціальних мережах і потреба в надійних оціночних показниках, є постійними перешкодами для розробки ефективних методів виявлення недостовірних новин. Вирішення цих невирішених питань має важливе значення для підвищення надійності та ефективності методів аналізу текстів у боротьбі з поширенням недостовірних новин.

2. МОДЕЛІ МАШИННОГО НАВЧАННЯ

2.1. Машинні алгоритми для перевірки новин

Існує доволі багато алгоритмів для аналізу текстових даних. Аналіз природньої мови є однією з найбільш поширених сфер використання машинного навчання.

Серед найкраще пристосованих алгоритмів для поставленої задачі дуже чітко виділяються використання наївного Баєсового класифікатора та метод опорних векторів. Однак не можна точно сказати, що один гарантовано кращий за інший: у багатьох дослідженнях результати точності відрізнялись. Окрім них також використовується згорткові нейронні мережі та методи кластеризації, проте останні хоч поступаються в точності в порівнянні з іншими методами.

2.2. Класифікатори та векторизатори

Найбільш популярним алгоритмом класифікації можна вважати Naive Bayes – це контрольований алгоритм машинного навчання, який використовується для завдань класифікації, зокрема, для класифікації тексту [58]. Naive Bayes є частиною сімейства алгоритмів генеративного навчання, він прагне моделювати розподіл вхідних даних певного класу або категорії. На відміну від дискримінаційних класифікаторів, таких як логістична регресія, він не вивчає, які ознаки є найважливішими для диференціації між класами.

Naive Bayes також відомий як ймовірнісний класифікатор, оскільки він базується на теоремі Байєса. Ця теорема, також відома як правило Байєса, дозволяє нам «інвертувати» умовні ймовірності.

Теорема Байєса відрізняється використанням послідовних подій, де додаткова інформація, отримана пізніше, впливає на початкову ймовірність:

$$\mathrm{P}(A|B) = \frac{\mathrm{P}(B|A) * \mathrm{P}(A)}{\mathrm{P}(B)}, \qquad (2.1)$$

де $P(A|B)$ – апостеріорна ймовірність, тобто ймовірність гіпотези A щодо події B, що спостерігається; $P(B|A)$ – ймовірність правдоподібності: ймовірність доказів, враховуючи, що ймовірність гіпотези істинна; $P(A)$ – це попередня ймовірність: ймовірність гіпотези до спостереження доказів; $P(B)$ – гранична ймовірність: вірогідність доказів.

Наївні байєсівські класифікатори отримали свою назву через кілька ключових припущень, які вони роблять, вважаючи їх "наївними". Одне з цих припущень полягає у тому, що предиктори в цій моделі умовно незалежні, тобто не мають жодних взаємозв'язків між собою. Крім того вважається, що всі ознаки мають однаковий вплив на результат.

Хоча ці припущення можуть не відповідати реальності (наприклад, наступне слово в електронному листі часто залежить від попереднього), вони спрощують проблему класифікації, роблячи її більш простою для обчислень. Отже, для кожної змінної потрібно враховувати лише одну ймовірність, що полегшує обчислення моделі. Незважаючи на це спрощення, алгоритм класифікації працює досить ефективно, особливо при малих об'ємах вибірки.

Існує кілька типів наївного байєсівського класифікатора [58]. Найбільш популярні типи відрізняються залежно від розподілу значень ознак.

- Гаусівський Naive Bayes використовується з розподілами Гауса, тобто нормальний розподіл і неперервні змінні. Ця модель підганяється шляхом знаходження середнього значення та стандартного відхилення кожного класу.
- Мультиноміальний Naive Bayes припускає, що ознаки походять із мультиноміальних розподілів. Цей варіант корисний під час використання дискретних даних, таких як підрахунок частоти, і зазвичай застосовується у випадках використання обробки природної мови та класифікації спаму.

- Бернулівський Naive Bayes використовується з булевими змінними, тобто змінними з двома значеннями, такими як True і False.

Машина, або метод опорних векторів (SVM) є алгоритмом машинного навчання, спрямованим на розв'язання складних завдань класифікації, регресії та виявлення викидів. Уперше метод опорних векторів був згаданий у 1963 році, коли його основна ідея була вперше озвучена Володимиром Н. Вапником та Олексієм Я. Червоненків [59]. Використовуючи моделі навчання під наглядом, метод визначає оптимальні трансформації даних, які встановлюють границі між точками даних на основі передбачених класів, міток або виходів. Метод широко застосовуються в таких сферах, як охорона здоров'я, обробка природної мови, програми обробки сигналів, розпізнавання мови та зображень. Основною метою алгоритму є ідентифікація гіперплощини, яка чітко розділяє точки даних різних класів. Гіперплощина локалізована таким чином, що найбільша відстань розділяє потрібні класи. Відстань означає максимальну ширину розрізу, який проходить паралельно гіперплощині без будь-яких внутрішніх опорних векторів. Такі гіперплощини легше визначити для лінійно роздільних задач; однак для реальних проблем або сценаріїв алгоритм SVM намагається максимізувати відстань між опорними векторами, що призводить до неправильної класифікації для менших ділянок точок даних.

Машини опорних векторів загалом класифікуються на два типи: прості або лінійні SVM, та нелінійні SVM.

Лінійний метод SVM відноситься до типу SVM, який використовується для класифікації лінійно розділених даних. Це означає, що коли набір даних можна розділити на категорії або класи за допомогою однієї прямої лінії, він називається лінійним SVM, а дані називаються лінійно різними або роздільними. Крім того, класифікатор, який класифікує такі дані, називається лінійним класифікатором SVM. Простий SVM використовується для проблем класифікації та регресійного аналізу.

Нелінійні дані, які не можна розділити на окремі категорії за

допомогою прямої лінії, класифікуються за допомогою ядра або нелінійної SVM. Тут класифікатор називається нелінійним класифікатором. Класифікація може бути виконана з нелінійним типом даних шляхом додавання ознак у вищі виміри, а не покладатися на 2D простір. Тут нещодавно додані функції відповідають гіперплощині, яка допомагає легко розділяти класи чи категорії. Нелінійні SVM зазвичай використовуються для вирішення проблем оптимізації, які мають кілька змінних.

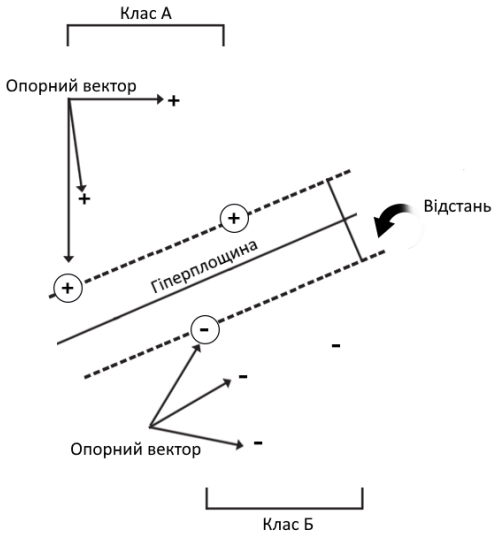


Рис. 2.1. Ідентифікація гіперплощини у методі опорних векторів

Метод опорних векторів належить до набору алгоритмів машинного навчання, що використовують методи ядра для перетворення функцій даних за допомогою функцій ядра. Функції ядра покладаються на процес відображення складних наборів даних у вищі виміри таким чином, що полегшує розділення точок даних. Ця функція спрощує межі даних для нелінійних задач, додаючи вищі розміри для відображення складних точок даних. При введенні додаткових параметрів дані не трансформуються повністю, оскільки вони можуть діяти як обчислювальний процес. Цю техніку зазвичай називають трюком ядра, коли перетворення даних у вищі розміри досягається ефективно та недорого.

Пасивно-агресивний класифікатор належить до сімейства алгоритмів, які зазвичай використовуються для масштабного навчання. Це алгоритми інтерактивного навчання, де вхідні дані надходять послідовно, і модель машинного навчання оновлюється крок за кроком, на відміну від пакетного навчання, де весь навчальний набір даних використовується один раз [52]. Це дуже корисно в ситуаціях, коли є велика кількість даних, і обчислювальні можливості не дозволяють навчити модель на всьому наборі даних через його величезний обсяг. Вони можуть реагувати на нові дані та адаптуватися до змін в режимі реального часу. Це може бути корисним, наприклад, для виявлення недостовірних новин на соціальному медіа, де нові дані додаються щосекунди. Пасивно-агресивні алгоритми схожі на модель персептрона в тому сенсі, що вони не вимагають коефіцієнта навчання, але вони включають параметр регуляризації. Пасивною модель називається тому, що якщо передбачення моделі є правильним, то модель залишається без змін і не вносить жодних корекцій. Іншими словами, дані в прикладі недостатньо для внесення змін у модель. Якщо модель вже правильно передбачає дані, немає необхідності її змінювати. Агресивною модель називається тому, що якщо передбачення моделі є неправильним, то вносяться зміни до моделі. Іншими словами, певні зміни в моделі можуть виправити помилку та покращити її точність. Якщо модель робить неправильні передбачення, то важливо вживати агресивні дії для корекції цих помилок.

Найбільш підходящим для бінарної класифікації є алгоритм логістичної регресії [60]. Модель логістичної регресії обчислює суму вхідних ознак (зазвичай включається також параметр зсуву), і обчислює логістичну функцію від результату. Для дослідження [60] використовується біноміальна логістична регресія, оскільки окремі набори даних були спрощені до бінарної класифікації.

Алгоритм логістичної регресії працює шляхом застосування рівняння з незалежними або пояснювальними змінними для передбачення значення

відповіді, і ознаки пов’язані за допомогою лінійного рівняння:

$$z = \beta_0 + \beta_1 x_1 + \cdots \beta_n x_n, \tag{2.2}$$

де β – параметри моделі; x – пояснювальне значення.

Для відображення передбачених значень в ймовірності використовується сигмоїдна функція. Ця функція відображає будь-яке дійсне значення в інше значення, яке знаходиться в діапазоні від 0 до 1. Сигмоїдна функція має невід'ємну похідну в кожній точці і рівно одну точку інфлексії. Функція має вигляд:

$$\mathrm{S}(x) = \frac{1}{1 + e^{-x}}. \tag{2.3}$$

Тоді модель логістичної регресії набуває вигляд:

$$\mathrm{P}(y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1)}}. \tag{2.4}$$

Наступний класифікатор – класифікатор k-найближчих сусідів – є непараметризованим лінивим алгоритмом класифікації з наглядом. Доволі простий і часто використовується для різних даних та задач регресії [61]. Метою алгоритму k-найближчих сусідів (KNN) є ідентифікація найближчих сусідів даної запитової точки з метою присвоєння їй мітки класу. Для досягнення цієї мети KNN має початкові вимоги до метрик відстані та k. Метрики відстані використовуються для того, щоб визначити які точки даних знаходяться найближче. В Scikit-learn, за замовчуванням використовується відстань Мінковського:

$$\mathrm{d}(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i| \right)^{\frac{1}{p}}. \tag{2.5}$$

Параметр p в Scikit-learn за замовчуванням має значення 2, тому формула має вигляд:

$$\mathrm{d}(x, y) = \sqrt{\sum_{i=1}^{n} (y_i - x_I)^2}, \tag{2.6}$$

що є простою формулою Евкліда відстані між двома точками на відрізку.

Достатньо популярним алгоритмом навчання з наглядом, який використовується для проблем регресії та класифікації/категоризації, є метод випадкового лісу [62]. Сам класифікатор є ансамблевим методом. Процес полягає у поділі даних на n частин та створення дерев рішень для кожної частини. Кожне дерево має свій підхід до класифікації даних, і остаточним рішенням класифікації є або найбільш популярний результат серед дерев, або середнє значення результатів.

Ансамблеві методи класифікації створюють кілька моделей і потім поєднують їх для отримання покращених результатів. Ансамблеві методи в машинному навчанні зазвичай дають більш точні рішення, ніж одна окрема модель. Ансамблеві методи можуть включати в себе різні підходи, такі як багатокласова класифікація [63]. Ідея полягає в тому, що, об'єднавши прогнози кількох моделей, можна отримати більш стійкий і точний результат, оскільки різні моделі можуть виявити свою ефективність в різних ситуаціях та знизити вплив шуму.

Ансамбль голосування [64] комбінує прогнози кількох моделей одночасно. Розрізняється жорстке та м’яке голосування. Жорстке голосування (голосування більшості), передбачає вибір найпопулярнішого передбачення серед методів. М’яке голосування, або зважене голосування, бере середнє значення передбачень належності даних до класу. У випадку аналізу на недостовірність новин, метод голосування порівняє середнє значення можливості належності даних до класу «недостовірності» до середньої можливості належності даних до класу «правди». На відміну від жорсткого голосування, м’яке бере до уваги передбачення кожної моделі й може побудувати більш точне передбачення в реальних умовах.

Алгоритми машинного навчання частіше за все приймають числові значення для тренування, тому потрібно перетворити всі текстові документи на їх числове представлення. Прикладом такого числового представлення є вектор, індексами якого є індекси унікальних слів з документів, а значеннями є кількість появ цих слів в документах.

Найбільш поширеним у машинному навчанні є векторизатор, що базується на TF-IDF (Term Frequency – Inverse Document Frequency) – показнику оцінки важливості слів в документах [65]. Важливість слова – кількість появи/вживання слова в документах.

$$\mathrm{TFIDF}(t,d) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t), \tag{2.7}$$

де $\mathrm{TF}(t,d)$ – кількість вживання слова в документах.

Під зворотньою частотою мається на увазі унікальність слова в колекції документів. Формула для зворотньої частоти має вигляд:

$$\mathrm{IDF}(t) = \log\left(\frac{|D|}{1 + |\{d \ : \ t \ \in \ d\}|}\right), \tag{2.8}$$

де $|\{d \ : \ t \ \in \ d\}|$ – вага, кількість унікальних документів, де зустрічається слово, а $|D|$ – кількість документів. Якщо слово не з'являється взагалі, то може відбутись поділ на 0, і тому до ваги додається 1.

Велике значення TF-IDF досягається, коли значення TF велике і коли вага в знаменнику в формулі IDF має мале значення.

Один із векторизаторів, що є вбудованим в бібліотеку Scikit-learn, і є найпростішим в імплементації, є кількісний векторизатор. Він створює з усіх документів матрицю, де кожне унікальне слово є стовпцем, а документи є рядками. Значення в кожній комірці є числовим і означає кількість разів коли зустрічається це слово в документі. В імплементації Scikit-learn слова не зберігаються як текст, а радше як індекси.

Векторизатор з хешуванням є модифікацією кількістного [66]. Найбільшою відмінністю від кількістного є те, що векторизатор з хешуванням не зберігає словник індексів слів.

Функція Murmurhash3 [67] хешує кожне слово/токен, які потім відображуються прямо в стовпець матриці. Це є перевагою, оскільки наявні датасети є доволі великими.

2.3. Згорткові нейронні мережі

CNN зазвичай використовується в сфері комп'ютерного зору. CNN є ядром більшості систем комп'ютерного бачення сьогодні, від автоматичного позначення фотографій Facebook до безпілотних автомобілів. Нещодавно CNN почали застосовуватись до проблем обробки природної мови та було отримано цікаві результати.

Згорткова нейронна мережа CNN працює, отримуючи зображення, призначаючи йому певну вагу на основі різних об'єктів зображення, а потім відрізняючи їх один від одного [68]. CNN вимагає дуже мало даних для попередньої обробки порівняно з іншими алгоритмами глибокого навчання. Однією з головних можливостей CNN є те, що він застосовує примітивні методи навчання своїх класифікаторів, що робить його достатньо хорошим для вивчення характеристик цільового об'єкта.

CNN базується на аналогічній архітектурі нейронів людського мозку, зокрема зорової кори. Кожен із нейронів дає відповідь на певний стимул у певній області зорової зони, визначеної як рецептивне поле. Ці колекції перекриваються, щоб охопити всю візуальну область.

CNN приймає зображення як вхідні дані, розрізняє свої об'єкти на основі трьох колірних площин і ідентифікує різні колірні простори. Нейронна мережа також вимірює розміри зображення. Зображення RGB мають різні кольори на основі триколірної площини: червоного, зеленого та синього. Визначаються різні колірні простори, у яких містяться зображення, наприклад RGB, CMYK, відтінки сірого та багато іншого. Це може стати виснажливим завданням під час вимірювання розмірів зображення, наприклад, якщо зображення має розмір 8k (коли одна сторона екрану має приблизно 8 тисяч пікселів). Ось одна зі зручних можливостей CNN: вона зменшує розміри зображення до такої міри, що його легше обробляти, що також зберігає всі його функції як єдине ціле. Це робиться для того, щоб отримати кращий прогноз. Ця здатність має вирішальне значення при розробці архітектур, які мають не тільки кращі функції навчання, але й

можуть працювати з масивними наборами даних зображень.

Ядро CNN працює на основі формули за розмірами зображення та кількістю каналів RGB. Ядро переміщується праворуч із визначеним значенням для «Кроку». Попутно він аналізує об’єкти зображення, поки не досягне повної ширини. Потім він стрибає до другого рядка ліворуч і рухається так само, як у верхньому рядку, доки не закриє все зображення. Процес повторюється, доки не буде проаналізовано кожну частину зображення.

Якщо існує кілька каналів, як, наприклад, у зображеннях RGB, тоді ядро містить таку саму глибину, що й у вхідному зображенні. Множення матриці реалізовано на основі числа Ks. Процедура виконується як у форматі стека , наприклад {K1, I1}, {K2, I2} тощо. Результати генеруються на основі підсумовування зміщення. Результат у формі стиснутого «каналу 1 глибини» заплутаних функцій.

Метою операції згортки є отримання всіх високорівневих характеристик зображення. Функції високого рівня також можуть включати краї зображення. Цей рівень не обмежується лише функціями високого рівня; він також виконує операції над функціями низького рівня, такими як колір і орієнтація градієнта. Ця архітектура виходить на новий рівень і, таким чином, включає ще два типи рівнів. Ці два шари відомі як дійсне заповнення та однакове заповнення.

Мета цих шарів – зменшити розмірність зображення, яке міститься у вихідному вхідному зображенні, і збільшити розмірність або, у деяких випадках, залишити її незмінною, залежно від необхідного результату. Такий самий відступ застосовується для згортання зображення до різних розмірів матриці, тоді як дійсне доповнення застосовується, коли немає необхідності змінювати розмір матриці.

Як ідентичний розпізнаному шару «згортки», головною метою рівня об’єднання є зменшення просторового розміру згорнутого об’єкта. Це використовується для зменшення необхідної обчислювальної потужності

для обробки даних методом зменшення розмірності. Крім того, це також корисно для вилучення домінуючих характеристик, які в основному є обертальними, а також позиційними інваріантами, тому необхідна ефективна підтримка процесу.

Замість пікселів зображення вхідними даними для більшості завдань НЛП є речення або документи, представлені у вигляді матриці [69]. Кожен рядок матриці відповідає одній лексемі, як правило, слову, але це може бути символ. Тобто кожен рядок є вектором, який представляє слово. Як правило, ці вектори є вбудованими словами (низькорозмірними представленнями), як-от word2vec або GloVe, але вони також можуть бути одноразовими векторами, які індексують слово в словнику. Для речення з 10 слів із використанням 100-вимірного вбудовування ми матимемо матрицю 10×100 як вхідні дані. Це наш «імідж».

У баченні наші фільтри ковзають по локальних ділянках зображення, але в НЛП ми зазвичай використовуємо фільтри, які ковзають по повним рядкам матриці (словам). Таким чином, «ширина» наших фільтрів зазвичай така ж, як і ширина вхідної матриці. Висота або розмір області може відрізнятися, але типовим є ковзання вікон, що містять 2-5 слів за раз.

Інваріантність розташування та локальна композиційність мали інтуїтивний сенс для зображень, але не настільки для НЛП. Для аналізу є дуже важливим розуміти, де в реченні з'являється слово. Пікселі, розташовані близько один до одного, ймовірно, будуть семантично пов'язані (частина одного об'єкта), але це не завжди вірно для слів. У багатьох мовах частини фраз можуть бути розділені кількома іншими словами. Композиційний аспект теж не очевидний. Зрозуміло, що слова складаються певним чином, як прикметник, що змінює іменник, але як саме це працює, що насправді «означають» уявлення вищого рівня, не так очевидно, як у випадку з комп'ютерним зором.

Враховуючи все це, здається, що CNN не підійдуть для завдань НЛП. Повторювані нейронні мережі мають більш інтуїтивний сенс. Вони

нагадують те, як ми обробляємо мову, або принаймні те, як ми думаємо, що обробляємо мову: послідовне читання зліва направо. Це не означає, що CNN не працюють. CNN, застосовані до проблем НЛП, працюють досить добре. Модель «мішок слів» є очевидним надмірним спрощенням із неправильними припущеннями, але, незважаючи на це, протягом багатьох років була стандартним підходом і дає досить хороші результати.

Великим аргументом на користь CNN є те, що вони порівняно з іншими моделями навчаються швидко. Згортки є центральною частиною комп'ютерної графіки та реалізовані на апаратному рівні на графічних процесорах. Порівняно з n-grams, CNN також ефективні з точки зору представлення. З великим словниковим запасом обчислення, що перевищує 3 грами, може стати дорогим. Навіть Google не надає понад 5 грамів. Згорткові фільтри автоматично вивчають хороші представлення, не потребуючи представлення всього словника. Багато вивчених фільтрів у першому шарі захоплюють функції, досить схожі (але не обмежуючись) на n-грами, але представляють їх у більш компактний спосіб.

2.4. Обґрунтування обраних методів дослідження

Порівняно з глибоким навчанням, моделями мови або згортковими/рекурентними нейронними мережами, звичайні класифікатори зазвичай відстають за точністю прогнозування.

Використання класифікаторів для даного дослідження було обрано з декількох причин.

По-перше, для тестування ансамблевих методів. Якщо одинарні класифікатори відстають за точністю від більш складних моделей, то можна спробувати використовувати кілька класифікаторів і вибирати найкращі прогнози для максимізації точності.

По-друге, класифікатори набагато простіше реалізовувати і швидше навчати, тому вони ідеально підходять для роботи в умовах обмеженого часу або коли потрібно тестувати інші налаштування чи параметри.

По-третє, класифікатори працюють набагато краще з невеликими наборами даних, ніж складніші моделі. В усіх використаних наборах даних обсягом досить невеликим і були взяті у готовому вигляді з окремих джерел, без використання додаткових засобів масового інформаційного пошуку або вебскраперів, і не мають масштабу для навчання складних моделей рівня ChatGPT, тому вони добре підходять для більш компактного дослідження. Навпаки, мовні моделі потребують досить великого обсягу даних для оптимальних результатів, що є додатковою проблемою.

У дослідженні тексти поділені на три категорії: неправдиві новини, сатиру та мову ненависті. Щоб спростити навчання та використання класифікаторів, було вирішено тренувати кожну категорію окремо, тобто спростити класифікаційну задачу до бінарної. Навчені класифікатори зберігаються для подальшого використання в ансамблевих методах. Також потрібно знайти ймовірність того, що текст належить до певної категорії.

Бібліотека Scikit-learn має вбудовану функцію predict_proba(), але вона не працює з усіма використовуваними класифікаторами, тому деякі з них були модифіковані відповідно.

Класифікатор Naive Bayes був обраний переважно через свою просту роботу з класифікацією, через його популярність та ефективність. Він підходить для класифікації коротких текстів при умові, що ознаки тексту не пов'язані між собою, але таке припущення не завжди вірне. Один з наборів даних заповнений коментарями з мережі X (Twitter), і цей фактор також вплинув на вибір класифікатора.

Багатоваріантний варіант Баєса, обраний для дослідження, може ефективно моделювати спільний розподіл ознак і пов'язані з ними класи, використовуючи багатовимірний розподіл. Частотний розподіл для кожного терміна може бути покращений шляхом включення показника варіації, такого як TF-IDF, який враховує кількість документів, в яких кожен з них зустрічається. Це може значно покращити продуктивність, надаючи більшого вагу термінам, які зустрічаються в меншій кількості документів,

тим самим покращуючи їх розрізняючі здібності.

Хоча багатовимірний розподіл добре працює при використанні напряму з частотами термінів, він також ефективно працює для дробових значень, таких як значення TF-IDF. Пасивно-агресивний класифікатор вважається одним з найкращих. Він часто використовується для розділення тексту на дві групи, тому дуже підходить для використання у цьому дослідженні. Крім того, він часто використовується для фільтрації спамових електронних листів та виявлення шахрайства. Його вибрано переважно через його новаторську природу, а також за його здатність працювати в сценаріях онлайн-навчання та пристосовуватися до змінних потоків даних.

Пасивно-агресивний (PA) класифікатор не має методу ймовірності, тому було обрано створення певного виду обгортки для нього:

Лістинг 2.1. Обгортка-метод ймовірності пасивно-агресивного класифікатора

```
def predict_proba (self , x_test):
    arr1= 1-(1. / (1. + np.exp (self.decision_function (x_test))))
    arr2= -(arr1-1)
    return np.stack ((arr2, arr1), axis =1)
```

Функція decision_function() вирішує, як алгоритм приймає рішення щодо класифікації та повертає масив numpy, де кожний елемент надається для значень x зразка x_test (дані для тестування) показує відстань від гіперплощини з різних сторін, зі значеннями від -1 до 1, при цьому 0 точно посередині. Метод опорних векторів SVM, хоча й повільніший за байєсівський класифікатор, вважається одним з найкращих і використовується як для класифікації, так і для регресії. Він був включений за свою надійність у роботі з високовимірними даними і ефективність у розділенні не лінійно роздільних класів. У бібліотеці Scikit-learn модуль SVM є обгорткою для бібліотеки libsvm і підтримує різні ядра, а не тільки лінійні. У даній роботі використовується функція LinearSVC(), яка підтримує тільки лінійне ядро, але працює швидше за модуль SVM. Потім

ця функція калібрується за допомогою CalibratedClassifierCV. Це було зроблено з метою отримання ймовірностей відповідності тексту до типу.

Логістична регресія підходить для бінарної класифікації і негайно повертає ймовірність. У бібліотеці Scikit-learn функція LogisticRegression() реалізує регуляризовану логістичну регресію за допомогою бібліотеки liblinear. Цей класифікатор був обраний через його інтерпретованість і придатність для бінарних класифікаційних завдань.

Класифікатор K-найближчих сусідів (KNN) є одним з найпоширеніших алгоритмів, використовуваних для класифікації тексту, і обраний за свою простоту і інтуїтивно зрозумілу концепцію класифікації на основі більшості його сусідів. У деяких конкретних випадках KNN може виявити себе краще за моделі типу BERT.

У 2023 році група дослідників з Університету Уотерлоо опублікувала прикладний код, де використовувався алгоритм з бібліотекою gzip для стиснення метрик відстаней. Це було зроблено для того, щоб довести, що схожість між двома текстовими документами тісно пов'язана з їх стисканням у файли меншого розміру.

Класифікатор випадкового лісу є популярним вибором для класифікації тексту, оскільки він добре справляється з великою кількістю текстових даних і може враховувати складні взаємозв'язки між словами в тексті та категоріями, до яких вони належать. Цей класифікатор був включений за його здатність до роботи з шумними та корельованими ознаками, уникаючи перенавчання, коли модель занадто складна і точно пасує до навчальних даних, що призводить до поганої продуктивності на нових даних.

У даному дослідженні використовуються наступні комбіновані ансамблеві методи:

- класифікатор Наївного Баєса, KNN, логістична регресія;
- класифікатор KNN, пасивно-агресивний класифікатор;
- SVM, пасивно-агресивний класифікатор, логістична регресія.

Ці комбінації були вибрані відповідно до різних, але не жорстких, принципів. Класифікатор Naive Bayes, KNN та логістична регресія є найпопулярнішими алгоритмами серед вибраних, тому було цікаво перевірити їх ефективність у порівнянні один з одним. SVM, пасивно-агресивний класифікатор та логістична регресія показали найкращі результати під час попередніх тестів на наявних наборах даних.

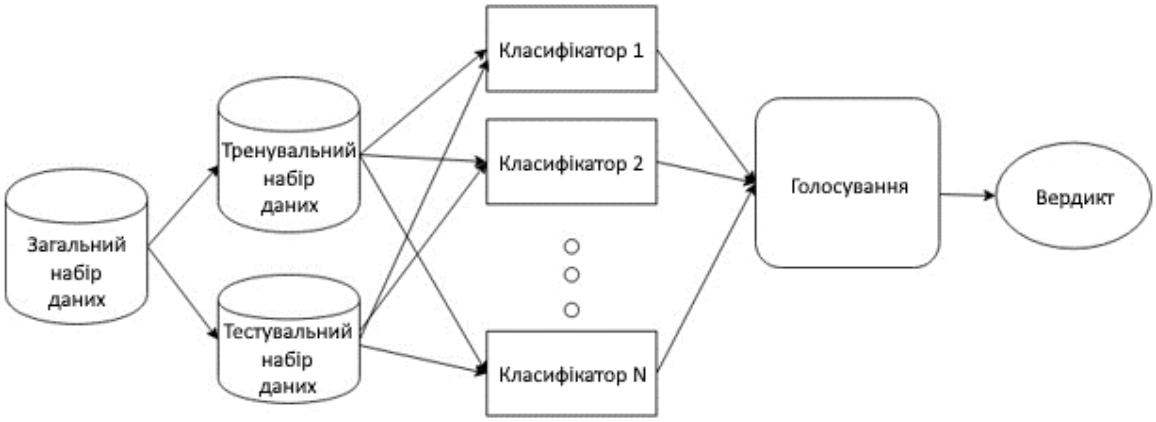


Рис. 2.2. Використання ансамблевого методу для виявлення неправдивого тексту

2.5. Висновки до розділу 2

За принципами роботи алгоритмів можна побачити, як вони використовують наведені методи. Оскільки алгоритми аналізують лише текст, вони не використовують тематично-агностичний підхід, оскільки він полягає в аналізі ознак поза головним текстом. Алгоритми в природі не є ручними методами аналізу, тому вони не є експертним підходом. Оскільки вони не використовують тематично-агностичний підхід, вони також не аналізують джерела та бібліотеки. Можна припустити, що метод опорних векторів є підходом «мішку зі словами», оскільки він бере до уваги екстремальні виключення з тексту. Класифікатор Байєса можна вважати і «мішком зі словами», оскільки він є наївним, але він також може вважатись семантичним, бо він рахує частоту використання слова.

3. ЗАПРОПОНОВАНИЙ ПРОГРАМНИЙ МЕТОД

3.1. Вимоги до програмного забезпечення

Розроблювана програмна система для аналізу текстів новин на неправдивість повинна:

- надавати можливість обробляти та аналізувати «сирий» текст новин ансамблевими методами класифікації;
- надавати результати за окремими, або за всіма категоріями разом;
- надавати результати вимірювання швидкості роботи моделей;
- надавати точність результатів класифікації.

3.2. Обрані засоби розробки програмного забезпечення

Мова Python є дуже гнучкою мовою програмування, але це не повністю пояснює її поширеність у сфері машинного навчання.

Однією з причин є історична. Найпершими великими конкурентами у сфері пошукових сервісів були Yahoo та Google. Обидві компанії мали плани стосовно використання мов високого рівня з відкритим вихідним кодом, оскільки час розробника був важливішим у цій галузі, ніж час процесора, і на той час було не так багато варіантів з необхідними веббібліотеками та числовими бібліотеками. Основними варіантами були Perl та Python; Yahoo обрали для інвестування Perl, а Google обрали Python. З часом Google витіснили Yahoo зі сфери пошукових сервісів. Це дозволило їм інвестувати значні кошти в розробку Python, числових бібліотек та бібліотек машинного навчання. Завдяки інвестиціям Google та розвитку екосистеми мови, інші великі компанії, такі як Facebook та Amazon, також почали використовувати Python, що допомогло поширенню мови.

Не зовсім зрозуміло, чи сама мова Python вплинула на потік робіт у машинному навчанні, чи навпаки, але типи проблем, які вирішує ця галузь, піддаються швидкому написанню одноразових скриптів.

Аналітики використовують блокноти Jupyter для дослідження даних,

бо вони дозволяють швидко створити скрипти різної складності, без попереднього планування складної архітектури ПЗ. Фахівці не зацікавлені у вивченні низькорівневих складних мов для підвищення продуктивності, особливо з огляду на те, що їхня мета – одноразовий експериментальний код.

Повертаючись до складності мов, Python синтаксично дуже нагадує псевдокод. Python набагато простіше зрозуміти, ніж C#, Java, C++, Go, R, Erlang. Але важливо те, що об'єкти Python можна легко генерувати з інших мов. Більшість бібліотек для машинного навчання написані на C, як-от NumPy. Ці бібліотеки дозволяють отримати доступ до високопродуктивних низькорівневих структур, таких як масиви C, з Python. Вони працюють так само, як списки в Python, але набагато швидше і не мають тієї складності, яка необхідна для правильної роботи з програмою на C або C++, яка може використовувати масиви C.

Бібліотека Scikit-learn є в якомусь сенсі виключенням, бо вона написана на Python. Це популярна бібліотека машинного навчання, яка має широкий асортимент алгоритмів, а також інструменти для візуалізації ML, попередньої обробки, підгонки моделі, вибору та оцінки.

Створена на основі NumPy, SciPy і Matplotlib, бібліотека Scikit-learn містить низку ефективних алгоритмів для класифікації, регресії та кластеризації. До них належать опорні векторні машини, випадкові ліси, посилення градієнта, k-середні та DBSCAN.

Scikit-learn може похвалитися своїм узгодженим і ефективно розробленим API, розширеній документації для більшості алгоритмів і численним онлайн-посібникам. Поточні випуски доступні для популярних платформ, включаючи Linux, MacOS і Windows.

Scikit-learn надає наступні можливості для побудови, підгонки та оцінки моделі ML:

- попередня обробка відноситься до інструментів Scikit-learn, корисних для вилучення функцій і нормалізації під час аналізу даних;
- класифікація відноситься до набору інструментів, які ідентифікують категорію, пов’язану з даними в моделі машинного навчання;
- регресія стосується створення моделі машинного навчання, яка намагається зрозуміти зв’язок між вхідними та вихідними даними, такими як поведінка або ціни на акції;
- інструменти кластеризації в Scikit-learn автоматично групують дані зі схожими характеристиками в набори, наприклад, дані клієнтів, упорядковані в набори на основі фізичного розташування;
- зменшення розмірності зменшує кількість випадкових величин для аналізу, щоб підвищити ефективність візуалізації, сторонні дані можна прибрати;
- вибір моделі стосується алгоритмів і їх здатності пропонувати інструменти, які порівнюють, перевіряють і вибирають оптимальні параметри для використання в проєктах машинного навчання даних;
- pipeline відноситься до утиліт для створення моделі робочого процесу;
- візуалізації для машинного навчання дозволяють швидко будувати графіки та візуально їх коригувати.

Scikit-learn використовує NumPy для високопродуктивної лінійної алгебри, а також для операцій з масивами. Деякі основні алгоритми навчання Scikit написані на Cython для підвищення загальної продуктивності. Як бібліотека вищого рівня, яка включає кілька реалізацій різних алгоритмів машинного навчання, Scikit-learn дозволяє користувачам

створювати, навчати та оцінювати модель у кількох рядках коду.

Scikit-learn надає єдиний набір API високого рівня для побудови конвеєрів або робочих процесів ML. Конвеєр Scikit-learn ML Pipeline використовується щоб передати дані через трансформатори для вилучення ознак і оцінювача для створення моделі, а потім оцінити прогнози для вимірювання точності моделі.

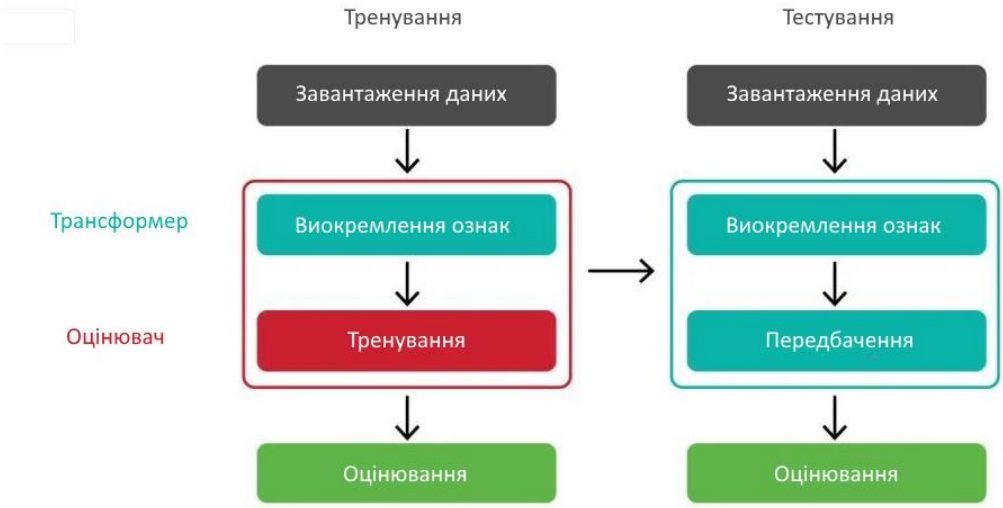


Рис. 3.1. Загальні алгоритми тренування та тестування

Алгоритм, який перетворює або вводить дані для попередньої обробки, називається трансформером.

Алгоритм машинного навчання, який навчає або підбирає дані для створення моделі, яку можна використовувати для прогнозів, називається оцінювачем.

Конвеєр об’єднує трансформатори та оцінювачі, щоб визначити робочий процес машинного навчання.

Для розробки серверної частини вебсервісу для тестування алгоритмів було використано мікрофреймворк Flask, призначений для швидкого й простого створення та масштабування вебдодатків. Його класифікують як мікрофреймворк, оскільки він не потребує спеціальних інструментів чи бібліотек. Він не має рівня абстракції бази даних, перевірки форм, а також

інших компонентів, де вже існуючі бібліотеки сторонніх розробників забезпечують деякі загальні функції. Крім того, він підтримує розширення, які можуть додавати функції програми, а також реалізовані в ньому. Існують розширення для об'єктно-реляційної системи, перевірки форм, обробки завантажень, а також різноманітних технологій відкритої автентифікації та кількох поширених інструментів, пов'язаних із фреймворками. Розширення оновлюється набагато частіше, ніж основна програма Flask.

Flask надає інструменти, бібліотеки та технології. Це дозволяє створювати вебдодаток, а також цей вебдодаток може бути вебсторінками, блогом, вікі-сайтом або бути настільки великим, як вебдодаток-календар або комерційний вебсайт.

Flask є частиною категорій, які називаються мікрофреймворком. Мікрофреймворк – це фреймворк із зовнішніми бібліотеками. У нього також є переваги і недоліки. Переваги полягають у тому, що фреймворк легкий, навіть є невелика залежність від оновлення та спостереження за помилками безпеки, тоді як недоліки полягають у тому, що іноді вам доведеться виконувати роботу самостійно, а також збільшується список залежностей, додаючи плагіни, які іноді дуже дратують.

Особливості Flask:

- можливість управління конфігурацією для реалізації життєвого циклу із певними етапами;
- самостійна програма Flask із Gunicorn;
- обслуговування запиту на статичні файли за допомогою Nginx;
- програма в контейнерах Docker на виділеному сервері Linux;
- налаштування та розгортання бази даних PostgreSQL для програми;
- налаштування черги завдань для обробки тривалих завдань.

3.3. Джерела даних для дослідження

Датасети, що містять тексти сатиричного характеру, можуть бути зібрані з різних джерел, включаючи сатиричні журнали, гумористичні блоги

та комедійні вебсайти. Такі тексти зазвичай містять висміювання, гумор та іронію, і їх розрізнення від справжніх недостовірних новин може бути викликом для моделі.

Для недостовірних новин можна використовувати датасети, зібрані з різних джерел, таких як агрегатори новин, соціальні медіа або спеціалізовані ресурси з виявлення недостовірних новин. Ці дані містять тексти, що мають недостовірний контент або викриваються як недостовірні після перевірки.

Важливо збирати дані з різних джерел та додатково проводити оцінку та перевірку достовірності даних. Комбінація датасетів, що представляють різні категорії сатири, недостовірності та мови ворожнечі, дозволяє моделі навчатися розпізнавати різні типи недостовірних новин і робити більш точні та надійні прогнози.

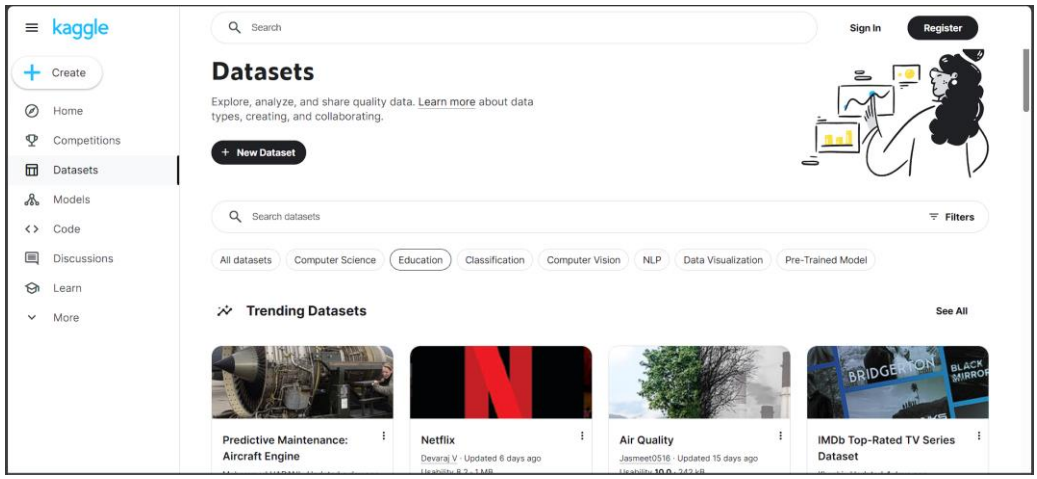


Рис. 3.2. Сторінка сервісу Kaggle

Використання відкритих джерел, таких як Kaggle [70], для перевірки недостовірних новин за критеріями сатири, недостовірності та мови ворожнечі має кілька обґрунтованих переваг.

Відкриті джерела, такі як Kaggle, надають доступ до широкого спектру даних. Це дозволяє тренувати модель на різних типах недостовірних новин, що допомагає зрозуміти та розпізнавати

характеристики кожного типу.

Використання відкритих джерел даних, таких як Kaggle, забезпечує можливість отримати велику кількість реальних прикладів недостовірних новин. Ці дані можуть бути зібрані з різних джерел, включаючи соціальні медіа, новинні сайти та інші ресурси. Це дозволяє побудувати модель, яка буде більш репрезентативною та адаптованою до реальних умов.

Використання відкритих джерел даних дозволяє проводити валідацію та порівняння результатів з іншими дослідниками та командами, які також працюють зі схожими даними. Це важливо для забезпечення надійності та достовірності моделі, а також для спільної роботи та обміну знаннями у галузі перевірки недостовірних новин.

Відкриті джерела даних можуть містити оновлені та актуальні набори даних, які відображають нові тренди та виклики у сфері недостовірних новин. Це дозволяє моделі бути оновленою та адаптованою до сучасних методів створення та розповсюдження недостовірної інформації.

3.4. Попередня підготовка датасетів для аналізу

Основна ідея полягає в аналізі новинного тексту, без сторонньої інформації (дати публікації, назви вебсайту, додаткових медіа). Текст аналізується за трьома категоріями окремо: правдивість новини, сатира, чи мова ворожнечі. Для навчання моделей за цими метриками використовувалися набори даних сервісу Kaggle, а для тестування в «реальних умовах» – довільно вибрані тексти новин і коментарів. Ці набори даних відформатовані за допомогою тексту в одній колонці та бінарної мітки в іншій колонці, яка означає, чи виконується певний критерій. Набір даних [71] містить 6335 рядків новинних текстів і міток «достовірність» або «недостовірність» Набір даних для перевірки на наявність сатири – це комбінація двох окремих наборів даних, один з новинної служби BBC [72], інший – з гумористичного сайту Onion [73]. Об'єднаний набір даних налічує 8341 рядок. Останній набір даних складався з коментарів мережі X і містив

понад 24 000 рядків [74]. Розглядаючи надійність даних, отриманих з Kaggle, порівняно з даними, отриманими безпосередньо з новинних вебсайтів, важливо визнати сильні та слабкі сторони кожного з підходів. Набори даних Kaggle можуть запропонувати зручність і доступність, часто надаючи впорядковані та попередньо оброблені дані, готові до аналізу. Однак надійність цих наборів даних може варіюватися залежно від таких факторів, як методи збору даних, методи попередньої обробки та надійність джерела даних.

У випадку новинних статей, отримання даних безпосередньо з авторитетних новинних вебсайтів, таких як BBC та The Guardian, може запропонувати більш надійну та актуальну інформацію порівняно з попередніми наборами даних на Kaggle. Такі вебсайти, як BBC та The Guardian, мають репутацію журналістської чесності та точності, що може підвищити надійність зібраних даних. Поважні інформаційні агентства, такі як Reuters, також є надійними джерелами даних завдяки своїм суворим редакційним стандартам і процесам перевірки фактів. Ці організації надають пріоритет точності та достовірності у своїх повідомленнях, що робить їхні дані цінними для дослідницьких цілей. Однак важливо враховувати потенційні проблеми та упередження, пов'язані з безпосереднім вилученням новинних вебсайтів. Такі фактори, як зміни структури вебсайту, невідповідність форматування контенту та потенційні проблеми з авторським правом, можуть вплинути на надійність і законність зібраних даних. Крім того, ручне вилучення даних може вимагати ретельної уваги до етичних міркувань, таких як дотримання умов користування сайтом і конфіденційності користувачів. До цього слід додати, що аналіз тексту українською мовою створює унікальні виклики порівняно з англійською мовою. По-перше, існує дефіцит україномовних наборів даних, доступних на таких платформах, як Kaggle, що обмежує доступ до вже існуючих даних для навчання та оцінки. Цей дефіцит вимагає або ручного збору даних, або використання методів трансферного навчання для

адаптації моделей, навчених на англомовних даних, до українського тексту. Крім того, семантичні відмінності між мовами можуть впливати на завдання аналізу тексту. Українська мова має власну синтаксичну та семантичну структуру, ідіоматичні вирази та лінгвістичні нюанси, які можуть не перекладатися безпосередньо на англійську мову. Ці відмінності можуть впливати на роботу алгоритмів класифікації текстів, навчених на англійських даних, при застосуванні до українського тексту, що потенційно може призвести до зниження точності та надійності.

Для дослідження використовувалися технології програмування мовою Python (sklearn, pandas, numpy), процесор AMD Ryzen 5 4500U 6 ядер, 16 гігабайт оперативної пам'яті.

Найбільшу складність у комбінованому аналізі метрик становить різноманітність даних для окремих метрик, оскільки дані для перевірки на недостовірність – це довгі формалізовані тексти, а для перевірки на мову ворожнечі використовуються коментарі користувачів.

Перед навчанням моделей тексти наборів даних мають бути певним чином змінені (підготовка даних). Підготовка даних – це процес підготовки «сирих» даних і перетворення їх у формат, придатний для моделей машинного навчання. Це перший і обов'язковий крок у створенні моделей. Особливо це стосується аналізу текстових даних людської мови. Такі текстові дані мають «шум», який проявляється в пунктуації, емоціях і регістрі тексту, а також у відмінюванні слів. Стоп-слова – це найпоширеніші слова в тексті, які не несуть жодної інформації. Оскільки мова наборів даних – англійська, до стоп-слів належать артиклі, «this», «where» тощо. Для видалення таких слів було використано бібліотеку NLTK. Вона містить приблизно 180 стоп-слів.

Рис. 3.3. Підготовка даних

Скорочення слів, або стеммінг, полягає у відокремленні та видаленні допоміжних частин слова від кореня, або основної форми. Для скорочення слів використовується бібліотека NLTK. Зокрема, у цьому дослідженні було використано лематизацію.

На відміну від звичайного стеммінгу, де під час скорочення можна втратити початковий контекст і значення, під час лематизації перед скороченням відбувається морфологічний аналіз слова, хоча це і займає більше часу.

Залежно від набору даних, цифри також видаляються або замінюються на слова в даних, або скорочені слова «розширюються».

Алгоритми машинного навчання найчастіше приймають для навчання числові значення, тому нам потрібно перетворити всі текстові документи в їх числове представлення. Таким числовим представленням є вектор, індексами якого є індекси унікальних слів з документів, а значеннями –

кількість входжень цих слів у документах. Під час векторизації стоп-слова також можуть бути видалені.

3.5. Навчання класифікаторів

У Python навчання класифікатора включає декілька кроків. По-перше, ми імпортуємо необхідні бібліотеки, такі як Scikit-learn, яка надає широкий спектр алгоритмів машинного навчання. Потім ми готуємо дані, розділючи їх на навчальні та тестові набори за допомогою таких функцій, як train_test_split(). Далі ми ініціалізуємо наш класифікатор і «підганяємо» його до навчальних даних за допомогою методу fit(). Під час цього процесу класифікатор вивчає закономірності та зв’язки в даних.

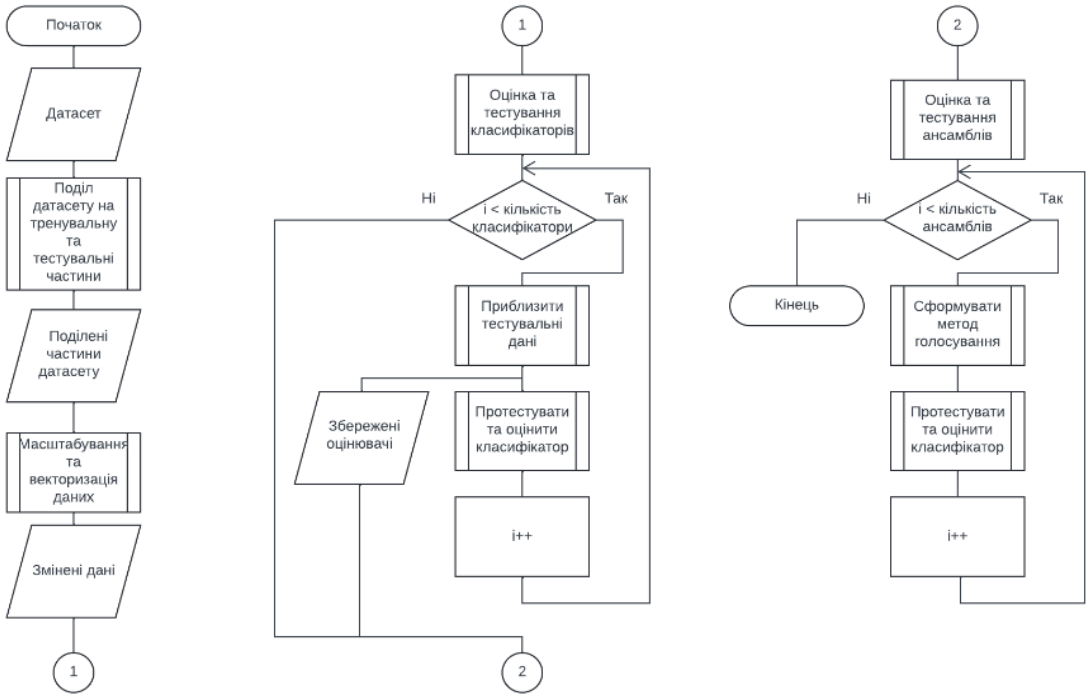


Рис. 3.4. Схема тренування всіх класифікаторів

Векторизація класифікатора передбачає перетворення необроблених текстових даних у числові характеристики, які можуть використовуватися алгоритмами машинного навчання. Зазвичай це реалізується за допомогою

вбудованих в бібліотеку машинного навчання методів імплеметації векторизаторів, наприклад CountVectorizer або TF-IDFVectorizer з бібліотеки Scikit-learn.

Лістинг 3.1. Обгортка-метод ймовірності пасивно-агресивного класифікатора

```
from sklearn.feature_extraction.text import TfidfVectorizer
corpus = ["This is the first document.", "This document is the second
document.", "And this is the third one.", "Is this the first document?"]
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus)
```

Навчання класифікатора виконується циклами, окремо для кожного класифікатора/комбінації, окремо для кожного набору даних і окремо за допомогою векторизаторів.

Після навчання класифікаторів проводиться їх тестування на прикладах кожної текстової категорії. Класифікатори повинні давати ймовірність того, що текст належить або не належить до обраної категорії.

Рис. 3.5. Тестування класифікаторів

Для цього використовується функція Scikit-learn predict() і predict proba(). Метод predict визначає клас даних, а метод predict proba визначає ймовірність того, що дані належать до певного класу для кожного з них, і

повертає масив цих значень.

Для оптимізації тестування оцінювачі, які містяться в ансамблевих методах, тимчасово зберігаються в пам’яті та викликаються пізніше для тестування та оцінки в комбінованих алгоритмах.

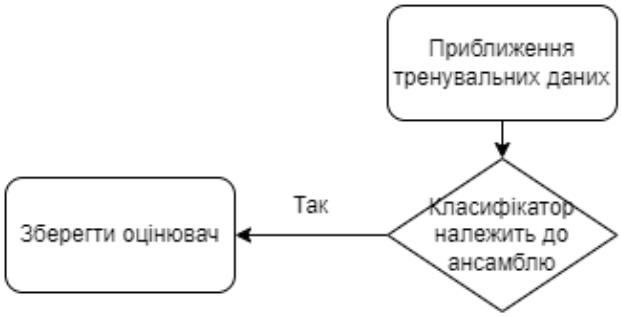


Рис. 3.6. Збереження ансамблевих оцінювачів

3.6. Архітектура розробленого програмного забезпечення

Сервіс перевірки текстів новин на недостовірність має архітектуру, яка включає в себе віддалений сервер з моделлю машинного навчання та API, написаний на Python з використанням фреймворку Flask.

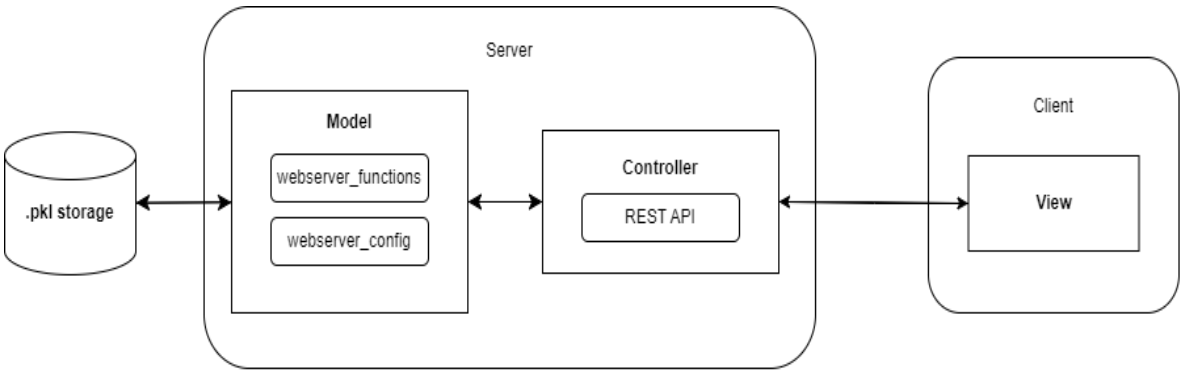


Рис. 3.7. Архітектура програмного забезпечення

Основні компоненти архітектури є наступними.

- Віддалений сервер, на якому розгорнуто API для обробки запитів. Сервер отримує запити від клієнтів, передає текст новини для

перевірки і повертає результати перевірки. Він реалізований з використанням фреймворку Flask.

- API для роботи з натренованими моделями. API дозволяє клієнтам взаємодіяти з сервісом, надсилаючи тексти новин та обираючи векторизатори, класифікатори та критерії оцінки.
- Сховище моделей, натренованих на кількох різних датасетах, які розрізняють різні типи недостовірних новин, включаючи сатиру, недостовірність, та радикальні тексти. Моделі експортовані в окремі файли за допомогою бібліотеки Pickle, і викликаються окремо в залежності від запиту користувача.

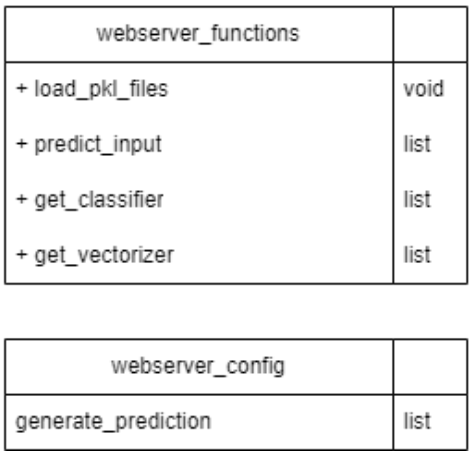


Рис. 3.8. Головні методи модулів

Така архітектура дозволяє клієнтам надсилати тексти новин або URL вебсторінок для перевірки на недостовірність. Застосовуючи модель машинного навчання, натреновану на різних типах недостовірних новин, сервіс може виявляти та ідентифікувати недостовірні тексти.

3.7. Висновки до розділу 3

У даному розділі були описані використані технології для розробки вебсервісу та для тренування класифікаторів. Для розробки сервісу було використано мову програмування Python та ряд популярних бібліотек, таких

як Scikit-learn, NLTK, NumPy, SciPy, Matplotlib, Keras. Проведено пошук та створення набору датасетів, який включав сатиричні новини, неправдиві новини та мову ворожнечі з сервісу Kaggle.

Архітектура програмного забезпечення включає віддалений сервер, на якому розгорнута модель машинного навчання. Модель, натренована на датасетах, що містять сатиричні, неправдиві, ворожі тексти, здатна розпізнавати типи недостовірних новин. Сервер використовує фреймворк Flask для створення API, що надає можливість клієнтам надсилати тексти новин або URL вебсторінок для перевірки. Текст передається моделі для аналізу. Додатково сервіс використовує бібліотеки Keras та Scikit-learn для тренування та застосування моделі машинного навчання.

Застосування Python, Flask, Scikit-learn дозволяє реалізувати потужний та гнучкий сервіс для перевірки недостовірних новин. Запропонована архітектура дозволяє забезпечити зручну комунікацію між клієнтами та віддаленим сервером, а також забезпечує точність та надійність перевірки текстів новин на недостовірність.

4. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1. Оцінка ефективності класифікаторів

Разом з ансамблевими методами, всього в роботі використано 9 класифікаторів. Класифікатор тренується окремо для кожного з 2 векторизаторів та кожного з 3 наборів даних. Натренований класифікатор тестується на додаткових 6 зразках даних.

	A	B	C	D	E	F	G	H	I	J
1	classifier	model-weigh	vectorize	datase	test-sample	wordcoun	required-predictic	predicte	values	speed
2	Naive Bayes	33555080	Hashing	fake1	a BBC article	351	0	1	[0.48202891 0.51797109]	0.040640354
3	Naive Bayes	33555080	Hashing	fake1	a Guardian article	207	0	1	[0.16912048 0.83087952]	0.027109861
4	Naive Bayes	33555080	Hashing	fake1	an Onion article	173	1	1	[0.14287339 0.85712661]	0.025998831
5	Naive Bayes	33555080	Hashing	fake1	a Clickhole article	223	1	1	[0.12636493 0.87363507]	0.027196646
6	Naive Bayes	33555080	Hashing	fake1	an angry copypasta	1753	1	1	[0.03034505 0.96965495]	0.028562929
7	Naive Bayes	33555080	Hashing	fake1	racist speech	158	1	1	[0.46582588 0.53417412]	0.026590347
8	Passive-Aggressiv	8389554	Hashing	fake1	a BBC article	351	0	1	[0.69748095 0.30251905]	0.001997709
9	Passive-Aggressiv	8389554	Hashing	fake1	a Guardian article	207	0	1	[0.60797763 0.39202237]	0.000999451
10	Passive-Aggressiv	8389554	Hashing	fake1	an Onion article	173	1	1	[0.74773386 0.25226614]	0.001507521
11	Passive-Aggressiv	8389554	Hashing	fake1	a Clickhole article	223	1	1	[0.62420636 0.37579364]	0.001001835
12	Passive-Aggressiv	8389554	Hashing	fake1	an angry copypasta	1753	1	1	[0.77662913 0.22337087]	0.002001288
13	Passive-Aggressiv	8389554	Hashing	fake1	racist speech	158	1	1	[0.53855545 0.46144455]	0.001003981
14	SVM	41945340	Hashing	fake1	a BBC article	351	0	0	[0.52581423 0.47418577]	0.007017612
15	SVM	41945340	Hashing	fake1	a Guardian article	207	0	1	[0.08029198 0.91970802]	0.004010201
16	SVM	41945340	Hashing	fake1	an Onion article	173	1	1	[0.1214276 0.8785724]	0.005089017
17	SVM	41945340	Hashing	fake1	a Clickhole article	223	1	1	[0.10626794 0.89373206]	0.003997326
18	SVM	41945340	Hashing	fake1	an angry copypasta	1753	1	1	[0.10504556 0.89495444]	0.004615545
19	SVM	41945340	Hashing	fake1	racist speech	158	1	1	[0.36569149 0.63430851]	0.004000425
20	Logistic Regressio	8389334	Hashing	fake1	a BBC article	351	0	0	[0.63917936 0.36082064]	0.002018929
21	Logistic Regressio	8389334	Hashing	fake1	a Guardian article	207	0	1	[0.17977204 0.82022796]	0.000998735
22	Logistic Regressio	8389334	Hashing	fake1	an Onion article	173	1	1	[0.19220679 0.80779321]	0.002007723
23	Logistic Regressio	8389334	Hashing	fake1	a Clickhole article	223	1	1	[0.15187128 0.84812872]	0.001033783
24	Logistic Regressio	8389334	Hashing	fake1	an angry copypasta	1753	1	1	[0.15511183 0.84488817]	0.002001288
25	Logistic Regressio	8389334	Hashing	fake1	racist speech	158	1	1	[0.34463756 0.65536244]	0.000997543
26	KNN	11897828	Hashing	fake1	a BBC article	351	0	0	[0.8 0.2]	0.313453436
27	KNN	11897828	Hashing	fake1	a Guardian article	207	0	0	[0.6 0.4]	0.113417625
28	KNN	11897828	Hashing	fake1	an Onion article	173	1	1	[0.2 0.8]	0.049681636
29	KNN	11897828	Hashing	fake1	a Clickhole article	223	1	1	[0. 1.]	0.043172836
30	KNN	11897828	Hashing	fake1	an angry copypasta	1753	1	1	[0. 1.]	0.12808099
31	KNN	11897828	Hashing	fake1	racist speech	158	1	0	[0.6 0.4]	0.03912878
32	Random Forest	1173604	Hashing	fake1	a BBC article	351	0	1	[0.43371988 0.56628012]	0.565265894
33	Random Forest	1173604	Hashing	fake1	a Guardian article	207	0	1	[0.41878565 0.58121435]	0.920092821
34	Random Forest	1173604	Hashing	fake1	an Onion article	173	1	1	[0.4113973 0.5886027]	0.772684336
35	Random Forest	1173604	Hashing	fake1	a Clickhole article	223	1	1	[0.41185061 0.58814939]	0.806729794
36	Random Forest	1173604	Hashing	fake1	an angry copypasta	1753	1	1	[0.44826669 0.55173331]	0.826175213
37	Random Forest	1173604	Hashing	fake1	racist speech	158	1	1	[0.42878922 0.57121078]	0.613699198
38	NP, LR, KNN	7016404	Hashing	fake1	a BBC article	351	0	1	[0.44995612 0.55004388]	1.233187199
39	NP, LR, KNN	7016404	Hashing	fake1	a Guardian article	207	0	1	[0.42336254 0.57663746]	1.231333971
40	NP, LR, KNN	7016404	Hashing	fake1	an Onion article	173	1	1	[0.41259471 0.58740529]	1.44809866
41	NP, LR, KNN	7016404	Hashing	fake1	a Clickhole article	223	1	1	[0.4001683 0.5998317]	1.847808123
42	NP, LR, KNN	7016404	Hashing	fake1	an angry copypasta	1753	1	1	[0.44771389 0.55228631]	2.057058334
43	NP, LR, KNN	7016404	Hashing	fake1	racist speech	158	1	1	[0.425905 0.574095]	1.522792101
44	PA, KNN	4703205	Hashing	fake1	a BBC article	351	0	1	[0.44609074 0.55390926]	0.979823112
45	PA, KNN	4703205	Hashing	fake1	a Guardian article	207	0	1	[0.41197865 0.58802135]	1.041075468
46	PA, KNN	4703205	Hashing	fake1	an Onion article	173	1	1	[0.40334928 0.59665072]	1.126484394
47	PA, KNN	4703205	Hashing	fake1	a Clickhole article	223	1	1	[0.39058035 0.60941965]	1.267370462
48	PA, KNN	4703205	Hashing	fake1	an angry copypasta	1753	1	1	[0.42880265 0.57119735]	1.135686159
49	PA, KNN	4703205	Hashing	fake1	racist speech	158	1	1	[0.43468478 0.56531522]	0.894541025
50	SVM, PA, LR	6986471	Hashing	fake1	a BBC article	351	0	1	[0.45327498 0.54672502]	1.365042448
51	SVM, PA, LR	6986471	Hashing	fake1	a Guardian article	207	0	1	[0.41662154 0.58337846]	1.761568785
52	SVM, PA, LR	6986471	Hashing	fake1	an Onion article	173	1	1	[0.41518526 0.58481474]	1.795326948
53	SVM, PA, LR	6986471	Hashing	fake1	a Clickhole article	223	1	1	[0.40378919 0.59621081]	1.200579166
54	SVM, PA, LR	6986471	Hashing	fake1	an angry copypasta	1753	1	1	[0.43228162 0.56771838]	1.273168325
55	SVM, PA, LR	6986471	Hashing	fake1	racist speech	158	1	1	[0.42265453 0.57734547]	1.034662247
56	Naive Bayes	1168238	TF-IDF	fake1	a BBC article	351	0	1	[0.1424688 0.8575312]	0.006030083
57	Naive Bayes	1168238	TF-IDF	fake1	a Guardian article	207	0	1	[0.13420037 0.86579963]	0.002504587
58	Naive Bayes	1168238	TF-IDF	fake1	an Onion article	173	1	1	[0.15205969 0.84794031]	0.001999655

Рис. 4.1. Перші 57 рядків таблиці результатів

Таблиця результатів містить 324 рядки та 10 стовпців. Для оцінки

точності використовуються 3 стовпці: required prediction, predicted та values, а також окремо вираховується кількість правильних передбачень за стовпцями required/predicted.

Значення у масиві values відповідають вірогідності індексу передбачення. Значення індексів передбачення наведені у required prediction та predicted, де required prediction – ідеальне та правильне передбачення, а predicted – отриманий результат передбачення.

Оскільки для кожного класифікатора в результаті тренування отримується сукупно 36 різних результатів точності передбачення та правильності, для загальної оцінки класифікатора використовується середнє значення усіх результатів точності та правильності.

4.2. Результат роботи натренованих класифікаторів

Для векторизації TF-IDF класифікатор опорних векторів SVM виділяється найвищою середньою точністю 95,74%. Це вказує на те, що SVM досяг найвищої частки правильних прогнозів порівняно з іншими класифікаторами під час навчання на даних, перетворених TF-IDF. Крім того, класифікатор SVM також досяг найвищого середнього правильного прогнозу 4,67, що свідчить про його надійність у точній категоризації текстових даних новин. З іншого боку, при використанні векторизації хешування класифікатор SVM зберіг свою продуктивність, досягнувши найвищої середньої точності 97,26% і найвищого середнього правильного передбачення 4,67. Результати демонструють ефективність SVM у обробці простору ознак великої розмірності, створеного методом векторизації хешування.

Ансамблеві методи, зокрема, комбінація SVM, PA та LR, показали високу продуктивність, особливо з векторизацією хешування. Цей ансамбль досяг середньої точності 96,93% і середнього правильного прогнозу 4.67, ще більше підкреслюючи переваги поєднання кількох стратегій класифікатора для підвищення точності класифікації (табл. 1).

Вивчаючи масштабованість ансамблевих моделей порівняно з індивідуальними класифікаторами, ми можемо отримати уявлення з наданих даних CSV. Розглянуті ансамблеві класифікатори, а саме "NB, LR, KNN", "PA, KNN" та "SVM, PA, LR", об'єднують різні базові класифікатори для підвищення точності прогнозування. Дивлячись на дані, ансамблеві моделі, як правило, демонструють довший час виконання порівняно з окремими класифікаторами, такими як Naive Bayes, логістичної регресії, опорних векторів та найближчих сусідів. Це пов'язано з тим, що ансамблеві моделі об'єднують декілька базових класифікаторів, що збільшує обчислювальні витрати.

Таблиця 1

Порівняння результатів класифікації

Класифікатор	Середня точність	Середня кількість вдалих передбачень	Середня швидкість (в секундах)
Наївний Баєсів (NB)	93.23%	4	0.0148
Пасивно-агресивний (PA)	94.8%	4.67	0.002
Лінійний опорних векторів (SVM)	95.58%	4.67	0.0056
Логістична регресія (LR)	94.27%	4.5	0.0018
k-найближчих сусідів (KNN)	90.79%	4.83	0.0406
Випадкового лісу (FOREST)	86.74%	4	0.2868
NB, KNN, LR	94.54%	4.33	0.6833
KNN, PA	78.72%	4	0.4743
SVM, PA, LR	94.98%	4.83	0.716

Вибір методів векторизації, таких як хешування або TF-IDF, впливає на масштабованість системи. TF-IDF вимагає більших обчислювальних ресурсів, оскільки включає в себе складні обчислення для визначення частоти термінів і зворотної частоти документів. Довжина текстових зразків також впливає на швидкість обробки даних різними класифікаторами, зокрема ансамблевими моделями. Обробка довших текстів вимагає більшої обчислювальної потужності для вилучення ознак, підбору моделей і прогнозування. Однак відносна масштабованість ансамблевих моделей порівняно з іншими класифікаторами може змінюватися залежно від характеру даних і складності ансамблю.

Хоча ансамблеві моделі можуть дещо подовжувати час виконання порівняно з окремими класифікаторами, вони все одно зберігають ефективну масштабованість. Це особливо очевидно при обробці великих наборів даних або складних завдань класифікації. Ансамблеві методи використовують різноманітність базових класифікаторів для підвищення надійності та продуктивності узагальнення, що потенційно компенсує незначне збільшення часу виконання.

Коли справа доходить до керування файлами моделей на серверах, ансамблеві класифікатори можуть становити проблему через більший розмір файлів порівняно з індивідуальними класифікаторами. Це може вплинути на вимоги до сховища і пропускну здатність мережі при передачі або завантаженні моделей в пам'ять для прогнозування. Однак, підвищена складність моделей ансамблевих класифікаторів може бути перевагою в сценаріях, де висока точність прогнозування має вирішальне значення.

З точки зору точності прогнозування, ансамблеві класифікатори, як правило, демонструють конкурентоспроможну продуктивність, часто перевершуючи індивідуальні класифікатори. Ансамблевий підхід використовує різноманітність базових класифікаторів, щоб пом'якшити недоліки будь-якої окремої моделі та підвищити загальну надійність прогнозування. Це видно з наведених даних, де ансамблеві класифікатори

послідовно досягають високої точності на різних типах текстових зразків, включаючи мову ворожнечі, неправдиві новини та сатиру.

Отже, хоча ансамблеві класифікатори можуть вимагати більше обчислювальних ресурсів і створювати проблеми в управлінні файлами моделей, їхня підвищена точність прогнозування і надійність роблять їх практичним вибором у сценаріях, де висока точність має першочергове значення. Важливо враховувати компроміси між масштабованістю, складністю моделі та точністю прогнозування при прийнятті рішення про використання ансамблевих класифікаторів у практичному застосуванні.

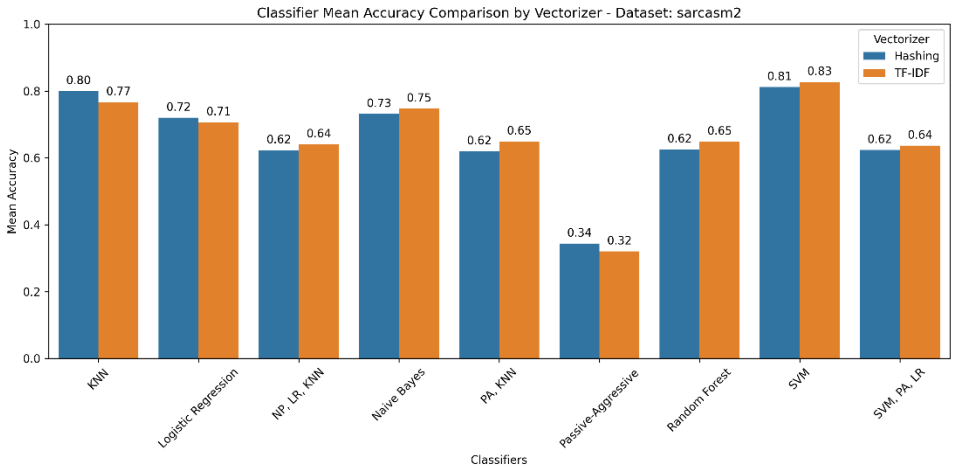


Рис. 4.2. Порівняння точності класифікаторів за категорією сарказму

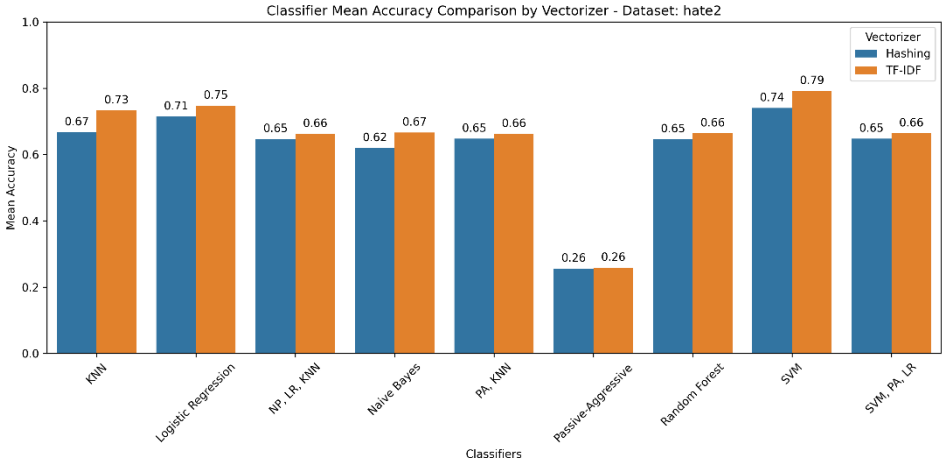


Рис. 4.3. Порівняння точності класифікаторів за категорією мови ворожнечі

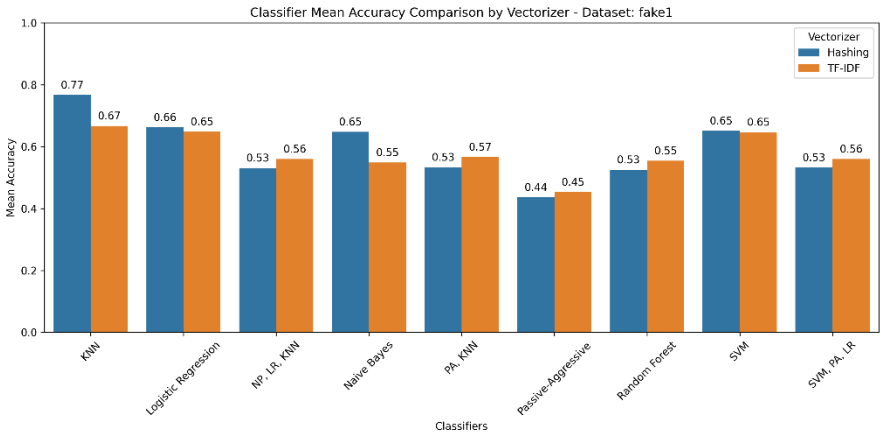


Рис. 4.4. Порівняння точності за категорією неправдивості

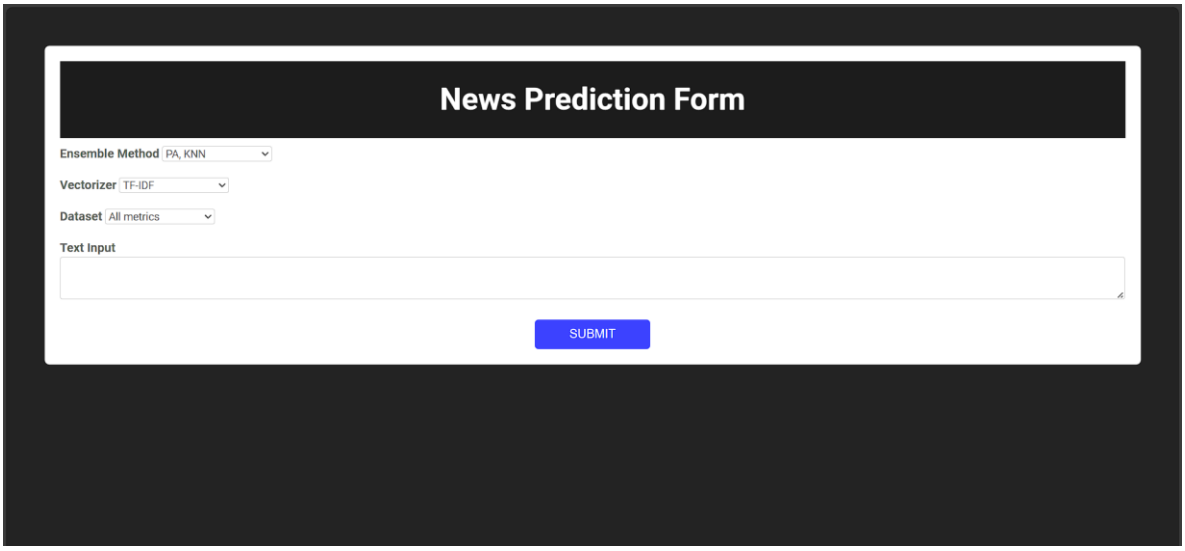


Рис. 4.5. Інтерфейс розробленого застосунку

Рис. 4.6. Вибір ансамблевого методу класифікації

Рис. 4.7. Вибір векторизатора

Рис. 4.8. Вибір категорії новин

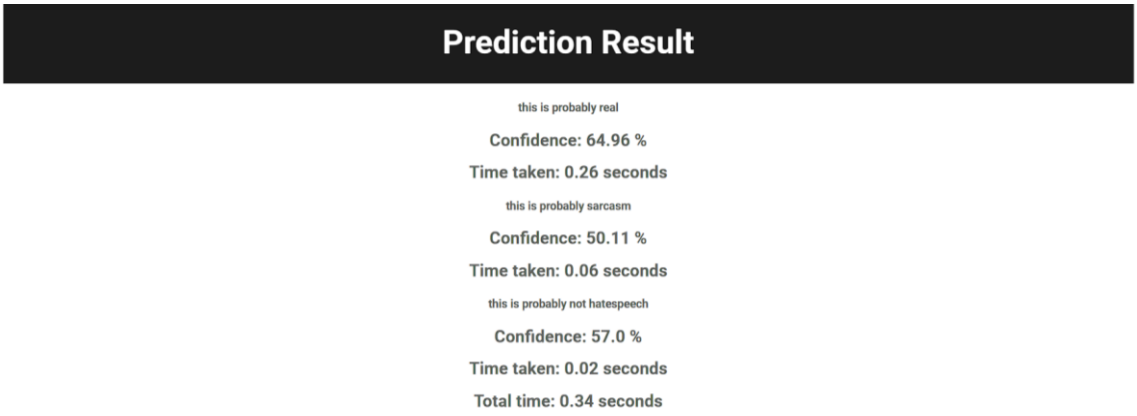


Рис. 4.9. Результат передбачення

Запропонований програмний метод дозволяє аналізувати різні категорії новин з різною структурою тексту та способом подачі інформації (недостовірні новини, сатира, мова ворожнечі тощо) та виконувати перевірку достовірності новинних текстів на основі аналізу різними методами класифікації, у тому числі ансамблевими методами.

4.3. Висновки до розділу 4

У цьому розділі розглянуто використання ансамблевих методів класифікації недостовірних текстів для підвищення точності прогнозування та оцінено різні моделі класифікації тексту за допомогою окремих наборів даних. Оцінено наївний байєсівський, пасивно-агресивний класифікатори, метод опорних векторів, логістичну регресію, метод k-найближчих сусідів та класифікатори випадкового лісу. Для подальших досліджень можна розширити аналіз застосування ансамблевих методів класифікації на різноманітніші типи даних та ситуації, що можуть виникнути у контексті виявлення недостовірних текстів. Також можна дослідити вплив використання різних комбінацій класифікаторів та алгоритмів на точність та надійність прогнозування, а також ефективність цих методів у різних областях, таких як соціальні мережі, новинні портали чи форуми.

Дослідження вирізняється своїм новим підходом до вирішення складних проблем, пов'язаних із виявленням неправдивих новин, сатиричних новин та мови ворожнечі. У той час як попередні дослідження зосереджувалися переважно на окремих типах дезінформації, це дослідження має цілісний погляд, визнаючи необхідність розробки єдиної системи, здатної виявляти різні форми оманливого контенту.

На відміну від існуючих досліджень, які переважно використовували ансамблеві методи, такі як Naive Bayes, логістична регресія та SVM, це дослідження вивчає нові комбінації, зокрема NB, LR, KNN; PA, KNN; та SVM, PA, LR. Урізноманітнюючи набір класифікаторів, дослідження має на меті визначити «найменший спільний знаменник» – класифікатор, який демонструє надійну роботу з різними типами дезінформації.

Включення наборів даних сатиричних новин і мови ворожнечі вносить новий вимір у дослідницький ландшафт. Сатиричні новини з їхнім навмисним використанням гумору та іронії створюють особливі проблеми для виявлення, тоді як дані про мову ворожнечі, що характеризуються природною мовою в коментарях, представляють унікальні лінгвістичні

патерни. Окремо аналізуючи кожен тип дезінформації, це дослідження прагне виявити нюанси і відповідно вдосконалити модель виявлення.

Дослідження вивчає потенціал комбінацій ансамблевих методів для підвищення точності виявлення без значних компромісів. Використовуючи сильні сторони декількох класифікаторів та оптимізуючи їхню взаємодію, дослідження має на меті досягти чудових результатів у виявленні оманливого контенту в різних контекстах.

На відміну від попередніх робіт, які зосереджувалися переважно на конкретних аспектах виявлення недостовірних новин, це дослідження пропонує комплексний і нюансований підхід. Вивчаючи ефективність різних комбінацій класифікаторів і поширюючи аналіз на сатиричні новини та мову ворожнечі, дослідження збагачує наше розуміння складнощів, притаманних виявленню дезінформації.

Інтеграція штучного інтелекту в процес виявлення неправдивих новин за допомогою ансамблевих методів машинного навчання відкриває багатообіцяючі перспективи та можливості для підвищення довіри до ЗМІ та цілісності інформації. Використовуючи AI, розробники можуть створювати більш складні та адаптивні системи, здатні аналізувати великі масиви даних для виявлення закономірностей і аномалій, що вказують на дезінформацію. Використання ансамблевих методів, які об'єднують кілька моделей машинного навчання, може значно підвищити точність виявлення, використовуючи сильні сторони різних алгоритмів і пом'якшуючи їхні слабкі сторони. Такий підхід не лише підвищує надійність виявлення фейкових новин, але й адаптується до еволюції тактик дезінформації. Крім того, здатність штучного інтелекту безперервно навчатися на нових даних дає можливість цим системам випереджати швидкозмінний ландшафт цифрової інформації.

ВИСНОВКИ

У даній магістерській дисертації поставлено за мету розроблення програмного методу для аналізу текстів новин з метою виявлення недостовірної інформації, мови ворожнечі та сатири за допомогою алгоритмів машинного навчання. Результатом досліджень має бути розробка алгоритму для швидкого аналізу новин за окремими категоріями недостовірності, а також програмного продукту, який дозволить взаємодіяти з алгоритмом.

У першому розділі розглянуто комерційні проєкти та наукові підходи до аналізу текстів новин, які включають тематично-агностичний, мовний, експертний та «бібліотечний» підходи, а також використання краудсорсингу та експертів.

У другому розділі проведено порівняльний аналіз алгоритмів машинного навчання, що включають наївний баєсівський, пасивно-агресивний, SVM, логістичну регресію, KNN та класифікатори випадкового лісу. Обрано використання ансамблевих методів з методом голосування, що поєднує в собі кілька алгоритмів для досягнення кращої точності та надійності результатів.

У третьому розділі описані використані технології для розробки системи, зокрема Flask та Scikit-learn. Окреслено архітектуру системи, що включає вебсервер з API для взаємодії з клієнтською частиною, а також використання бібліотек Keras та Scikit-learn для тренування та застосування моделі машинного навчання.

У четвертому розділі проведено аналіз використання ансамблевих методів класифікації недостовірних текстів для підвищення точності прогнозування. Проаналізовано різні моделі класифікації тексту та визначено напрямки подальших досліджень для оптимізації використання ансамблевих методів у боротьбі з поширенням недостовірної інформації в мережі Інтернет.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Burkhardt JM. (2017) History of fake news. Libr. Technol. Rep. 2017;53(8):37.
2. N. F. Baarir and A. Djeffal (2021). "Fake News detection Using Machine Learning," 2020 2nd International Workshop on Human-Centric Smart Environments for Health and Well-being (IHSH), Boumerdes, Algeria, 2021, pp. 125-130, doi: 10.1109/IHSH51661.2021.9378748.
3. Chen Y, Conroy Nadia K., Rubin Victoria L. (2015). News in an online world: The need for an “automatic crap detector” Proceedings of the Association for Information Science and Technology. 2015;52(1):1–4.
4. Stahl, K. (2018). Fake news detection in social media. California State University Stanislaus, 6, pp. 4-15.
5. Castelo, S., Almeida, T., Elghafari, A., Santos, A., Pham, K., Nakamura, E., Freire, J.: (2019) A topic-agnostic approach for identifying fake news pages. In: Companion Proceedings of the 2019 World Wide Web Conference on – WWW 2019, pp. 975–980 (2019). 10.1145/3308560.3316739.
6. Ahmed, S., Hinkelmann, K., Corradini, F. (2019). Combining machine learning with knowledge engineering to detect fake news in social networks – a survey. In: Proceedings of the AAAI 2019 Spring Symposium, vol. 12.
7. Vlachos, A., Riedel, S.: (2014) Fact checking: task definition and dataset construction. In: Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science, Baltimore, MD, USA, pp. 18–22. Association for Computational Linguistics (2014). 10.3115/v1/W14-2508.
8. Hassan, N., Arslan, F., Li, C., Tremayne, M. (2017). Toward automated fact-checking: detecting check-worthy factual claims by claimbuster. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining – KDD 2017, Halifax, NS, Canada, pp. 1803–1812. ACM Press (2017). 10.1145/3097983.3098131.

9. ClaimBuster [Електронний ресурс]. — Режим доступу: https://idir.uta.edu/claimbuster/
10. Pennycook G, Rand DG.(2019). Fighting misinformation on social media using crowdsourced judgments of news source quality. Proc. Natl. Acad. Sci. 2019;116(7):2521–2526. doi: 10.1073/pnas.1806781116.
11. Ruchansky, N., Seo, S., Liu, Y. (2017). CSI: a hybrid deep model for fake news detection. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, pp. 797–806.
12. Okoro EM, Abara BA, Umagba AO, Ajonye AA, Isa ZS (2018). A hybrid approach to fake news detection on social media. Niger. J. Technol. 2018;37(2):454.
13. Meedan Check [Електронний ресурс]. — Режим доступу: https://meedan.com/check
14. Logically [Електронний ресурс]. — Режим доступу: https://www.logically.ai/
15. Full Fact Електронний ресурс]. — Режим доступу: https://fullfact.org/
16. Fabula AI. Wikipedia (2019). [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Fabula_AI
17. Grover [Електронний ресурс]. — Режим доступу: https://grover.allenai.org/
18. Adverif.ai [Електронний ресурс]. — Режим доступу: https://adverifai.com/about/
19. Alto Intelligence [Електронний ресурс]. — Режим доступу: https://www.altointelligence.com/
20. Blackbird AI [Електронний ресурс]. — Режим доступу: https://blackbird.ai/
21. Defudger [Електронний ресурс]. — Режим доступу: https://thehub.io/startups/defudger
22. Snopes [Електронний ресурс]. — Режим доступу: https://www.snopes.com/

23. CaptainFact [Електронний ресурс]. — Режим доступу: https://captainfact.io/ https://reporterslab.org/duke-students-tackle-big-challenges-in-automated-fact-checking/
24. MBFC [Електронний ресурс]. — Режим доступу: https://mediabiasfactcheck.com/mbfcs-data-api/
25. Hamilton Dashboard [Електронний ресурс]. — Режим доступу: https://securingdemocracy.gmfus.org/hamilton-dashboard/
26. IffyQuotient [Електронний ресурс]. — Режим доступу: https://csmr.umich.edu/projects/iffy-quotient/
27. NewsWhip [Електронний ресурс]. — Режим доступу: https://www.newswhip.com/
28. Politifact [Електронний ресурс]. — Режим доступу: https://www.politifact.com/
29. Polygraph [Електронний ресурс]. — Режим доступу: https://www.polygraph.info/
30. FactCheck.org [Електронний ресурс]. — Режим доступу: https://www.factcheck.org/
31. Shalini Pandey, Sankeerthi Prabhakaran, N. V. Subba Reddy, Dinesh Acharya (2021). Fake News Detection from Online media using Machine Learning Classifiers, in Journal of Physics: Conference Series, 1st International Conference on Artificial Intelligence, Computational Electronics and Communication System (AICECS 2021) 28-30 October 2021, vol. 2161, no. 27, pp 2-9. DOI 10.1088/1742-6596/2161/1/012027.
32. Vasu Agarwal, H. Parveen Sultana, Srijan Malhotra, Amitrajit Sarkar (2019). Analysis of classifiers for fake news detection, Procedia Comput. Sci., 165 (2019), pp. 377-383, DOI: 10.1016/j.procs.2020.01.035.
33. Chary Deekshith P., Singh R.P. (2020). Review on Advanced Machine Learning Model: Scikit-Learn (July 4, 2020), International Journal of Scientific Research and Engineering Development (IJSRED) Vol.3, Issue 4, 526-529.

34. Dietterich T.G. (2000). Ensemble Methods in Machine Learning. In: Multiple Classifier Systems. MCS 2000. Lecture Notes in Computer Science, vol 1857. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45014-9_1.
35. Urszula Krzeszewska, Aneta Poniszewska-Maranda, Joanna Ochelska-Mierzejewska (2022). Systematic Comparison of Vectorization Methods in Classification Context. Applied Sciences. 12. 5119. DOI: 10.3390/app12105119.
36. Shu K., Sliva A., Wang S., Tang J., & Liu H. (2017). Fake News Detection on Social Media: A Data Mining Perspective. ACM SIGKDD Explorations Newsletter, 19(1), 22-36. DOI: 10.1145/3137597.3137600.
37. Wang W., Cui P., Zhu W., & Yang S. (2018). Fake News Detection with Deep Diffusive Neural Network. Proceedings of the 2018 World Wide Web Conference on World Wide Web (pp. 797-806).
38. Rubin V. L., Conroy N. J., & Chen Y. (2015). Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News. Proceedings of the Association for Information Science and Technology, 52(1), 1-4. DOI: 10.18653/v1/W16-0802.
39. Reis J. C., Correia A., Murai F., Veloso A., Benevenuto F., & Cambria E. (2019). Supervised Learning for Fake News Detection.IEEE Intelligent Systems, 34(2), 76-81. DOI: 10.1109/MIS.2019.2899143.
40. Pal S., Kumar T. S., & Pal S. (2019). Applying Machine Learning to Detect Fake News. Indian Journal of Computer Science, 4(1), 7-12. DOI: 10.17010/ijcs/2019/v4/i1/142411.
41. Ogunsuyi Opeyemi J., Adebola K. OJO. (2022). K-Nearest Neighbors Bayes Approach to False News Detection from Text on Social Media. International Journal of Education and Management Engineering (IJEME), Vol.12, No.4, pp. 22-32. DOI:10.5815/ijeme.2022.04.03.

42. Marina Azer, Mohamed Taha, Hala H. Zayed, Mahmoud Gadallah. (2021). Credibility Detection on Twitter News Using Machine Learning Approach. International Journal of Intelligent Systems and Applications (IJISA), Vol.13, No.3, pp.1-10. DOI:10.5815/ijisa.2021.03.01.
43. Prashengit Dhar, Md. Zainal Abedin. (2021). Bengali News Headline Categorization Using Optimized Machine Learning Pipeline. International Journal of Information Engineering and Electronic Business (IJIEEB), Vol.13, No.1, pp. 15-24. DOI:10.5815/ijieeb.2021.01.02.
44. Bodunde Akinyemi, Oluwakemi Adewusi, Adedoyin Oyebade. (2020). An Improved Classification Model for Fake News Detection in Social Media. International Journal of Information Technology and Computer Science (IJITCS), Vol.12, No.1, pp.34-43. DOI:10.5815/ijitcs.2020.01.05.
45. Shubham Bauskar, Vijay Badole, Prajal Jain, Meenu Chawla. (2019). Natural Language Processing based Hybrid Model for Detecting Fake News Using Content-Based Features and Social Features. International Journal of Information Engineering and Electronic Business (IJIEEB), Vol.11, No.4, pp. 1-10. DOI:10.5815/ijieeb.2019.04.01.
46. Yuan L., Jiang H., Shen H., Shi L., Cheng N. (2023). Sustainable Development of Information Dissemination: A Review of Current Fake News Detection Research and Practice. Systems. 11(9):458. https://doi.org/10.3390/systems11090458.
47. Zhiying Jiang, Matthew Y.R. Yang, Mikhail Tsirlin, Raphael Tang, Yiqin Dai, Jimmy Lin (2023). “Low-Resource” Text Classification: A Parameter-Free Classification Method with Compressors
48. Iftikhar Ahmad, Muhammad Yousaf, Suhail Yousaf, Muhammad Ovais Ahmad, "Fake News Detection Using Machine Learning Ensemble Methods", Complexity, vol. 2020, Article ID 8885861, 11 pages, 2020.

49. Kasra Majbouri Yazdi, Adel Majbouri Yazdi, Saeid Khodayi, Jingyu Hou, Wanlei Zhou, & Saeed Saedy. (2020). Improving Fake News Detection Using K-means and Support Vector Machine Approaches. International Journal of Electrical, Electronic and Communication Sciences, 13.0(2).
50. R.J. Poovaraghan, M.V. Keerti Priya, P.V. Sai Surya Vamsi, Mansi Mewara, Sowmya Loganathan (2019). Fake News Accuracy using Naive Bayes Classifier
51. P.O. Abas Sunarya, Rina Refianti, Achmad Benny Mutiara, Wiranti Octaviani. (2019). Comparison of Accuracy between Convolutional Neural Networks and Naïve Bayes Classifiers in Sentiment Analysis on Twitter
52. Crammer K., Dekel O., Keshet J., Shalev-Shwartz S., Singer Y.. (2006) Online Passive-Aggressive Algorithms, Journal of Machine Learning Research 7 (2006) 551–585.
53. Ihsan A., Mohamad N. B. A., Palaiahnakote S., Nurul F. B. M. N., (2022). Fake News Detection Techniques on Social Media: A Survey
54. Reshmi T S, Daniel Madan Raja S, Priya J (2021). Fake News Detection Using Source Information and Bayes Classifier
55. Z Khanam, B N Alwasel, H Sirafi, M Rashid (2021). Fake News Detection Using Machine Learning Approaches
56. Berrondo-Otermin M., Sarasa-Cabezuelo A. (2023). Application of Artificial Intelligence Techniques to Detect Fake News: A Review. Electronics. 12 (24): 5041. https://doi.org/10.3390/electronics12245041.
57. Akhtar P., Ghouri A.M., Khan H.U.R. et al. (2023). Detecting fake news and disinformation using artificial intelligence and machine learning to avoid supply chain disruptions. Ann Oper Res 327, 633–657. https://doi.org/10.1007/s10479-022-05015-5.
58. What are Naïve Bayes classifiers? IBM. [Електронний ресурс]. — Режим доступу:. https://www.ibm.com/topics/naive-bayes

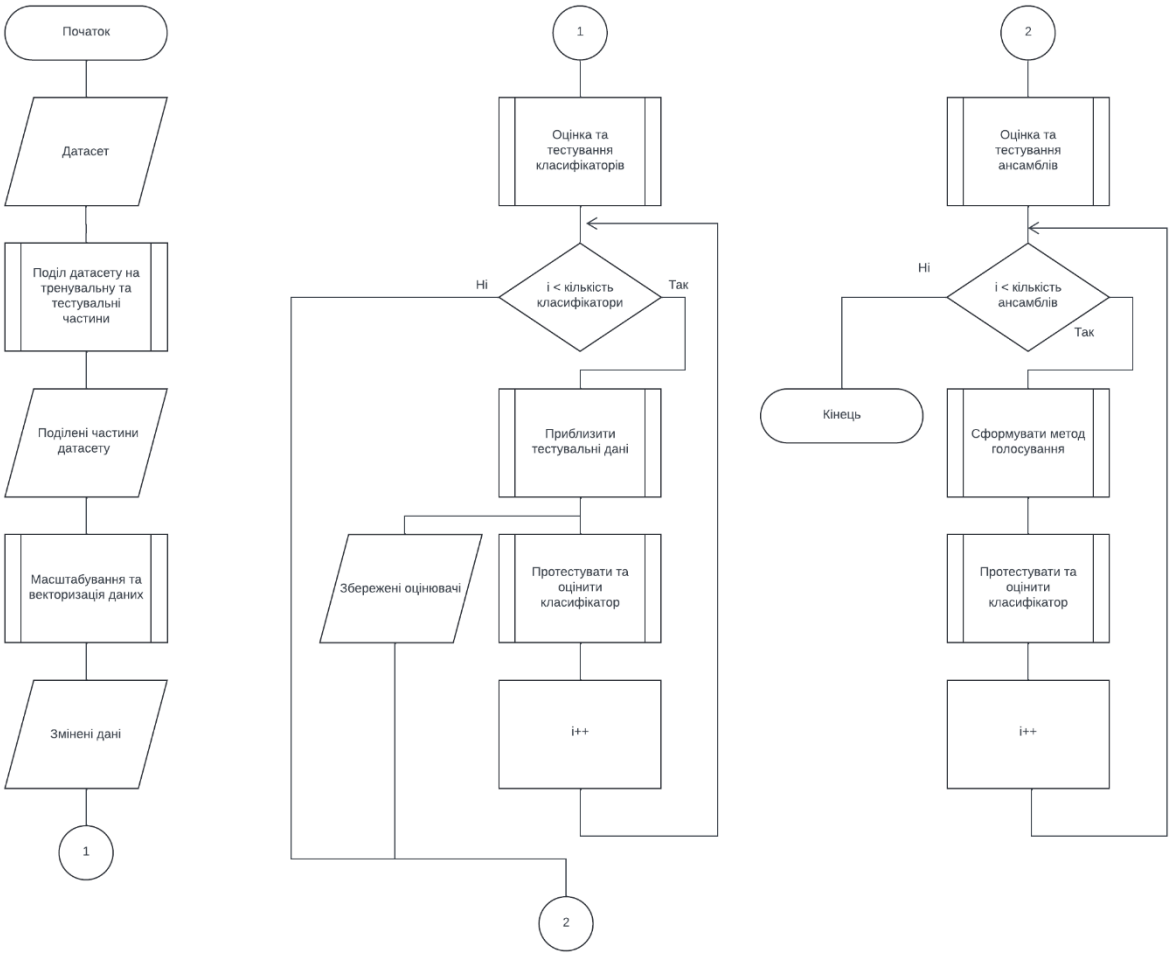
59. A. Y. Chervonenkis. “Early history of support vector machines,” in Empirical Inference, B. Schölkopf, Z. Luo, and V. Vovk, Eds. Berlin: Springer, 2013, pp. 13–20.
60. Hosmer, David W.; Lemeshow, Stanley (2000). Applied Logistic Regression (2nd ed.). Wiley. ISBN 978-0-471-35632-5.
61. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE transactions on information theory, 13 (1), 21-27.
62. Ho, Tin Kam (1995). Random Decision Forests. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.
63. Rokach, L. (2010). "Ensemble-based classifiers". Artificial Intelligence Review. 33 (1–2): 1–39.
64. Kyriakides, G., & Margaritis, K. G. (2019). Hands-On Ensemble Learning with Python: Build highly optimized ensemble machine learning models using scikit-learn and Keras. Packt Publishing Ltd.Serrano, L. (2021). Grokking Machine Learning. Simon and Schuster.
65. Rajaraman, A.; Ullman, J.D. (2011). "Data Mining" (PDF). Mining of Massive Datasets. pp. 1–17.
66. MurmurHash1. GitHub [Електронний ресурс]. — Режим доступу: https://github.com/aappleby/smhasher/wiki/MurmurHash1
67. MurMurHash3, an ultra fast hash algorithm for C#/.NET. Adam Horvath’s Blog [Електронний ресурс]. — Режим доступу: https://blog.teamleadnet.com/2012/08/murmurhash3-ultra-fast-hash-algorithm.html
68. Stanford University’s Course — CS231n: Convolutional Neural Network for Visual Recognition by Prof. Fei-Fei Li, Justin Johnson, Serena Yeungю
69. What are convolutional neural networks? IBM (2021). [Електронний ресурс]. — Режим доступу: https://www.ibm.com/topics/convolutional-neural-networks.

70. Kaggle. [Електронний ресурс]. — Режим доступу: https://www.kaggle.com/
71. Jillani Soft Tech. (2022). Fake or Real News. Kaggle. [Електронний ресурс]. — Режим доступу: https://www.kaggle.com/datasets/jillanisofttech/fake-or-real-news
72. Bijoy Bose. (2019). BBC News Classification. Kaggle. [Електронний ресурс]. — Режим доступу: https://kaggle.com/competitions/learn-ai-bbc
73. Undefinenull (2022). Satirical News from The Onion. Kaggle. [Електронний ресурс]. — Режим доступу: https://www.kaggle.com/datasets/undefinenull/satirical-news-from-the-onion
74. Andrii Samoshyn. (2020). Hate Speech and Offensive Language Dataset. Kaggle. [Електронний ресурс]. — Режим доступу: https://wsww.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset/data

ДОДАТКИ

Додаток 1
Копії графічних матеріалів

Блок-схема запропонованого методу аналізу текстів новин на неправдивість

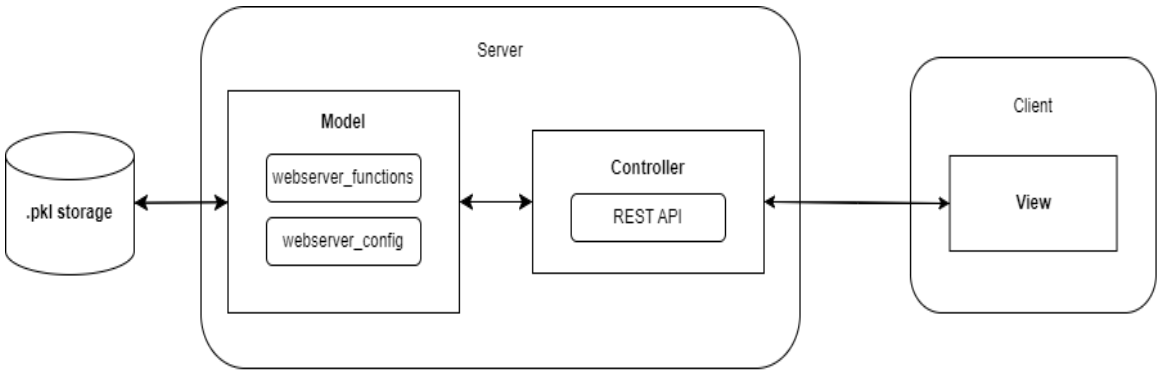


Олексій Мельничук
група КП-21мн

Підготовка даних для аналізу

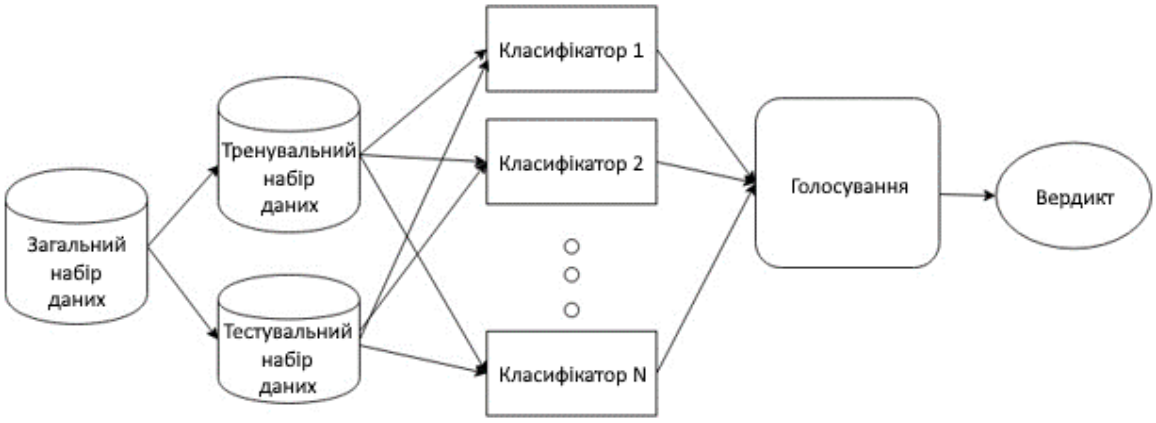
Олексій Мельничук
група КП-21мн

Архітектура програмного забезпечення



Олексій Мельничук
група КП-21мн

Використання ансамблевого методу для виявлення неправдивого тексту



Олексій Мельничук
група КП-21мн

Додаток 2
Фрагменти коду запропонованого програмного методу аналізу текстів новин на неправдивість

vectors.py

```
from metrics import get_metrics
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import HashingVectorizer

vectorizers = {
  "Count": CountVectorizer(analyzer = "word", stop_words='english'),
  "TF-IDF":  TfidfVectorizer(stop_words='english', max_df=0.7),
  "Hashing":                    HashingVectorizer(stop_words='english',
alternate_sign=False)
}

def get_vectorizer(name):
  return vectorizers[name]

def template_vectorization(vectorizer, x_train, x_test):
  vectorizer = get_vectorizer(vectorizer)
  train = vectorizer.fit_transform(x_train)
  test = vectorizer.transform(x_test)
  return [vectorizer, train, test]

def get_vectorized(vectorizer, sample):
  test = vectorizer.transform(sample)
  return test
```

predict.py

```
import pandas as pd
from sklearn.model_selection import train_test_split
import sklearn

from sklearn import metrics
# from pandas_ml import ConfusionMatrix
from matplotlib import pyplot as plt

import itertools
import numpy as np

from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)

from classifiers import get_tokens, get_model, test_model, classifiers,
estimators, ensemble
from vectors import template_vectorization, get_vectorized
from config import get_dataset, testdict
from preprocess import stemming

from pathlib import Path

from import_export import import_pickle, export_pickle

from time import time
```

```
   def prepare_dataframe(dataset):
     df = pd.read_csv(dataset["path"])


     print("Stemming...")
     print(dataset["column_text"])
     print(df[dataset["column_text"]].head())

     df[dataset["column_text"]].head()


     df = df.set_index("Unnamed: 0")

     # Separate the labels and set up training and test datasets
     y = df[dataset["column_label"]]

     # Drop the `label` column
     df.drop(dataset["column_label"], axis=1)      #where numbering of news
article is done that column is dropped in dataset
     X_train, X_test, y_train, y_test = get_sets(df, dataset["column_text"],
y)
     return [X_train, X_test, y_train, y_test]


   def get_sets(df, column_text, y):
     # Make training and test sets
     # test set is 33% of all
     X_train,          X_test,          y_train,          y_test          =
train_test_split(df[column_text].values.astype('U'),   y,   test_size=0.33,
random_state=53)
     return [X_train, X_test, y_train, y_test]


   def predict(input, clf):
     return [clf.predict(input), clf.predict_proba(input)]

   ### reading the test files to be fed into predictions
   testsarr = list(testdict.keys())
   for testobj in testsarr:
     testdict[testobj]["data"]                                              =
[Path(testdict[testobj]["path"]).read_text(encoding="utf8")]

   vectorizers = ["TF-IDF", "Hashing"]
   datasets = ['fake1', 'sarcasm2', 'hate2']

   classarr = list(classifiers.keys())

   result_dictionary = {}
   fieldnames = ["Accuracy"]
   for testobj in testsarr:
     for idk in ["Expected: ", "Predicted: "]:
       fieldnames.append(idk + testdict[testobj]["desc"])

   selected_ensembles = {
     "NP, LR, KNN": {
       "combo": ["Naive Bayes", "Logistic Regression", "KNN"],
       "trained": []
```

```
        },
      "PA, KNN": {
        "combo": ["Passive-Aggressive", "KNN"],
        "trained": []
        },
      "SVM, PA, LR":
        {
          "combo": ["SVM", "Passive-Aggressive", "Logistic Regression"],
        "trained": []
        }
    }

    def test_prediction(chosen_vec, textdata):
        print('ok')


    ### iterating through datasets for each type of prediction

    for dataset_name in datasets:

      result_dictionary[dataset_name] = {}

      print("----------------------Chosen dataset: " + dataset_name)

      dataset = get_dataset(dataset_name)

      X_train, X_test, y_train, y_test = prepare_dataframe(dataset)


      column_fake_vals = dataset["label_values"]
      label_meaning = dataset["label_meaning"]
      measure = dataset["measure"]


      print("Training...")

      for vec in vectorizers:
        vectorizer, vector_train, vector_test = template_vectorization(vec,
X_train, X_test)

        export_pickle(vectorizer, 'vec', dataset_name + '/' + vec + '/')

        print("-------------Chosen vectorizer: " + vec)

        for single_ens in list(selected_ensembles.keys()):
          selected_ensembles[single_ens]["trained"] = []

        # checking and training every classifier
        for classif_name in classarr:
          print("------Chosen classifier: " + classif_name)



          clf = get_model(classif_name, vector_train, y_train)


          # t0 = time()
          cm, score = test_model(clf, vector_test, y_test, column_fake_vals)
```

```
          # print('Time to classify: ')
          # print(time() - t0)

          export_pickle(clf, classif_name, dataset_name + '/' + vec + '/')

          # saving classifiers in array in memory is wasteful
          # for single_ens in list(selected_ensembles.keys()):
          #   if classif_name in selected_ensembles[single_ens]["combo"]:
          #
selected_ensembles[single_ens]["trained"].append((estimators[classif_name],
clf))


          result_dictionary[dataset_name][classif_name + ", " + vec] =
{"Accuracy": score}

          # testing trained classifiers on test input data
          print("\t\t\tExpected\tPredicted Probability")

          testing_success = True

          for testobj in testsarr:

            t0 = time()
            prediction, probability = predict(get_vectorized(vectorizer,
testdict[testobj]["data"]), clf)
            print('Time to classify: ')
            print(time() - t0)

            predicted_value = prediction[0]
            expected_value  = testdict[testobj]["expected_vals"][measure]

            print(testdict[testobj]["desc"] + "\t" + str(expected_value) +
"\t\t" + str(predicted_value) + "      " + str(probability[0]))
            if predicted_value != expected_value:
              testing_success = False

            result_dictionary[dataset_name][classif_name + ", " +
vec]["Expected: " + testdict[testobj]["desc"]] = expected_value
            result_dictionary[dataset_name][classif_name + ", " +
vec]["Predicted: " + testdict[testobj]["desc"]] = predicted_value

          if testing_success == True:
            print("All correct")
          else:
            print("Predictions failed")


        # ensemble methods
        for single_ens in list(selected_ensembles.keys()):
          print("------Custom " + single_ens)
          # ensemble method:
          print(selected_ensembles[single_ens]["trained"])

          # array of classifiers is taken from selected_ensembles instead of
being loaded form fs
```

```
        #                    trained_classifier_array                    =
selected_ensembles[single_ens]["trained"]

        trained_classifier_array = []
        for clf_name in selected_ensembles[single_ens]["combo"]:
          model = import_pickle(classif_name, dataset_name + '/' + vec +
'/')
          trained_classifier_array.append((estimators[clf_name], model))

        print(trained_classifier_array)

        clf_ensemble  =  ensemble(trained_classifier_array,  vector_train,
y_train)

        export_pickle(clf_ensemble, single_ens, dataset_name + '/' + vec +
'/')


        cm,   score   =   test_model(clf_ensemble,   vector_test,   y_test,
column_fake_vals)

        print("\t\t\tExpected\tPredicted Probability")

        testing_success = True

        for testobj in testsarr:

          t0 = time()
          prediction,  probability  =  predict(get_vectorized(vectorizer,
testdict[testobj]["data"]), clf_ensemble)
          print('Time to classify: ')
          print(time() - t0)

          predicted_value = prediction[0]
          expected_value  = testdict[testobj]["expected_vals"][measure]

          print(testdict[testobj]["desc"]  +  "\t"  +  str(expected_value)  +
"\t\t" + str(predicted_value) + "      " + str(probability[0]))
          if predicted_value != expected_value:
            testing_success = False

          result_dictionary[dataset_name][classif_name    +    ",    "    +
vec]["Expected: " + testdict[testobj]["desc"]] = expected_value
          result_dictionary[dataset_name][classif_name    +    ",    "    +
vec]["Predicted: " + testdict[testobj]["desc"]] = predicted_value

        if testing_success == True:
          print("All correct")
        else:
          print("Predictions failed")
```

classifiers.py

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
```

```
from sklearn import metrics
# from pandas_ml import ConfusionMatrix
from sklearn.linear_model import PassiveAggressiveClassifier
import numpy as np

from sklearn.linear_model import LogisticRegression

from sklearn.svm import LinearSVC
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier

from sklearn.calibration import CalibratedClassifierCV

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import LogisticRegression


# from k.functions import plot_confusion_matrix, draw_matrix,
most_informative_feature_for_binary_classification


from sklearn import svm

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import GridSearchCV


# Modified Passive aggresive classifier with predict_proba method
class PA_classifier_prob(PassiveAggressiveClassifier):
  def __init__(self,X,y):
    super().__init__(X,y)
  def predict_proba(self,X):
    arr1= 1-(1. / (1. + np.exp(-self.decision_function(X))))
    arr2= -(arr1-1)
    return np.stack((arr2, arr1), axis=1)

  #method to wrap values from decision function in a sigmoid function to
get probabilities

# class LinearSVC_prob(LinearSVC):
#   def __init__(self,X,y):
#     super().__init__(X,y)
#     self = CalibratedClassifierCV(self)
```

```
  classifiers = {
    "Naive Bayes": MultinomialNB(alpha=.01,),
    "Passive-Aggressive": PA_classifier_prob(max_iter=50),
    "SVM": CalibratedClassifierCV(LinearSVC(loss='hinge')),
    "Logistic Regression": LogisticRegression(max_iter = 1000),
    "KNN": KNeighborsClassifier(),

    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
  }



  more_classifiers = {

    # 'Naive Bayes':  GaussianNB(),
    # 'SVM':  SVC(),
    'Neural Network': MLPClassifier(max_iter = 1000),
    'Extra Trees':  ExtraTreesClassifier(),
    'AdaBoost':  AdaBoostClassifier(),
    'GradientBoosting': GradientBoostingClassifier()
    # 'XGB':  XGBClassifier(),
    # 'CatBoost':  CatBoostClassifier(logging_level='Silent')
  }


  estimators = {
    "Naive Bayes": 'mnb',
    "Passive-Aggressive": 'pa',
    # "SVM": LinearSVC_prob(),
    "SVM": 'svc',
    # "SVM": svm.SVC(kernel='linear', probability=True),
    "Logistic Regression": 'lr',
    "KNN": 'knn',
    "Decision Tree": 'dt',
    "Random Forest": 'rf',
  }

  more_estimators = {
    'Neural Network': '',
    'Extra Trees':  'ExTrees',
    'AdaBoost':  '',
    'GradientBoosting': '',
    'XGB':  '',
    'CatBoost':  'CatBoost'
  }


  # why not regressive classifiers?
  #   https://www.cluzters.ai/forums/topic/575/accuracy-score-value-error-
can-t-handle-mix-of-binary-and-contin?c=1597
  # because they do not measure accuracy


  def get_classifier(name):
    return classifiers[name]

  def get_model(name, x_train,  y_train):
```

```
    print(name)
    clf = get_classifier(name)
    clf.fit(x_train, y_train)
    return clf


  def test_model(clf, x_test, y_test, labels):
    pred = clf.predict(x_test)

    score = metrics.accuracy_score(y_test, pred)
    # n binary classification, this function is equal to the jaccard_score
function.
    print("accuracy:   %0.5f" % score)

    cm = metrics.confusion_matrix(y_test, pred, labels=labels)
    # plot_confusion_matrix(cm, classes=labels)
    print(cm)
    return [cm, score]



  def get_tokens(classifier, tfidf_vectorizer):
    clf = get_classifier(classifier)
    print('Tokens: ' + classifier)

    feature_names = tfidf_vectorizer.get_feature_names_out()

    ### Most real
    # feature_log_prob_ instead of coef_
    sorted(zip(clf.feature_log_prob_[0], feature_names), reverse=True)[:20]

    ### Most fake
    sorted(zip(clf.feature_log_prob_[0],              feature_names))[:20]
# clearly there are certain words which might show political intent and source
in the top fake features (such as the words corporate and establishment).

    tokens_with_weights          =          sorted(list(zip(feature_names,
clf.feature_log_prob_[0])))
    print(tokens_with_weights)

  # ensembling
  # we are using voting classifiers

  from sklearn.ensemble import VotingClassifier

  # def get_estimators(array_classifiers):

  def ensemble(estimators, x_train, y_train):
    ensemble = VotingClassifier(estimators, voting='soft')
    ensemble.fit(x_train, y_train)
    return ensemble
```

import_export.py

```
  import pickle
  import os
```

```
from config import model_file_path

def export_pickle(model, name, subpath):
    model_pkl_file = model_file_path + subpath + name + '.pkl'

    os.makedirs(os.path.dirname(model_file_path + subpath),
exist_ok=True)

    with open(model_pkl_file, 'wb') as file:
        pickle.dump(model, file)

def import_pickle(name, subpath):
    model_pkl_file = model_file_path + subpath + name + '.pkl'
    filesize = os.path.getsize(model_pkl_file)

    print(model_pkl_file)
    with open(model_pkl_file, 'rb') as file:
        model = pickle.load(file)

    return model, filesize
```

webserver_functions.py

```
import pandas as pd
from sklearn.model_selection import train_test_split

from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)


from classifiers import get_tokens, get_model, test_model, classifiers,
estimators, ensemble
from vectors import template_vectorization, get_vectorized
from config import get_dataset, testdict
from preprocess import stemming

from pathlib import Path

from import_export import import_pickle, export_pickle


def prepare_dataframe(dataset):
  df = pd.read_csv(dataset["path"])


  print("Stemming...")
  print(dataset["column_text"])
  print(df[dataset["column_text"]].head())

  df[dataset["column_text"]].head()


  df = df.set_index("Unnamed: 0")
```

```
    # Separate the labels and set up training and test datasets
    y = df[dataset["column_label"]]

    # Drop the `label` column
    df.drop(dataset["column_label"], axis=1)      #where numbering of news
article is done that column is dropped in dataset
    X_train, X_test, y_train, y_test = get_sets(df, dataset["column_text"],
y)
    return [X_train, X_test, y_train, y_test]


  def get_sets(df, column_text, y):
    # Make training and test sets
    # test set is 33% of all
    X_train,          X_test,          y_train,          y_test          =
train_test_split(df[column_text].values.astype('U'),   y,   test_size=0.33,
random_state=53)
    return [X_train, X_test, y_train, y_test]


  def predict(input, clf):
    return [clf.predict(input), clf.predict_proba(input)]


  ### reading the test files to be fed into predictions
  testsarr = list(testdict.keys())
  def read_test_files():
    for testobj in testsarr:
      testdict[testobj]["data"]                                              =
[Path(testdict[testobj]["path"]).read_text(encoding="utf8")]

  vectorizers = ["TF-IDF", "Hashing"]
  datasets = ['fake1', 'sarcasm2', 'hate2']


  classarr = list(classifiers.keys())

  result_dictionary = {}
  fieldnames = ["Accuracy"]

  def do_the_thing():
    for testobj in testsarr:
      for idk in ["Expected: ", "Predicted: "]:
        fieldnames.append(idk + testdict[testobj]["desc"])

  selected_ensembles = {
    "NP, LR, KNN": {
      "combo": ["Naive Bayes", "Logistic Regression", "KNN"],
      "trained": []
      },
    "PA, KNN": {
      "combo": ["Passive-Aggressive", "KNN"],
      "trained": []
      },
    "SVM, PA, LR":
      {
        "combo": ["SVM", "Passive-Aggressive", "Logistic Regression"],
      "trained": []
```

```
        }
    }


    import csv

    from time import time

    def test_model(clf_ensemble, vectorizer, measure, classif_name,
vector_name, dataset_name, modelsize):
        # testing_success = True

        for testobj in testsarr:
            print('wordcount: ' + f'{testdict[testobj]["wordcount"]}')

            t0 = time()
            prediction, probability = predict(get_vectorized(vectorizer,
testdict[testobj]["data"]), clf_ensemble)

            td = time() - t0
            predicted_value = prediction[0]

            expected_value  = testdict[testobj]["expected_vals"][measure]

            print(testdict[testobj]["desc"] + "\t" + str(expected_value) +
"\t\t" + str(predicted_value) + "       " + str(probability[0]) + "    " +
f"{td}")


    def predict_input(clf_ensemble, vectorizer, input_data):
        t0 = time()
        prediction, probability = predict(get_vectorized(vectorizer,
input_data), clf_ensemble)

        td = time() - t0
        predicted_value = prediction[0]

        return [td, predicted_value, probability[0][predicted_value]]

    dataset_files = {}

    vectorizer = 'Hashing'
    dbname = 'fake1'

    modelnam = 'SVM, PA, LR'

    vectorizers = ['Hashing', 'TF-IDF']
    vec_name = 'vec'

    def load_pkl_files():
      # load pkl files
      for dataset_name in datasets:
          dataset = get_dataset(dataset_name)
          measure = dataset["measure"]

          dataset_files[dataset_name] = {}

          for vector in vectorizers:
```

```
            dataset_files[dataset_name][vector] = {}
            for ens_name in list(selected_ensembles.keys()):
                dataset_files[dataset_name][vector][ens_name] = {}
                dataset_files[dataset_name][vector][vec_name] = {}

                mode, size = import_pickle(ens_name, dataset_name + '/' +
vector + '/')

                vec, sizev = import_pickle(vec_name, dataset_name + '/' +
vector + '/')

                dataset_files[dataset_name][vector][ens_name]   =   [mode,
size]
                dataset_files[dataset_name][vector][vec_name] = [vec, size]

                #test_model(mode,   vec,   measure,   ens_name,   vector,
dataset_name, size)

  def get_classifier(classif_name, vector_name, dataset_name):
    return dataset_files[dataset_name][vector_name][classif_name]

  def get_vectorizer(vector_name, dataset_name):
    return dataset_files[dataset_name][vector_name][vec_name]


  load_pkl_files()
```

webserver_config.py

```
  from webserver_functions import predict_input
  from webserver_functions import get_classifier
  from webserver_functions import get_vectorizer

  chosen_datasets = [
      "fake1",
      "sarcasm2",
      "hate2"
  ]

  def get_datasets():
      return chosen_datasets

  from config import datasets


  def generate_prediction(ensemble_method, vectorizer, dataset, text_data):
      mode, size = get_classifier(ensemble_method, vectorizer, dataset)

      vec, sizev = get_vectorizer(vectorizer, dataset)

      prediction = predict_input(mode, vec, [text_data])
      prediction[32] = round(prediction[32] * 100, 2)
      prediction[0] = round(prediction[0], 2)
```

```
prediction.append((datasets[dataset]["label_meaning"][prediction[31]]).lowe
r())

        return prediction
```

main.py

```
    import requests
    from flask import Flask, render_template, request, jsonify # Flask object

    from server.webserver_config import get_datasets, generate_prediction

    app = Flask(__name__)

    @app.route("/")

    def index():
        return render_template('./index.html', datasets = get_datasets())

    @app.route('/predict', methods=['GET','POST'])

    def predict():
        if request.method == "POST":
            to_predict_list = request.form.to_dict()

            print(to_predict_list)

            predictions_arr = []

            total_time = 0

            if (to_predict_list['dataset'] == "all"):
                for chosen_dataset in get_datasets():
                    prediction                                              =
generate_prediction(to_predict_list['ensemble_method'],
to_predict_list['vectorizer'],                                chosen_dataset,
to_predict_list['text_input'])

                    predictions_arr.append(prediction)
                    total_time += prediction[0]
            else:
                prediction                                                  =
generate_prediction(to_predict_list['ensemble_method'],
to_predict_list['vectorizer'],                    to_predict_list['dataset'],
to_predict_list['text_input'])

                predictions_arr.append(prediction)
                total_time += prediction[0]


            return                              render_template('/predict.html',
predictions=predictions_arr, total_time = total_time)


    if __name__ == '__main__':
        app.run(debug=True)
```

Додаток З
Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ПРОГРАМНИЙ МЕТОД АНАЛІЗУ ТЕКСТІВ НОВИН НА НЕПРАВДИВІСТЬ

Доповідач: Мельничук Олексій Геннадійович

Науковий Керівник: к. т. н., доцент Олещенко Любов Михайлівна

Київ – 2024

АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ

- Відсутність універсального алгоритму для аналізу різних категорій новин з різною структурою тексту та способом подачі інформації (недостовірні новини, сатира, мова ворожнечі тощо).
- Відсутність оптимізації процесу перевірки достовірності новинних текстів на основі аналізу різними методами класифікації, у тому числі ансамблевими методами.

2

ПОСТАНОВКА ЗАДАЧІ

Об’єкт дослідження: процес аналізу текстів новинних статей на предмет невідповідності встановленим фактам.

Предмет дослідження: методи програмної інженерії для розроблення програмних систем аналізу текстів новин на неправдивість з використанням алгоритмів машинного навчання.

Мета дослідження: підвищення точності виявлення достовірності аналізу текстів новин для спрощення ідентифікування неправдивої інформації.

Завдання дослідження: розробити програмний метод для аналізу текстів новин на недостовірність.

Окремі завдання

1. Вивчення та опрацювання наукових матеріалів та розробок алгоритмів для поставленої задачі.
2. Створення ансамблевого алгоритмічного методу на базі обраних класифікаторів.
3. Розробка вебсервісу для тестування розробленого програмного методу.
4. Оцінка та порівняння ефективності отриманих результатів роботи запропонованого програмного методу з аналогами.

ВІДОМІ РІШЕННЯ

Наукові:

- Моделі обробки природнього мовлення
- Згорткові нейронні мережі
- RoBERTa, XLM-R Facebook/Meta

Комерційні:

- The Factual
- ClaimBuster
- FullFact

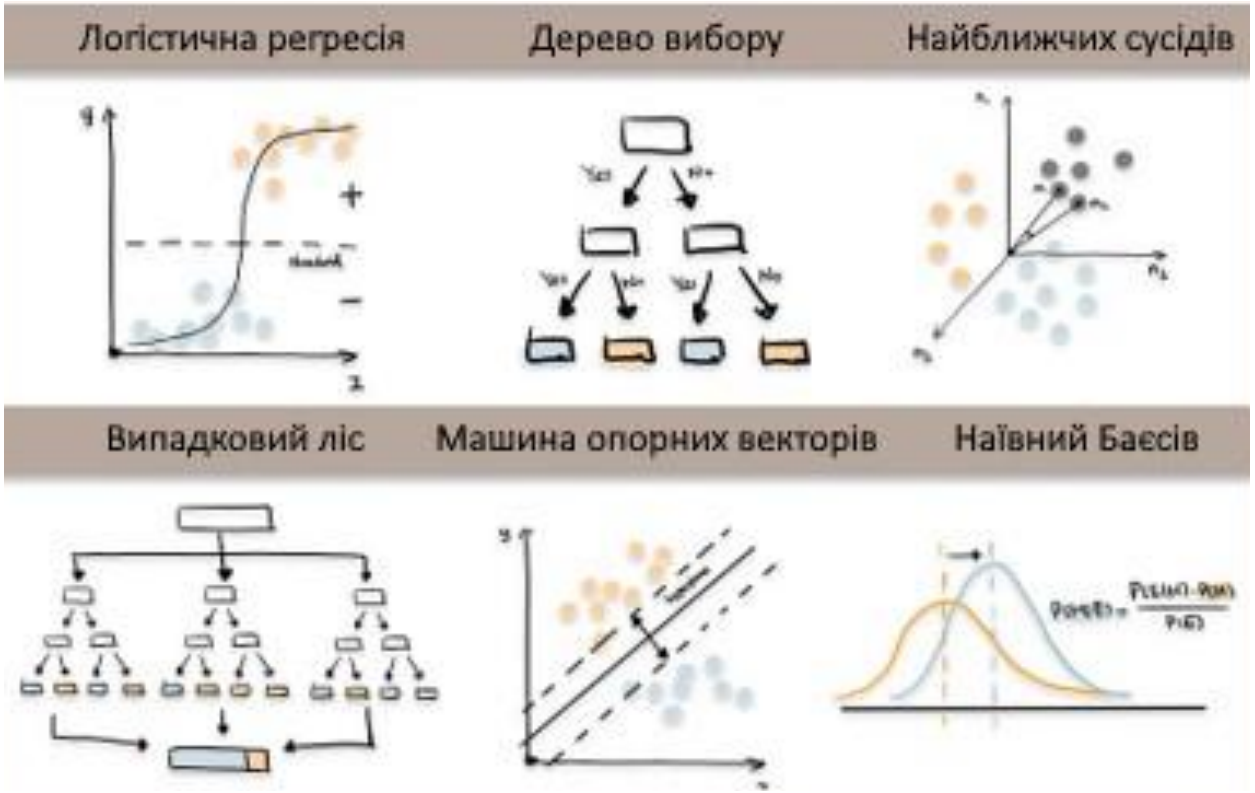
5

МЕТОДИ ОБРОБКИ ПРИРОДНЬОГО МОВЛЕННЯ

- З наглядом
- «Ліниве» навчання (K-найближчих сусідів)
- Пакетне навчання (пасивно-агресивний)
- Бінарні/мультикласові
- Лінійні/нелінійні

6

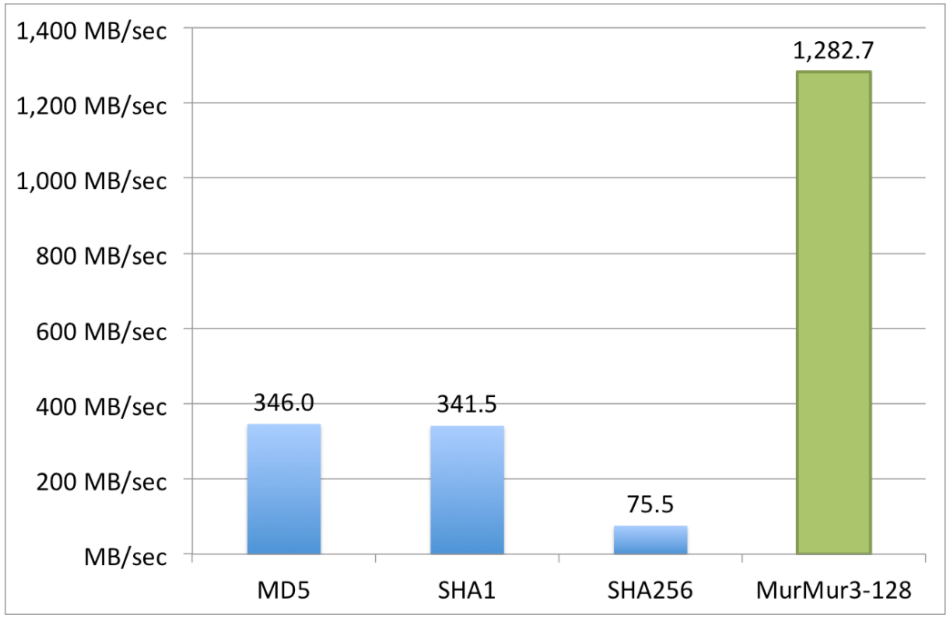
ВИКОРИСТАНІ КЛАСИФІКАТОРИ



7

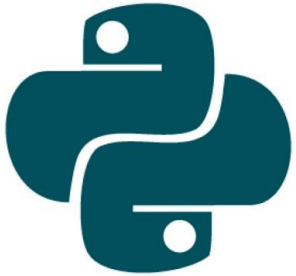
ВИКОРИСТАНІ ВЕКТОРИЗАТОРИ

Обрані векторизатори:
- TF-IDF
- (кількісний)
з хешуванням MurMur3



8

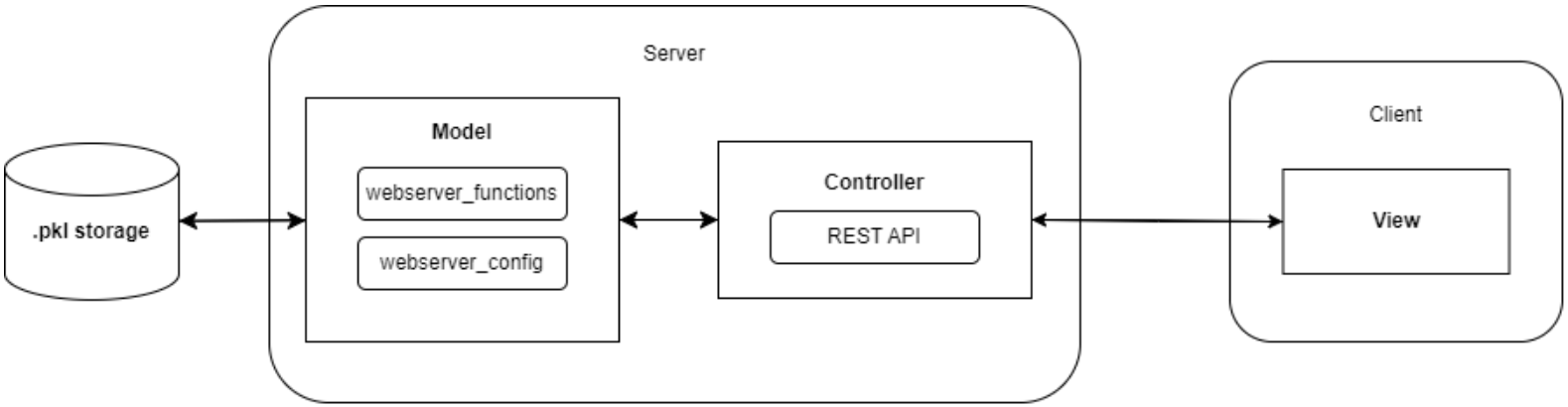
ТЕХНОЛОГІЇ ТРЕНУВАННЯ КЛАСИФІКАТОРІВ



NLTK

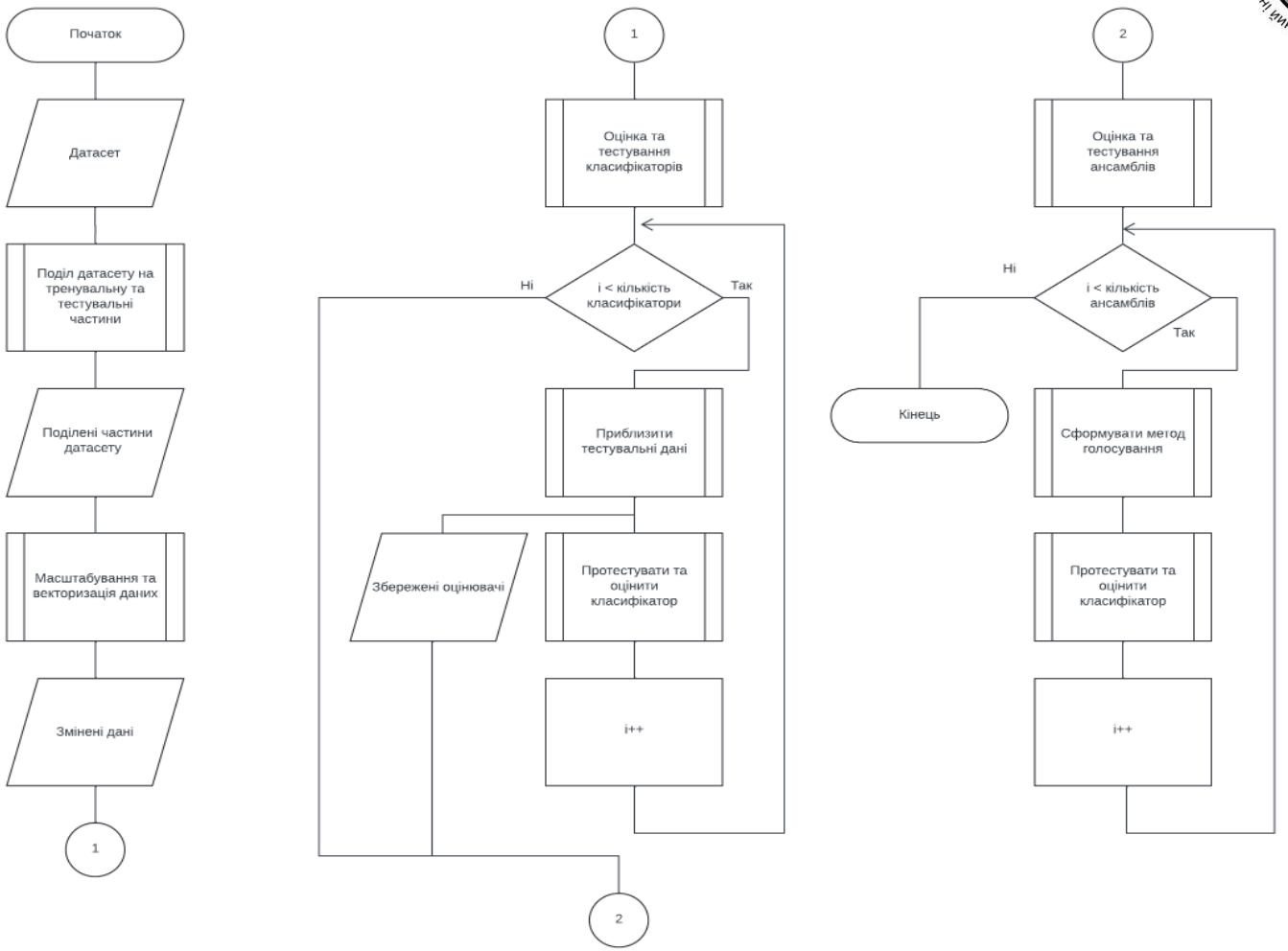
9

АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



10

СХЕМИ АЛГОРИТМУ ТРЕНУВАННЯ

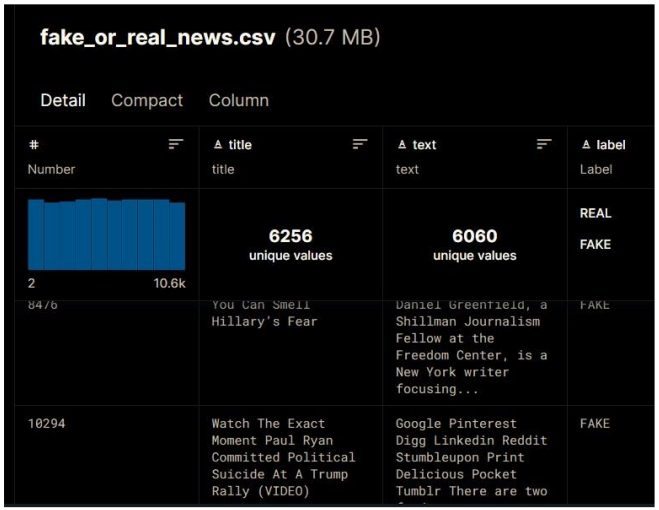


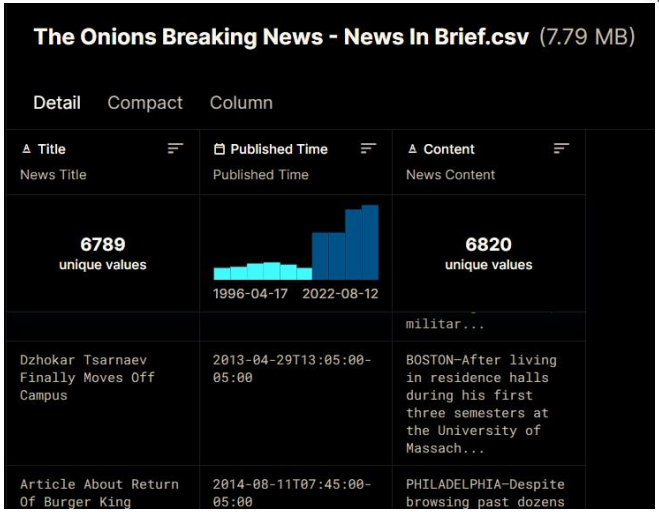
11

ПЕРЕТВОРЕННЯ НАБОРІВ ДАНИХ

12

ДЖЕРЕЛА НАБОРІВ ДАНИХ



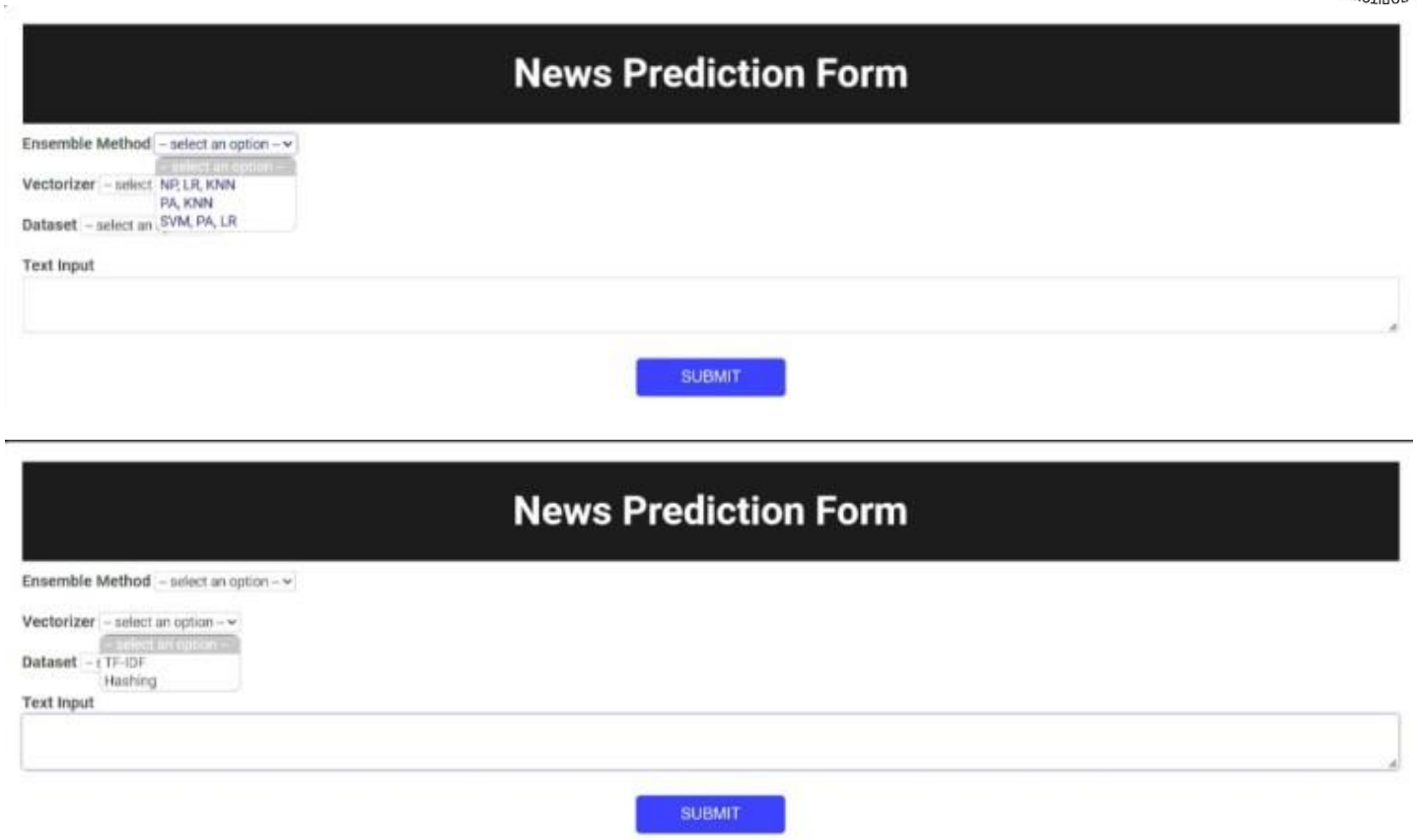


13

ВИКОРИСТАНІ АНСАМБЛЕВІ МЕТОДИ

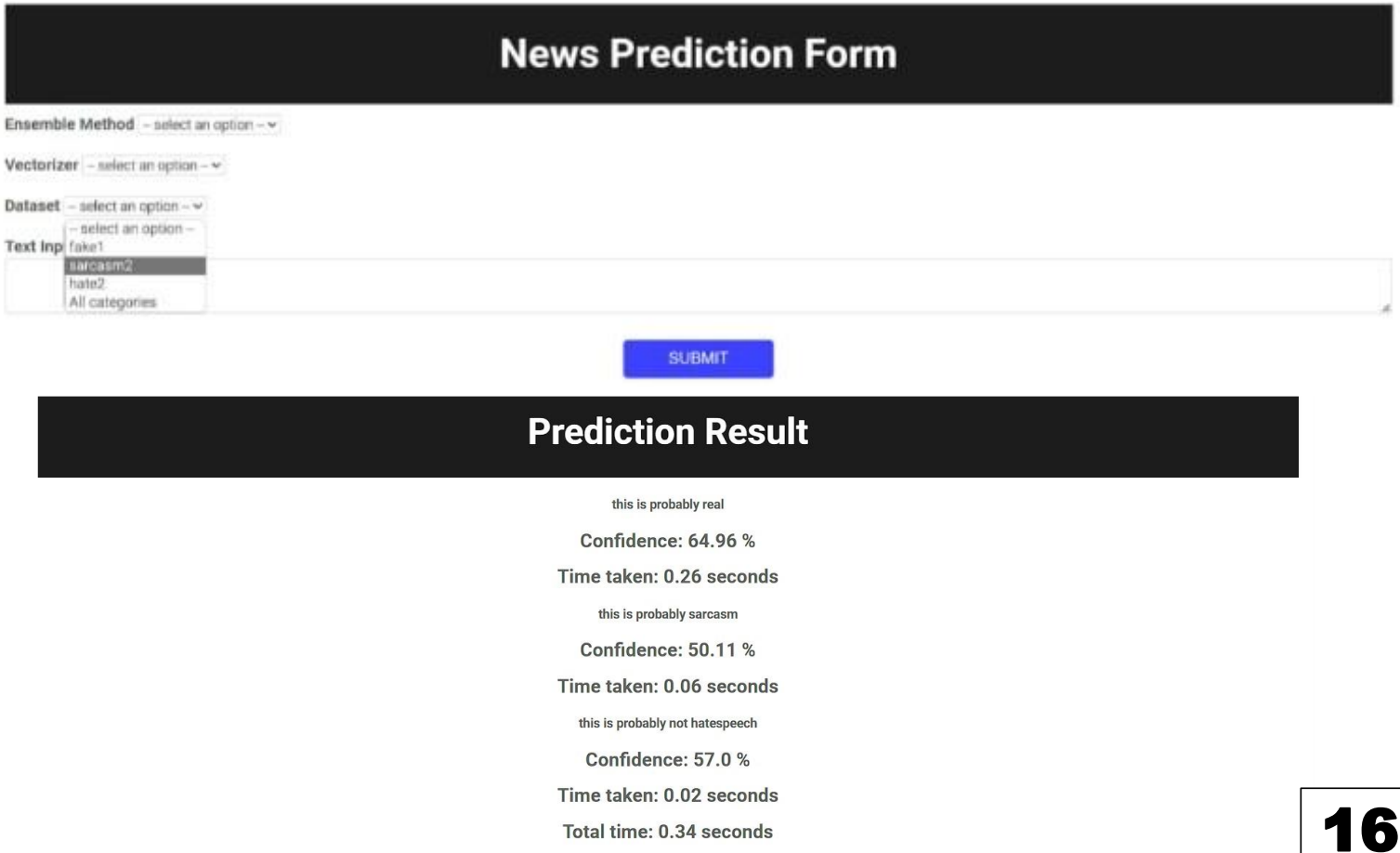
14

ВИБІР МЕТОДУ КЛАСИФІКАЦІЇ ТА ТИПУ ВЕКТОРИЗАТОРА



15

ВИБІР КАТЕГОРІЇ НОВИН ТА РЕЗУЛЬТАТ ПЕРЕДБАЧЕННЯ

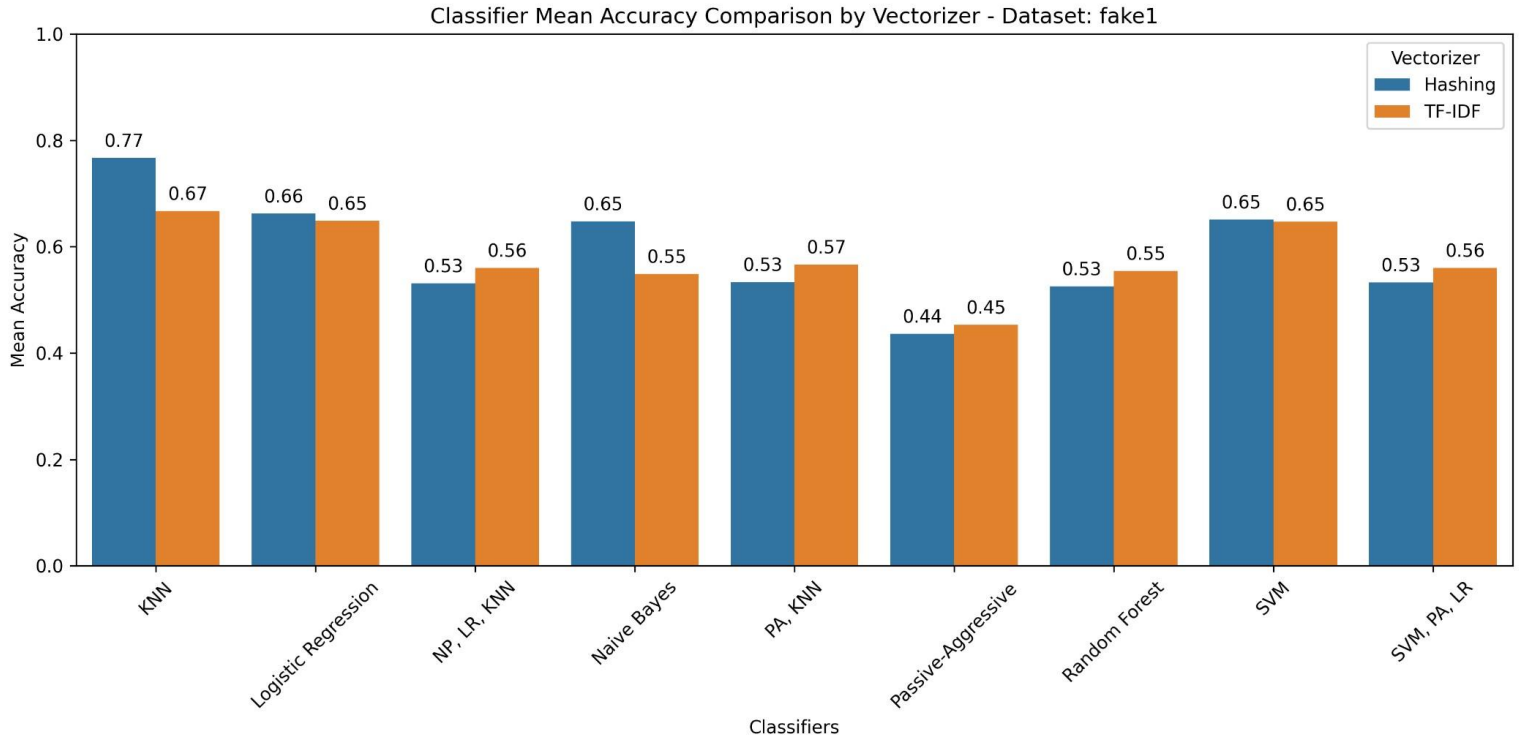


16

РЕЗУЛЬТАТ АНАЛІЗУ ТЕКСТУ

17

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

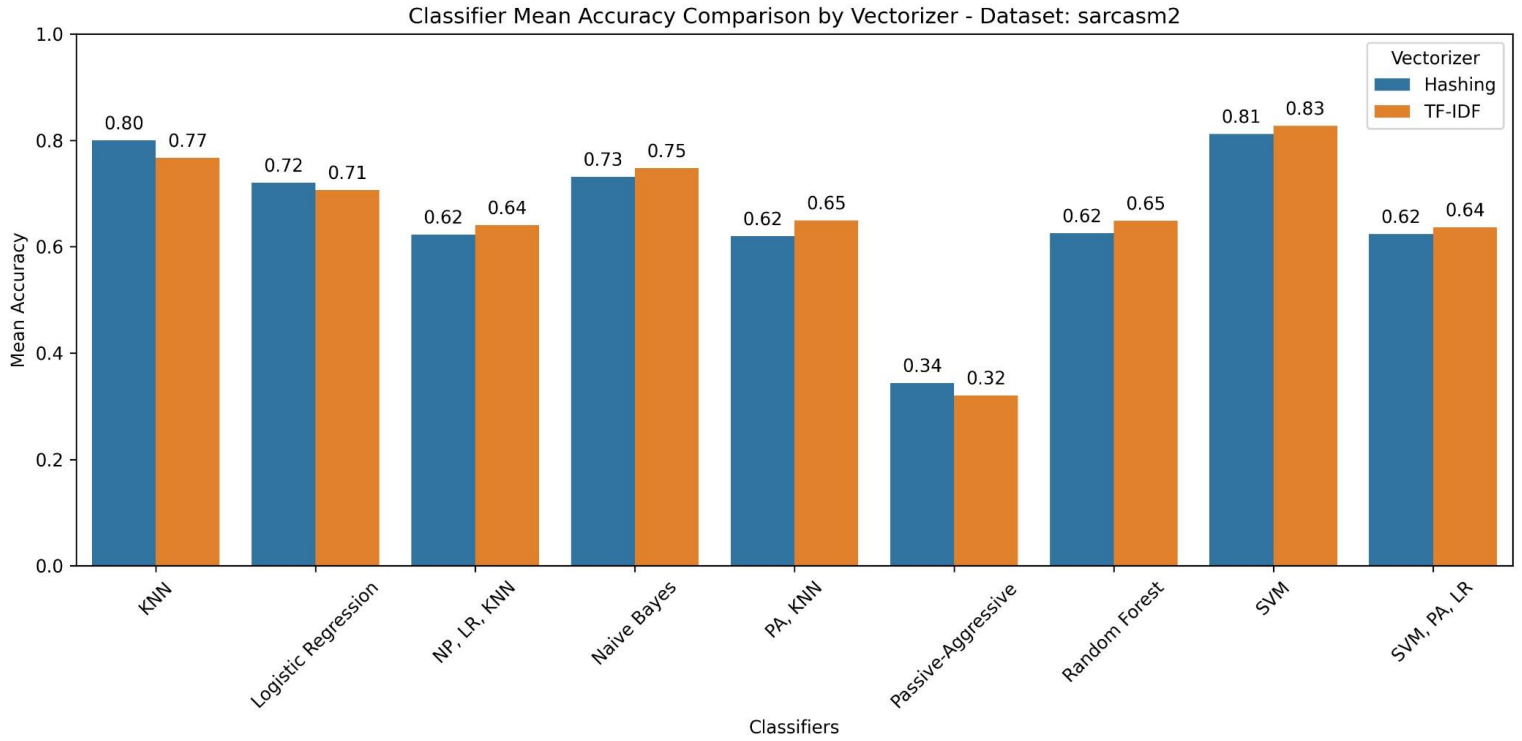


Найкращий TF-IDF: Логістична Регресія

Найкращий з хешуванням: К-найближчих

18

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

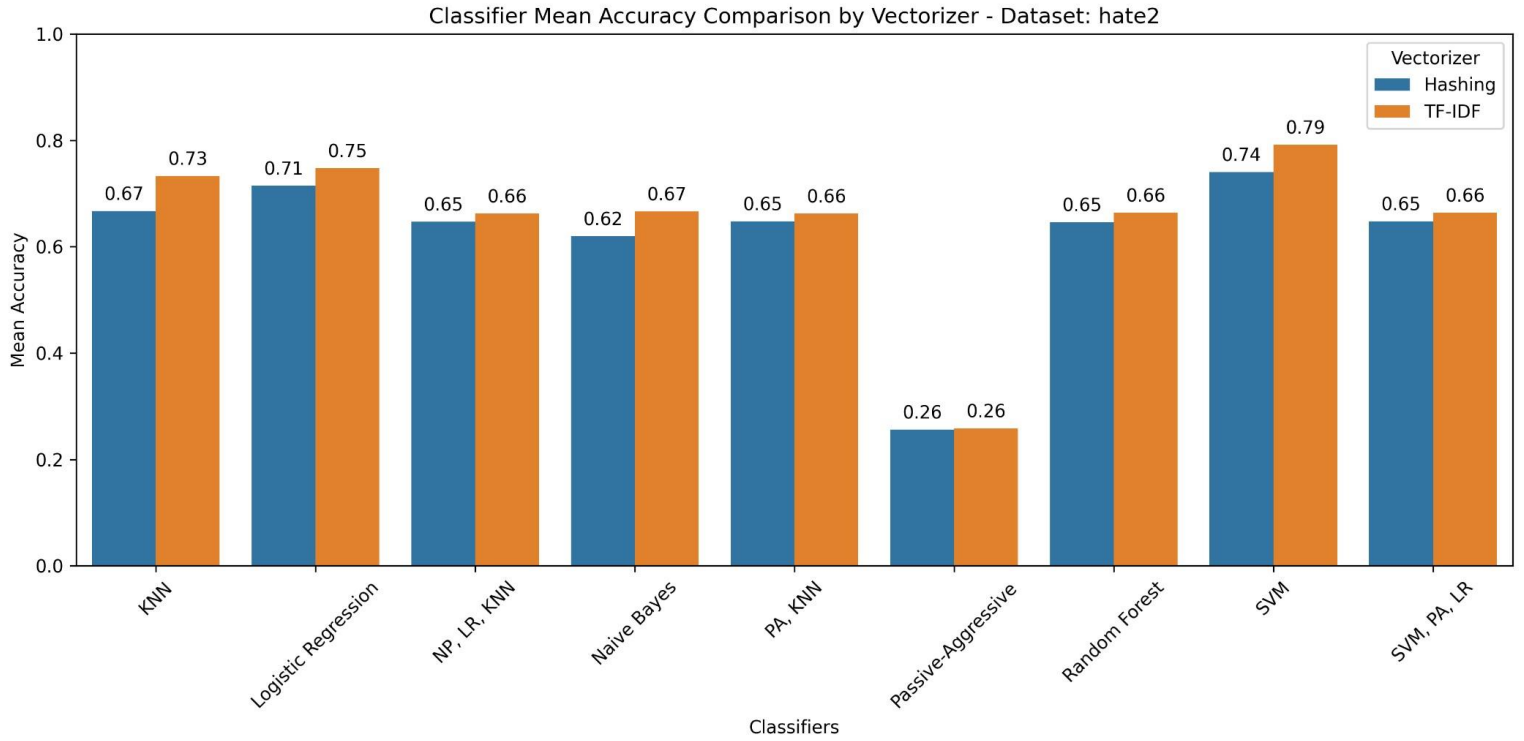


Найкращий TF-IDF: Опорних векторів

Найкращий з хешуванням: Опорних векторів

19

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ



Найкращий TF-IDF: Опорних векторів

Найкращий з хешуванням: Опорних векторів

20

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Класифікатор	Середня точність	Середня кількість вдалих передбачень	Середня швидкість (секунди)
Наївний Баєсів NB	93.23%	4	0.0148
Пасивно-агресивний (PA)	94.8%	4.67	0.002
Лінійний опорних векторів (SVM)	95.58%	4.67	0.0056
Логістична регресія (LR)	94.27%	4.5	0.0018
k-найближчих сусідів (KNN)	90.79%	4.83	0.0406
Випадкового лісу (FOREST)	86.74%	4	0.2868
NB, KNN, LR	94.54%	4.33	0.6833
KNN, PA	78.72%	4	0.4743
SVM, PA, LR	94.98%	4.83	0.716

НАУКОВА НОВИЗНА

Вперше розроблено удосконалений програмний метод аналізу текстів новинних статей, характерними рисами якого є використання алгоритму попередньої комбінованої обробки та аналізу текстів новин різних категорій ансамблевими методами класифікації, що дає змогу підвищити точність розпізнавання у середньому до 4.18% порівняно з класичними методами.

ПРАКТИЧНА ЦІННІСТЬ

Практична цінність полягає у програмній реалізації запропонованого вдосконаленого методу обробки та аналізу текстових даних на неправдивість у вигляді вебзастосунку, що є конкурентоспроможним та готовим до розгортання у вигляді комерційного продукту.

ВИСНОВКИ

1. Здійснено аналіз існуючих методів та засобів аналізу новинних текстів на неправдивість.
2. Створено ансамблевий метод аналізу тексту на базі обраних класифікаторів.
3. Розроблено програмний метод для аналізу новинних текстів на неправдивість та програмне забезпечення у вигляді вебзастосунку.
4. Проаналізовано отримані результати роботи алгоритму з аналогами.

АПРОБАЦІЯ

1. Тези конференції: Мельничук О.Г., Олещенко Л.М. Програмний метод аналізу текстів новин на неправдивість. Прикладна математика та комп'ютинг ПМК-2023. Збірник тез доповідей XVI наукової конференції магістрантів та аспірантів (Київ, 28 - 30 листопада 2023 р.), с. 517-521.
2. Стаття у фаховому журналі категорії Б:
Олещенко Л.М., Мельничук О.Г. Застосування ансамблевих методів машинного навчання для виявлення неправдивого тексту. Вісник Херсонського національного технічного університету. 2024. № 1(88). С.258-263.

25

Дякую за увагу!