Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Комп'ютерних інтелектуальних технологій та систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Розробка системи виявлення ознак посттравматичного розладу (ПТСР) з використанням методів машинного навчання
1(тема)

Виконав:
здобувач IV року навчання,
групи КІУКІ – 21–10
Юлія СПИЦИНА
(власне ім'я, прізвище)

Спеціальність
123 Комп'ютерна інженерія
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник ас. каф. КІТС Кирило ОЛІЙНИК
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри _____ Олег РУДЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Комп'ютерних інтелектуальних технологій та систем

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Спициній Юлії Андріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи виявлення ознак посттравматичного розладу (ПТСР) з використанням методів машинного навчання

затверджена наказом університету від "21" Травня 2025 р. № 399 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 19.06.2024

3. Вхідні дані до роботи
1. Документація мови програмування Python.
2. Анотовані датасети.
3. Алгоритми машинного навчання.
4. Методи обробки текстів.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області.
2. Дослідження обробки даних.
3. Розробка моделей машинного навчання.
4. Навчання та тестування моделей
5. Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)

Слайд-презентація – 19 слайдів

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення з літературними	26.05.2024-28.05.2024	Виконано
	джерелами, аналіз та вибір методу		
	вирішення поставленої задачі		
2	Розробка алгоритмів рішення,	01.06.2024-02.06.2024	Виконано
	вибір системних засобів вирішення		
	завдань роботи		
3	Проектування системи	03.06.2024-10.06.2024	Виконано
4	Налагодження та тестування системи	11.06.2024-12.06.2024	Виконано
5	Оформлення матеріалів атестаційної роботи	13.06.2024-15.06.2024	Виконано
6	Подання атестаційної роботи керівникові та	16.06.2024-17.06.2024	Виконано
	її попередній захист		
7	Подання роботи на рецензування	18.06.2024-19.06.2024	Виконано

Дата видачі завдання 26 травня 2025

Здобувач _____
(підпис)

Керівник роботи _____ ас. Олійник К.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 58 с., 10 рис., 2 дод., 10 джерел.

МАШИННЕ НАВЧАННЯ, ПТСР, БАЙЄС, LSTM, RNN

Метою проекту є розробка та впровадження моделей машинного навчання для визначення ймовірності посттравматичного стресового розладу (ПТСР) на основі текстових даних.

У ході виконання проекту проведено аналіз предметної області, що включає вивчення основних аспектів посттравматичного стресового розладу, його впливу на психологічний стан людини та можливостей автоматизованої діагностики. Було розглянуто сучасні методи машинного навчання, такі як класифікатор Байєса та LSTM. Сформульовано проблему дослідження з акцентом на специфіку ПТСР. Для реалізації проекту використано текстовий датасет, що містить емоційно забарвлені висловлювання. Проведено попередню обробку даних, яка включає очищення тексту, нормалізацію та токенізацію, що забезпечило якісний вхід для алгоритмів навчання.

Реалізовано два підходи: наївний класифікатор Байєса та модель LSTM. Класифікатор Байєса продемонстрував ефективність для аналізу простих текстів із чіткими емоційними маркерами, модель LSTM виявилася також потужною для аналізу контексту та послідовності слів. Навчання та тестування обох моделей дозволило порівняти їхню точність, чутливість.

Запропонована система демонструє високу ефективність і може бути використана для автоматизації первинної психологічної діагностики, підтримуючи роботу психологів і соціальних працівників.

Практичне значення роботи полягає у створенні інструменту для моніторингу психоемоційного стану людей, що має застосування у сфері психології, медицини та соціальної роботи.

ABSTRACT

Bachelor’s thesis: 58 pages, 10 figuress, 2 appendices, 10 sources.

MACHINE LEARNING, PTSD, BAYES, LSTM, RNN

The goal of the project is to develop and implement machine learning models to determine the probability of post-traumatic stress disorder (PTSD) based on text data.

During the project, an analysis of the subject area was conducted, which includes the study of the main aspects of post-traumatic stress disorder, its impact on the psychological state of a person, and the possibilities of automated diagnostics. Modern machine learning methods, such as the Bayes classifier and LSTM, were considered. The research problem was formulated with an emphasis on the specifics of PTSD.

A text dataset containing colored statements was implemented the project. Data preprocessing was performed, which included text cleaning, normalization, and tokenization, provided high-quality input for learning algorithms.

Two approaches were implemented: a naive Bayes classifier and an LSTM model. The Bayesian classifier demonstrated efficiency for analyzing simple texts with clear emotional markers, while the LSTM model also proved to be powerful for analyzing context and word sequences. Training and testing of both models allowed us to compare their accuracy and sensitivity.

The proposed system demonstrates high efficiency and can be used to automate primary psychological diagnostics, supporting the work of psychologists and social workers.

The practical significance of the work lies in creating a tool for monitoring the psycho-emotional state of people, which has applications in the fields of psychology, medicine and social work.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

МН – машинне навчання

ПТСР – посттравматичний стресовий розлад

LSTM – довга короткочасна пам'ять

RNN – рекуретна нейрона мережа

SGD – стохастичний градієнтний спуск

API – інтерфейс прикладного програмування

ВСТУП

Посттравматичний стресовий розлад (ПТСР) є серйозним психологічним станом, що виникає внаслідок переживання травматичних подій. Вчасна діагностика цього розладу є надзвичайно важливою для надання необхідної психологічної допомоги та підтримки. З огляду на широке поширення текстової комунікації, особливо в цифрових середовищах, аналіз текстових даних набуває все більшого значення у виявленні ознак ПТСР.

Вчасна діагностика ПТСР є критично важливою для надання необхідної психологічної допомоги. Однак через значну кількість випадків і обмежені ресурси традиційні методи діагностики не завжди дозволяють оперативно реагувати на виклики. У цьому контексті автоматизовані інструменти, які базуються на сучасних технологіях, таких як машинне навчання, можуть стати важливим засобом підтрим

Методи машинного навчання, зокрема класифікація текстів, відкривають нові можливості для автоматизації процесів діагностики. Використання алгоритмів, таких як наївний класифікатор Байєса або рекурентні нейронні мережі (RNN), дозволяє враховувати контекст тексту та визначати емоційний стан автора.

Актуальність теми дослідження обумовлена тим, що в умовах війни, яку переживає наша країна, проблема ПТСР стає надзвичайно актуальною. Велика кількість людей, зокрема військовослужбовці, цивільні, які постраждали від бойових дій, переселенці та ті, хто втратив близьких, мають підвищений ризик розвитку цього розладу.

Мета роботи полягає у дослідженні можливостей машинного навчання для аналізу текстових даних з метою визначення ймовірності ПТСР. У процесі виконання проекту передбачено розробку, навчання та оцінку ефективності моделей, заснованих на алгоритмах класифікації текстів.

1 АКТУАЛЬНІСТЬ ЗАДАЧІ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Посттравматичний стресовий розлад (ПТСР) є складним психічним порушенням, що виникає внаслідок травматичних подій, таких як війна, насильство, катастрофи або серйозні особисті потрясіння. ПТСР супроводжується тривожністю, нав’язливими спогадами, порушенням сну, емоційною притупленістю та іншими симптомами, які значно знижують якість життя людини.

Традиційні методи діагностики ПТСР базуються на клінічних інтерв’ю, стандартизованих опитувальниках та оцінці поведінки фахівцем. Однак ці методи мають низку обмежень: суб’єктивність оцінки, низький рівень доступності професійної допомоги, соціальна стигматизація та складність у виявленні ранніх ознак розладу. Особливо це актуально для вразливих груп населення, таких як ветерани, біженці або жертви насильства.

У зв’язку з активним розвитком цифрових технологій, зокрема аналізу тексту, обробки природної мови (NLP), соціальних мереж і машинного навчання, з’явилися нові можливості для автоматизованого виявлення психоемоційних станів. Обробка текстових повідомлень, медичних записів або даних з анкетування за допомогою інтелектуальних алгоритмів дозволяє ідентифікувати ознаки ПТСР навіть без прямої участі лікаря. Це відкриває шлях до ранньої діагностики, персоналізованого моніторингу та запобігання загостренням.

Сучасні дослідження демонструють ефективність використання методів машинного навчання, таких як Random Forest, SVM, нейронні мережі, зокрема LSTM та трансформери, для обробки природної мови та класифікації психологічних станів. У публікаціях журналів Frontiers in Psychiatry, Journal of Affective Disorders та IEEE Transactions on Affective Computing відзначається

висока точність таких моделей у задачах виявлення депресії, тривожності та ПТСР. Водночас більшість реалізацій мають обмежену відкритість або вузьку спеціалізацію.

Таким чином, предметна область поєднує завдання з обробки текстових даних, побудови класифікаційних моделей, розробки інтерфейсів для збору інформації та створення доступних інструментів для фахівців або самодіагностики.

1.2 Актуальність обраної теми

Зростання кількості випадків ПТСР, зокрема в умовах воєнного стану, кризових ситуацій і соціальної нестабільності, створює потребу у нових, технологічно просунутих інструментах для психічної допомоги. Зважаючи на обмежену кількість фахівців-психотерапевтів, високу вартість медичних послуг та стигматизацію психічних захворювань у суспільстві, виникає потреба у створенні автоматизованих систем для виявлення симптомів ПТСР.

Використання методів машинного навчання та обробки природної мови дозволяє ефективно аналізувати велику кількість текстових або соціальних даних, визначати потенційні ризики та формувати попередження про можливі психоемоційні порушення. Такий підхід забезпечує масштабованість, доступність і об’єктивність у процесі діагностики.

Особливої актуальності ця тема набуває в контексті розвитку систем eHealth, дистанційної медицини та цифрових помічників. Вони можуть бути інтегровані у мобільні застосунки, чат-боти чи системи підтримки лікарських рішень, що забезпечує не тільки раннє виявлення, а й динамічний моніторинг стану пацієнта.

Таким чином, розробка системи автоматичного виявлення ПТСР із використанням алгоритмів машинного навчання є надзвичайно актуальним і соціально важливим напрямом дослідження.

1.3 Постановка задачі

У зв’язку з необхідністю забезпечити доступну й ефективну діагностику ПТСР на ранніх етапах, актуальним стає розроблення системи, яка застосовуватиме методи машинного навчання для аналізу текстових даних і виявлення ознак посттравматичних розладів. Такий підхід дозволяє здійснювати автоматичну обробку великих обсягів інформації, отриманої з електронних повідомлень або соціальних мереж, і виявляти приховані закономірності, характерні для порушень. Впровадження подібних систем може значно підвищити точність і швидкість первинного скринінгу та розширити доступ до психіатричної допомоги.

Мета роботи — розробити прототип автоматизованої системи виявлення ПТСР на основі аналізу тексту з використанням методів машинного навчання.

Для досягнення цієї мети необхідно вирішити такі завдання:

– обрати оптимальні мови програмування, бібліотеки та інструменти для побудови системи;

– визначити вимоги до структури та наповнення навчального датасету;

– очистити та нормалізувати дані, застосовуючи NLP-методи (токенізація, лематизація, векторизація);

– розробити та реалізувати прототип системи виявлення ПТСР з використанням алгоритмів машинного навчання;

– провести тестування точності моделей.

Таким чином, дослідження поєднує актуальні потреби суспільства у сфері психічного здоров’я з сучасними технологічними можливостями, відкриваючи нові шляхи для ефективної діагностики ПТСР.

2 ТЕОРИТИЧНІ ВІДОМОСТІ

2.1 Основні поняття про ПТСР

ПТСР (посттравматичний стресовий розлад) — це стан психічного здоров’я, який виникає у деяких людей після того, як вони пережили травматичну подію або були свідками. Травматична подія може загрожувати життю або становити серйозну загрозу фізичному, емоційному чи духовному благополуччю. ПТСР вражає людей будь-якого віку.

Люди з ПТСР мають інтенсивні та нав’язливі думки та почуття, пов’язані з переживаннями, які тривають довго після події. ПТСР включає такі реакції на стрес, як:

- тривога, депресивний настрій або почуття провини чи сорому;
- наявність спогадів або кошмарів;
- уникання ситуацій, місць і дій, пов’язаних із травматичною подією.

Ці симптоми викликають стрес і заважають повсякденній діяльності.

Травма або травматична подія — це все, що серйозно загрожує людському існуванню або почуттю безпеки. Це не обов’язково має бути окрема подія (наприклад, автомобільна аварія) — це може бути довгострокова травма, як-от переживання війни. Травма також не обов’язково має статися безпосередньо з людиною, яка стала свідком травматичної події. Крім того, у особи може розвинутися посттравматичний стресовий розлад після того, як вона дізнаєтеся, що травматична подія сталася з близькою людиною [1].

Традиційні методи діагностики посттравматичного стресового розладу зазвичай базуються на використанні стандартизованих шкал або клінічних інтерв'ю. Ці методи мають низку суттєвих обмежень. Передусім відповіді пацієнтів можуть бути значно викривлені такими чинниками, як актуальний психологічний стан, контекст бесіди, особисті стосунки з клініцистом, поточний емоційний стан людини та спотворення спогадів через попередній

життєвий досвід. Крім того, такі підходи недостатньо враховують часову динаміку проявів ПТСР. Ще однією проблемою є те, що багато людей не усвідомлюють або соромляться своїх симптомів, через що не звертаються за професійною допомогою. Нарешті, діагностика шляхом особистих інтерв'ю є дорогим процесом як у фінансовому, так і в часовому аспекті, і тому може бути недоступною широким верствам населення. У зв’язку з цим виникає потреба у створенні більш ефективного, економічного та масштабованого методу діагностики ПТСР, придатного для охоплення великих груп людей.

2.2 Машине навчання

Машинне навчання — це напрям штучного інтелекту, що дає змогу комп’ютерним системам автоматично покращувати свої результати шляхом аналізу наявних даних, без необхідності чіткого програмування кожного кроку. Основна ідея полягає в тому, що замість написання алгоритмів з фіксованими правилами, системі надається доступ до великої кількості прикладів (даних), з яких вона самостійно виводить закономірності.

Машинне навчання фундаментально побудовано на даних, які служать основою для моделей навчання та тестування. Дані складаються з входів (об’єктів) і виходів (міток). Модель вивчає шаблони під час навчання та перевіряється на невидимих даних для оцінки її продуктивності та узагальнення. Для того, щоб робити прогнози, існують важливі кроки, через які дані проходять, щоб створити модель машинного навчання, яка може робити прогнози.

Машинне навчання можна умовно розділити на три типи:

– контрольоване навчання: навчає моделі на позначених даних для прогнозування або класифікації нових, невидимих даних;

– неконтрольоване навчання: знаходить закономірності або групи в немаркованих даних, як-от кластеризація або зменшення розмірності;

– навчання з підкріпленням: вчиться методом проб і помилок, щоб

максимізувати винагороду, що ідеально підходить для завдань прийняття рішень.

Ефективне впровадження машинного навчання надає компаніям конкурентні переваги, дозволяючи з точністю визначати тенденції та прогнозувати результати, що перевершує традиційну статистику або людський інтелект. Однак використання машинного навчання також супроводжується низкою бізнесових викликів. По-перше, це може бути доволі витратним процесом. Машинне навчання потребує дорогого програмного забезпечення, обладнання та відповідної інфраструктури для управління даними. Більше того, проєктами займаються спеціалісти, такі як науковці та інженери з обробки даних, які мають високий рівень зарплат. Ще однією суттєвою проблемою є упередженість алгоритмів. Якщо навчальні набори даних виключають певні групи або містять помилки, це може призводити до створення неточних моделей. У результаті такі моделі не лише можуть виявитися непридатними для використання, але й спричиняти дискримінаційні результати у найгіршому випадку [2].

У сфері машинного навчання важливо не тільки розуміти основні концепції, а й використовувати правильні інструменти. Ці інструменти, зокрема мови програмування та спеціалізовані бібліотеки, є основою для реалізації й розгортання алгоритмів машинного навчання. Розглянемо деякі з найбільш популярних інструментів у цій галузі. Python є загальновизнаною мовою для роботи з машинним навчанням завдяки своїй простоті та зрозумілості, що робить її чудовим вибором для початківців. Крім того, вона має потужну екосистему бібліотек, створених спеціально для роботи з даними та алгоритмами машинного навчання. Наприклад, бібліотеки NumPy і Pandas використовуються для обробки та аналізу даних, тоді як Matplotlib забезпечує можливості візуалізації. Scikit-learn пропонує різноманітний набір алгоритмів машинного навчання, а TensorFlow і PyTorch є популярними інструментами для проєктування та тренування нейронних мереж [3].

3 МЕТОДОЛОГІЯ РОЗРОБКИ СИСТЕМИ

3.1 Python

Python — це високорівнева, інтерпретована мова програмування універсального призначення, яка завдяки своїй лаконічності, логічній будові та читабельності коду здобула надзвичайну популярність у різних областях IT. Її синтаксис наближений до природної мови, що суттєво полегшує навчання та знижує поріг входу навіть для новачків. Зокрема, Python став головним інструментом для вирішення задач у галузі штучного інтелекту, машинного навчання та обробки природної мови [4].

Одним із основних факторів, що сприяли стрімкому поширенню Python у сфері машинного навчання, є наявність багатої екосистеми бібліотек та фреймворків, які покривають усі етапи розробки моделей — від збору та обробки даних до навчання, оптимізації та візуалізації результатів. Основні переваги Python у контексті машинного навчання включають:

– простоту та зрозумілість синтаксису, що дає змогу швидко реалізовувати ідеї та експериментувати з алгоритмами без необхідності витрачати багато часу на налаштування мови або деталей реалізації;

– велику екосистему бібліотек, таких як NumPy, Pandas, scikit-learn, TensorFlow, PyTorch, Keras, що забезпечують підтримку різноманітних алгоритмів, починаючи від класичних методів класифікації і регресії, і закінчуючи глибинними нейронними мережами;

– підтримку інтеграції з іншими мовами і технологіями, що дозволяє використовувати Python у складі комплексних систем, підключати швидкі нативні модулі на C/C++ або використовувати його для обробки даних і прототипування, а потім переносити критично важливі компоненти на більш продуктивні платформи;

– універсальність застосування — Python однаково ефективний як для дослідницьких проєктів, так і для промислових рішень, включаючи обробку великих обсягів даних, автоматичне розпізнавання образів, природну мову (NLP), системи рекомендацій, прогнозування та інші напрямки штучного інтелекту.

Завдяки цим особливостям Python став стандартом де-факто для розробки моделей машинного навчання. В роботі над системою виявлення ознак посттравматичного стресового розладу Python використовується для збору, очищення та попередньої обробки текстових даних, формування навчальних вибірок, реалізації алгоритмів класифікації, а також для аналізу отриманих результатів і візуалізації продуктивності моделей. Це дозволяє створити гнучку та ефективну платформу для автоматизованої діагностики на основі текстового аналізу[5].

3.2 Бібліотеки Python

Для реалізації машинного навчання у Python існує великий набір спеціалізованих бібліотек та фреймворків, які суттєво полегшують процес створення, навчання, тестування й оптимізації моделей. Ці інструменти дозволяють ефективно обробляти як структуровані, так і неструктуровані дані, зокрема текстову інформацію, що є надзвичайно важливим у задачах аналізу публікацій із соціальних мереж.

Завдяки цим бібліотекам розробник може сконцентруватися на логіці моделі та аналізі результатів, не витрачаючи надмірних зусиль на реалізацію низькорівневих функцій. Це значно пришвидшує цикл експериментування і сприяє створенню більш точних та адаптивних моделей.

– Matplotlib та Seaborn — бібліотеки для візуалізації даних і результатів класифікації, які допомагають інтерпретувати і аналізувати отримані результати;

– NLTK (Natural Language Toolkit) — набір інструментів для

попередньої обробки тексту: токенізація речень і слів, видалення стоп-слів, стемінг (зведення слів до їх базових форм), що є важливим етапом підготовки даних перед навчанням моделей;

– scikit-learn — популярна бібліотека машинного навчання, яка використовується для векторизації тексту (TF-IDF), розділення вибірки на навчальну і тестову, побудови класичних моделей (наївний баєсівський класифікатор, випадковий ліс, SVM), а також для обчислення метрик якості (точність, повнота, F1-міра, матриця помилок);

– scikit-plot — бібліотека для спрощеного візуального аналізу результатів класифікації, зокрема побудови матриць неточностей і ROC-кривих;

– TensorFlow і Keras — фреймворки для побудови глибинних нейронних мереж, які застосовуються для розробки рекурентних нейронних мереж (LSTM, BiLSTM), згорткових шарів (Conv1D), шарів ембеддінгу та інших компонентів моделей глибокого навчання;

– Gensim — бібліотека для створення і використання векторних подань слів (Word2Vec), що дозволяє моделювати семантичні зв’язки між словами.

Завдяки такому широкому набору інструментів у середовищі Python забезпечується повний цикл обробки текстових даних — від етапу їх очищення, нормалізації та попередньої підготовки до побудови, навчання й оптимізації моделей машинного та глибокого навчання. Після цього реалізуються етапи оцінювання точності, візуалізації результатів і подальшого вдосконалення моделей на основі отриманих метрик, що у комплексі дозволяє створювати ефективні, масштабовані та інтерпретовані системи аналізу текстової інформації [6].

3.3 Огляд існуючих API

Існують офіційні API (Application Programming Interface) соціальних мереж, які дозволяють отримувати доступ до публічної інформації

користувачів, зокрема до постів, коментарів та іншого контенту. Такі API забезпечують структурований, регламентований і безпечний спосіб збору даних із платформ, що активно використовуються у дослідженнях. У розробці систем для виявлення психологічних станів за текстовими повідомленнями ці інструменти відіграють важливу роль, оскільки надають доступ до релевантного користувацького контенту, необхідного для аналізу.

Twitter API є одним із найпопулярніших інструментів для збору текстових даних. Він забезпечує доступ до твітів, ретвітів, лайків, часових міток і метаданих користувачів. API підтримує як безкоштовний рівень з обмеженням кількості запитів, так і розширені платні тарифи. Серед ключових можливостей — повнотекстовий пошук, фільтрація за ключовими словами, хештегами, мовою та геолокацією. Завдяки великій кількості коротких повідомлень, платформа широко використовується в NLP-дослідженнях. Приклад коду для Twitter в лістингу 3.1.

Лістинг 3.1 – Приклад отримання постів з Twitter

```
import tweepy

api_key = 'YOUR_API_KEY'
api_secret = 'YOUR_API_SECRET'
access_token = 'YOUR_ACCESS_TOKEN'
access_secret = 'YOUR_ACCESS_SECRET'

auth = tweepy.OAuth1UserHandler(api_key, api_secret, access_token,
access_secret)
api = tweepy.API(auth)

username = "elonmusk"
tweets = api.user_timeline(screen_name=username, count=5,
tweet_mode="extended")

# Output posts
for tweet in tweets:
print(f"{tweet.created_at} — {tweet.full_text}\n{'-'*50}")
```

Reddit API дозволяє отримувати структуровані дані з різно манітних тематичних підфорумів (subreddit) за допомогою бібліотеки PRAW (Python Reddit API Wrapper). Можна збирати пости, коментарі, кількість голосів, час публікації та інші параметри. Reddit є цінним джерелом, особливоу сфері ментального здоров’я, оскільки користувачі активно діляться власними

переживаннями та досвідом у відповідних спільнотах. Приклад коду для Reddit в лістингу 3.2.

Лістинг 3.2 – Приклад отримання постів з Reddit

```
import praw

# Reddit API settings
reddit = praw.Reddit(
client_id='YOUR_CLIENT_ID',
client_secret='YOUR_CLIENT_SECRET',
user_agent='your_app_name'
username = "spez" # example user
user = reddit.redditor(username)

for post in user.submissions.new(limit=5):
print(f"{post.created_utc} — {post.title}")
print(post.selftext[:300] if post.selftext else "[No text]")
print('-' * 60)
```

Instagram Graph API надає доступ довмістуакаунтів бізнес-категорії та авторів контенту. API дозволяє отримувати пости, підписи, коментарі, згадки та аналітичні дані, але лише за умови попередньої авторизації та відповідності політиці конфіденційності Meta. У дослідницьких цілях найчастіше аналізують публічні пости з популярних хештегів або сторінок, що стосуються соціальних і психологічних тем. Приклад коду для Instagram в лістингу 3.3.

Лістинг 3.3 – Приклад отримання постів з Instagram

```
import requests1
access_token = 'YOUR_ACCESS_TOKEN'
user_id = 'user_id'
url = f"https://graph.facebook.com/v19.0/{user_id}/media"
params = {
    'fields': 'id,caption,media_type,media_url,timestamp,permalink',
    'access_token': access_token
}
response = requests.get(url, params=params)
print(response.json())
```

Існуючі API забезпечують досить широкі можливості для збору текстових даних із соціальних мереж. Найбільш зручними і відкритими для наукового аналізу є Twitter API та Reddit API. Інші сервіси, зокрема Facebook і Instagram, мають обмеження, пов’язані з конфіденційністю, що варто враховувати під час побудови систем обробки користувацьких повідомлень.

4 ОБРОБКА ДАНИХ

4.1 Використання датасету

Датасет - це впорядкований та структурований набір даних, який виступає як основа для навчання, тестування та оцінювання алгоритмів штучного інтелекту. Це основний елемент у створенні систем штучного інтелекту, оскільки якість, різноманітність та кількість даних безпосередньо покладаються на ефективність та точність алгоритмів. Добре структурований набір даних пропонує можливість вивчити тенденції, визначити моделі та зробити точні прогнози для нової, небаченої інформації.

Датасет може містити різні види даних, такі як письмові файли, зображення, звукозаписи, рухомі зображення, цифри, послідовності з часом та згрупована інформація. Їх можна показати як таблиці, групи файлів, послідовності або інші організовані структури. Для будь -якого виду роботи в ШІ вибираються дати, що показують риси та деталі проблеми. Наприклад, письмові дати з групами робіт, розмов або обмінів, позначених, щоб показати, що вони належать до певних категорій або мають особливі характеристики, використовуються в мовному аналізі. Завдання комп'ютерного зору використовують мітки, які допомагають моделям у визначенні предметів, їх розташування чи типах. У звуковому дослідженні вони є нотами шумів або розмовляють з відповідними символами [7].

У роботі з пошуку посттравматичного стресового розладу важливо використовувати найвищий набір тексту з повідомленнями з соціальних медіа, форумів, блогів чи інших місць. Кожен документ поставляється з тегом, що показує, якщо знаки ПТСР є чи ні, допомагаючи системі навчитися відрізняти важливі ознаки цього стану. Крім того, під час обробки набору даних розглянемо такі фактори, як розподіл класу, невідповідності даних, обробка неповних або неправильних записів та початкові кроки, такі як очищення

тексту, стандартизація, сегментація слів та виведення виноски. На рисунку 4.1 зображено вигляд датасету.

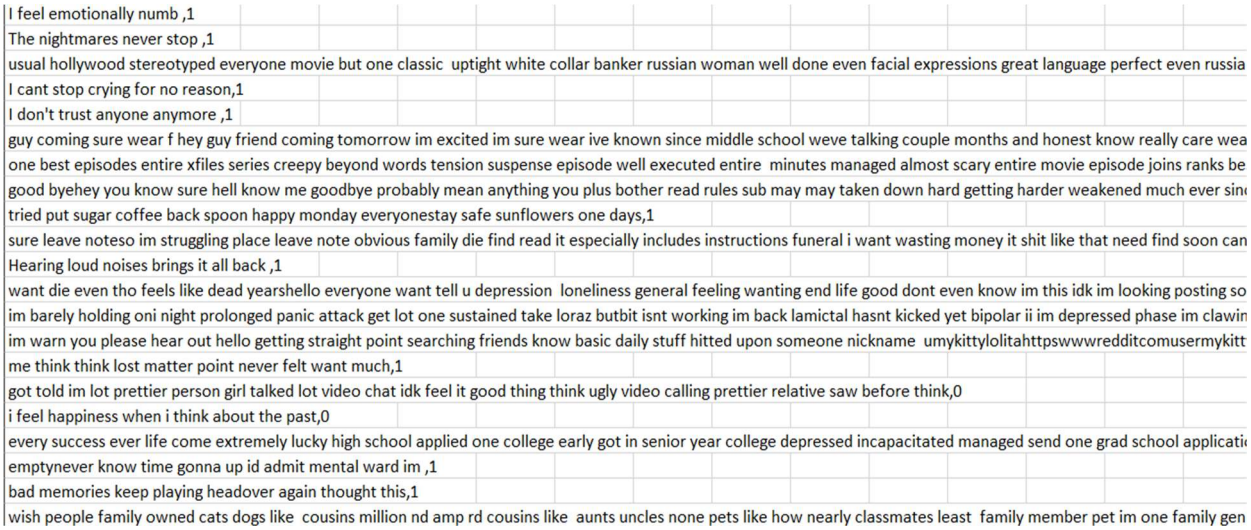
I feel emotionally numb ,1
The nightmares never stop ,1
usual hollywood stereotyped everyone movie but one classic uptight white collar banker russian woman well done even facial expressions great language perfect even russia
I cant stop crying for no reason,1
I don't trust anyone anymore ,1
guy coming sure wear f hey guy friend coming tomorrow im excited im sure wear ive known since middle school weve talking couple months and honest know really care wea
one best episodes entire xfiles series creepy beyond words tension suspense episode well executed entire minutes managed almost scary entire movie episode joins ranks be
good byehey you know sure hell know me goodbye probably mean anything you plus bother read rules sub may may taken down hard getting harder weakened much ever sin
tried put sugar coffee back spoon happy monday everyonestay safe sunflowers one days,1
sure leave noteso im struggling place leave note obvious family die find read it especially includes instructions funeral i want wasting money it shit like that need find soon can
Hearing loud noises brings it all back ,1
want die even tho feels like dead yearshello everyone want tell u depression loneliness general feeling wanting end life good dont even know im this idk im looking posting so
im barely holding oni night prolonged panic attack get lot one sustained take loraz butbit isnt working im back lamictal hasnt kicked yet bipolar ii im depressed phase im clawin
im warn you please hear out hello getting straight point searching friends know basic daily stuff hitted upon someone nickname umykittylolitahttpswwwredditcomusermykitt
me think think lost matter point never felt want much,1
got told im lot prettier person girl talked lot video chat idk feel it good thing think ugly video calling prettier relative saw before think,0
i feel happiness when i think about the past,0
every success ever life come extremely lucky high school applied one college early got in senior year college depressed incapacitated managed send one grad school applicati
emptynever know time gonna up id admit mental ward im ,1
bad memories keep playing headover again thought this,1
wish people family owned cats dogs like cousins million nd amp rd cousins like aunts uncles none pets like how nearly classmates least family member pet im one family gen

Рисунок 4.1 – Зображення датасету

Дані проходили етапи попередньої обробки, включаючи очищення від зайвих символів, виправлення орфографічних помилок і видалення нерелевантних записів. Потім дані були поділені на навчальну, валідаційну та тестову вибірки, щоб забезпечити надійність моделі. Модель навчалася на основі закономірностей у текстах, після чого оцінювалась її ефективність на нових даних.

Датасет, використаний у дослідженні, відіграв важливу роль у досягненні точності та надійності моделі для виявлення ПТСР.

4.2 Обробка текстових даних

Обробка тексту є важливим етапом у підготовці даних для систем машинного навчання, оскільки саме вона забезпечує лексичну та семантичну чистоту вхідної інформації, збільшує її якість, однорідність і структурованість, а також формує основу для подальшого ефективного аналізу та навчання моделей. Цей процес дозволяє видалити шум, нормалізувати лінгвістичні

властивості та зменшити кількість зайвих або повторюваних елементів, які можуть негативно впливати на точність класифікації.На початковому етапі завантажується файл CSV, який містить текстову інформацію. Завантаження здійснюється з використанням бібліотеки pandas, що дозволяє ефективно працювати з таблицями даних.

Далі проводиться обробка числових даних у стовпцях, які можуть містити пропущені значення або нескінченності. Усі ці проблеми вирішуються шляхом заміни пропусків на нулі та округлення чисел до цілих значень.

Код представлений у лістингу 4.1 відповідає за завантаження датасету.

Лістинг 4.1 – Завантаження датасету

```
data = pd.read_csv('../input/dataset-pstd/mental_health1.csv')
data.iloc[:, 1] = data.iloc[:, 1].fillna(0).replace([np.inf, -np.inf], 0).round().astype(int)
```

Додавання нових характеристик тексту: кількості слів та символів у кожному тексті. Ці метрики корисні для аналізу довжини тексту та можуть бути використані моделлю як ознаки (лістинг 4.2).

Лістинг 4.2 – Додавання нових характеристик

```
data.isnull().sum()
data['label'].value_counts()
```

На початковому етапі текст піддається базовій нормалізації, що включає перетворення всіх знаків до нижнього регістру. Це дозволяє уникнути повторення слів, які різняться лише регістром, наприклад, "Stress" і "stress", що в лінгвістичному сенсі є тотожними, але можуть сприйматися як різні об'єкти під час векторизації. Зниження регістру значно зменшує розмір словника й сприяє моделі зосередитись на змістовному аналізі без зайвого ускладнення простору ознак (лістинг 4.3). Окрім цього, з текстів вилучаються URL-адреси, які, як правило, не несуть корисної інформації для семантичного аналізу. Посилання часто є технічним змістом, що не має безпосереднього відношення до психологічного стану автора повідомлення. Видалення таких

елементів дозволяє знизити рівень шуму в даних та покращити якість ознак, що передаються до моделі.

Лістинг 4.3 – Код для обробки даних

```
def convert_lowercase(text):
    return text.lower()
data['text'] = data['text'].apply(convert_lowercase)

def remove_url(text):
    re_url = re.compile('https?://\S+|www\.\S+')
    return re_url.sub('', text)
data['text'] = data['text'].apply(remove_url)
```

Також текст очищується від пунктуації, яка може заважати моделі знаходити значущі закономірності в тексті. Видалення стоп-слів є наступним важливим етапом, під час якого з тексту вилучаються слова, що часто вживаються, але не містять специфічної інформації, наприклад «the», «and» та «is» (лістинг 4.4).

Лістинг 4.4 – Код для видалення зайвого в даних

```
exclude = string.punct1uation
def remove_punc(text):
    return text.translate(str.maketrans('', '', exclude))
data['text'] = data['text'].apply(remove_punc)
def remove_stopwords(text):
    new_list = []
    words = word_tokenize(text)
    stopwrds = stopwords.words('english')
    for word in words:
        if word not in stopwrds:
            new_list.append(word)
    return ' '.join(new_list)

data['text'] = data['text'].apply(remove_stopwords)
```

Додатково проводиться стемінг тексту, тобто приведення слів до їхньої основної форми. Це зменшує варіативність слів і дозволяє моделі ефективніше знаходити схожі слова в тексті. Для цього використовується алгоритм Porter Stemmer (лістинг 4.5).

Лістинг 4.5 – Код для преведення слов в основну форму

```
def perform_stemming(text):
    stemmer = PorterStemmer()
    new_list = []
    words = word_tokenize(text)
    for word in words:
        new_list.append(stemmer.stem(word))
    return " ".join(new_list)
data['text'] = data['text'].apply(perform_stemming)
```

Усі ці кроки є необхідними для стандартизації текстових даних і підвищення ефективності роботи моделі машинного навчання. Завдяки цьому підходу дані стають більш структурованими, що полегшує навчання моделі та покращує її здатність до точного прогнозування. Далі для наглядності оброблено слова з датасету в якому фрагмент коду аналізує текстові дані, щоб виявити слова, які найчастіше зустрічаються у постах. Для цього використовується набір текстів, де є текст та мітка. Для позначення текстів з ПТСР стоїть «1». Спочатку всі текстові повідомлення, що відповідають цій категорії, розбиваються на окремі слова за допомогою методу split. На наступному етапі обчислюється частота кожного слова зі списку за допомогою модуля Counter бібліотеки collections. Результати процесу показують 25 найпоширеніших слів, що зустрічаються в повідомленнях. Дані структуровано у форматі DataFrame за допомогою бібліотеки pandas для полегшення аналізу та візуалізації (лістинг 4.6).

Лістинг 4.6 – Код для преведення слов в основну форму

```
words = []
sentences = data[data['label'] == 1]['text']
for text in sentences:
    words.extend(text.split())

word_counts = Counter(words).most_common(25)
word_frequency_df = pd.DataFrame(word_counts, columns=['Word', 'Count'])
sns.set_context('notebook', font_scale=1.3)
plt.figure(figsize=(18, 8))
sns.barplot(data=word_frequency_df,         x='Count',         y='Word',
palette='coolwarm')
plt.title("Most Common Used Words")
plt.xlabel("Frequency")
plt.ylabel("Words")
plt.tight_layout()
plt.show()
```

Завершальним етапом є візуалізація даних: за допомогою бібліотеки seaborn створюється горизонтальна гістограма, яка показує найчастіші слова та їхню кількість. Цей графік дає чітке уявлення про те, які слова найчастіше зустрічаються в текстах, що важливо для подальшого використання. Ефективна структура та візуалізація робить результати більш зрозумілими та легкими для інтерпретації (рисунок 4.2).

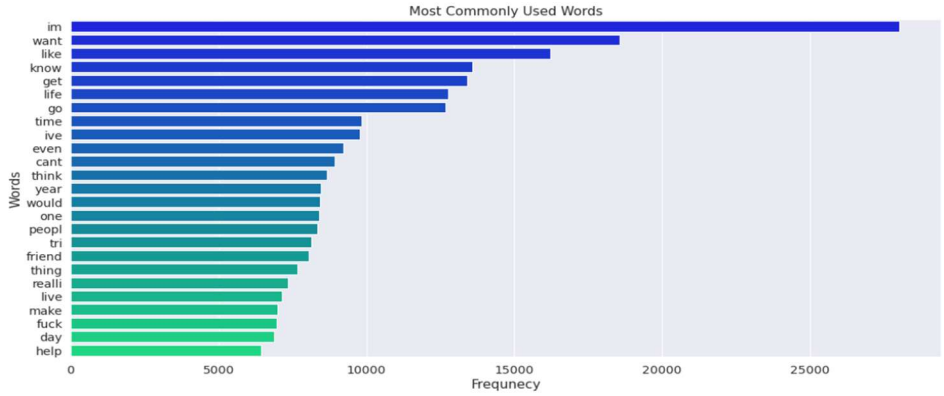

Рисунок 4.2 – Зображення найуживаніших слів в текстах з ПТСР

Необхідно виконати підготовку текстових даних для задачі класифікації, зокрема фільтрації, шляхом векторизації тексту та розділення даних на навчальну і тестову вибірки. Спочатку текстові дані і мітки класів витягуються із заданого набору даних. Потім виконується поділ даних на навчальну і тестову вибірки за допомогою функції train_test_split з бібліотеки scikit-learn. Далі перевіряється тип даних для навчальної та тестової вибірок. У разі, якщо дані представлені як масиви NumPy, вони перетворюються на списки для подальшої обробки. Це забезпечує сумісність із векторизатором, який працює з текстовими даними (лістинг 4.7).

Лістинг 4.7 – Підготовка кода для класифікації

```
X = data["text"]
y = data['label'].values
# Split data into training and test sets while preserving class balance
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2, random_state= 42, stratify = y)
```

5 ПРОЦЕС МАШИННОГО НАВЧАННЯ

5.1 Наївний класифікатор Байєса

5.1.1 Теоритичні відомості про наївний класифікатор Байєса

Наївні класифікатори Байєса — це група алгоритмів, що ґрунтуються на теоремі Байєса. Незважаючи на "наївне" припущення про незалежність ознак одна від одної, вони набули широкого застосування завдяки простоті реалізації та високій ефективності під час навчання. Важливо зазначити, що наївні класифікатори Байєса — не один конкретний алгоритм, а ціле сімейство методів класифікації, об'єднаних спільним принципом: кожна пара ознак вважається статистично незалежною. Один із найпростіших і найефективніших алгоритмів класифікації, наївний класифікатор Байєса, допомагає швидко розробляти моделі машинного навчання з можливостями швидкого прогнозування.

Для задач класифікації часто використовується наївний байєсівський алгоритм. Він широко застосовується для аналізу текстів, особливо у випадках, коли дані мають високу багатовимірність (кожне слово виступає як окрема ознака). Цей підхід знаходить застосування у фільтрації повідомлень, аналізі настроїв, рейтинговій класифікації та інших подібних завданнях. Основною перевагою цього методу є його ефективність і швидкість. Він дозволяє легко й оперативно здійснювати прогнози навіть із великими обсягами даних. Ця модель оцінює імовірність належності зразка до певного класу на основі заданого набору ознак. Вона є імовірнісним класифікатором, оскільки припускається, що кожна ознака в моделі функціонує незалежно від інших. Іншими словами, кожна ознака робить свій внесок у прогнозування окремо, без взаємодії з іншими. У реальності така умова виконується рідко. Алгоритм моделі базується на використанні теореми Байєса для навчання та

прогнозування [8].

У бібліотеці scikit-learn реалізовано кілька варіантів наївного байєсівського класифікатора, які відрізняються припущеннями про тип розподілу ознак при заданому класі. Серед них:

– Гаусівський наївний байєсівський класифікатор (GaussianNB). Використовується для неперервних числових ознак, які припускаються нормально розподіленими. Підходить для задач, де значення ознак мають природну варіативність (наприклад, температура або ріст);

– Мультиноміальний наївний байєсівський класифікатор (MultinomialNB). Застосовується для дискретних ознак, зокрема в задачах класифікації текстів, де ознаки — це кількість появ слів. Добре працює з «мішком слів» та частотними даними;

– Комплементарний наївний байєсівський класифікатор (ComplementNB). Це вдосконалена версія мультиноміального варіанту, яка краще справляється з незбалансованими наборами даних. Вона враховує не лише наявність ознак у класі, але й їх відсутність у решті класів, що підвищує точність класифікації при великій диспропорції класів;

– Бернуллівський наївний байєсівський класифікатор (BernoulliNB). Працює з бінарними (0 або 1) ознаками, які вказують на наявність або відсутність певної характеристики. Поширене застосування — текстові задачі, де ознаки позначають наявність конкретних слів у документі;

– Категоріальний наївний байєсівський класифікатор (CategoricalNB). Призначений для роботи з категоріальними ознаками, що приймають обмежений набір значень (наприклад, колір, регіон). Кожна ознака має власне категоріальне розподілення, і модель враховує це під час навчання.

Для задач виявлення ПТСР у текстах особливо добре підходить мультиноміальний наївний байєсівський класифікатор**.** Це пов’язано з тим, що такий тип класифікатора ефективно працює з текстовими даними, де ознаками виступають частоти слів або термінів. Саме така форма подання інформації є типовою для задач обробки природної мови, зокрема при аналізі повідомлень,

постів чи опитувань, пов’язаних із проявами ПТСР. Імовірність ознаки при заданому класі обчислюється за формулою 5.1.

$$P(x_i|y) = \frac{N_{yi} + \alpha}{N_y + \alpha n} \quad (5.1)$$

де Nyi — кількість появ ознаки і у класі y;
Ny — загальна кількість усіх ознак у класі y;
n — кількість різних ознак;
α — параметр згладжування, який запобігає появі нульових ймовірностей.

Таким чином, наївні класифікатори Байєса, завдяки своїй простоті, швидкодії та здатності ефективно працювати з високорозмірними текстовими даними, є цінним інструментом у задачах класифікації, зокрема при виявленні ПТСР. Серед різновидів таких класифікаторів особливе значення має мультиноміальний наївний байєсівський класифікатор, який оптимально підходить для обробки текстової інформації у вигляді «мішка слів». Його здатність працювати з частотними ознаками робить його ефективним у виявленні мовленнєвих патернів, характерних для проявів психоемоційних станів [9].

Попри численні переваги, наївні класифікатори Байєса мають і певні недоліки, які варто враховувати при їх застосуванні. Основним обмеженням є наївне припущення про незалежність ознак, яке рідко виконується в реальних даних. У разі порушення цієї умови модель може генерувати некоректні оцінки ймовірностей, що призводить до зниження точності класифікації. Крім того, наївні байєсівські класифікатори мають тенденцію віддавати перевагу класам із більшою кількістю прикладів у навчальній вибірці. Це може призвести до зміщення в бік домінантного класу у випадку незбалансованих даних, що є критично важливим при роботі з чутливими задачами, як-от виявлення ПТСР, де кількість прикладів може бути суттєво різною для

кожного класу. У таких випадках доцільно застосовувати методи балансування вибірки або модифіковані варіанти алгоритму.

5.1.2 Машинне навчання за класифікатором Байєса

Перед подачею текстових даних у модель машинного навчання їх необхідно перетворити у числовий формат. Для цього застосовується метод TF-IDF (Term Frequency–Inverse Document Frequency) — один з найефективніших способів векторизації текстів. Він дозволяє оцінити важливість термінів у документі відносно всього корпусу.TF-IDF поєднує дві метрики:

– TF (term frequency) — частота появи терміна в окремому документі;

– IDF (inverse document frequency) — обернена частота документів, у яких присутній термін. Зменшує вагу термінів, які трапляються часто в багатьох документах.

Таким чином, слова, які зустрічаються часто в одному документі, але рідко в інших, отримують вищу вагу, що допомагає моделі сфокусуватись на унікальних лексичних одиницях.

У Python реалізація векторизації здійснюється за допомогою класу TfidfVectorizer з бібліотеки scikit-learn. Перш ніж перейти до векторизації, потрібно переконатися, що вхідні дані мають формат списку рядків, оскільки деякі функції можуть повернути масив NumPy (ndarray), що призводить до помилок при обробці текстів. Якщо це так, масиви конвертуються назад у список рядків. (лістинг 5.1).

Лістинг 5.1 – Використання метода TfidfVectorizer

```
if isinstance(X_train, np.ndarray):
    X_train = X_train.tolist()
if isinstance(X_test, np.ndarray):
    X_test = X_test.tolist()

# Apply TfidfVectorizer
tfidf = TfidfVectorizer(max_features=2500, min_df=2)
X_train = tfidf.fit_transform(X_train).toarray()
X_test = tfidf.transform(X_test).toarray()
```

Далі необхідно виконати функцію для навчання та оцінки класифікаційної моделі з подальшою візуалізацією результатів. Модель навчається на попередньо підготовлених даних за допомогою методу fit, після чого на основі тестових даних проводиться прогнозування міток класів за допомогою методу predict і ймовірностей класів. Для оцінки моделі використовуються такі ключові метрики:

- точність (Accuracy): відображає частку правильних передбачень від загальної кількості;

- прецизія (Precision): оцінює, яку частку з передбачень є дійсно правильними;

- повнота (Recall): оцінює, яку частку реальних повідомлень модель змогла правильно визначити. Метрики округлюються до трьох знаків після коми для зручності.

Ці метрики дозволяють зрозуміти, наскільки ефективно модель класифікує дані, зокрема правильність передбачення повідомлень.

Додатково результати моделі візуалізуються для наочності. Матриця невідповідностей, створена за допомогою функції plot_confusion_matrix, відображає точність і кількість помилкових передбачень для кожного класу, що допомагає оцінити розподіл хибнопозитивних і хибнонегативних прогнозів. Крива ROC демонструє співвідношення між часткою хибнопозитивних передбачень і повнотою, що дозволяє оцінити загальну якість моделі.

У цьому прикладі використовується наївний Баєсовий класифікатор MultinomialNB, що добре підходить для текстових даних, враховуючи частоти термінів у документах. Модель навчається на навчальній вибірці, після чого її продуктивність тестується на відкладених даних. Підсумком виконання є числові значення метрик продуктивності разом із графічним представленням результатів, що дає змогу отримати повне уявлення про здатність моделі до класифікації. Код для функції train_model представлений в лістингу 5.2.

Лістинг 5.2 – Тренування моделі

```
def train_model(model):
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    probabilities = model.predict_proba(X_test)
    acc_score = round(accuracy_score(y_test, predictions), 3)
    prec_score = round(precision_score(y_test, predictions), 3)
    rec_score = round(recall_score(y_test, predictions), 3)

    print(f'Model Accuracy: {acc_score}')
    print(f'Model Precision: {prec_score}')
    print(f'Model Recall: {rec_score}')

    sns.set_context("talk", font_scale=1.4)
    fig, ax = plt.subplots(1, 2, figsize=(20, 8))
    ax1 = plot_confusion_matrix(y_test, predictions, ax= ax[0], cmap=
'coolwarm')
    ax2 = plot_roc(y_test, probabilities, ax= ax[1], plot_macro= False,
plot_micro= False, cmap= 'autumn')
nb = MultinomialNB()
train_model(nb)
```

На рисунку 5.1 зображено матрицю та ROC криву

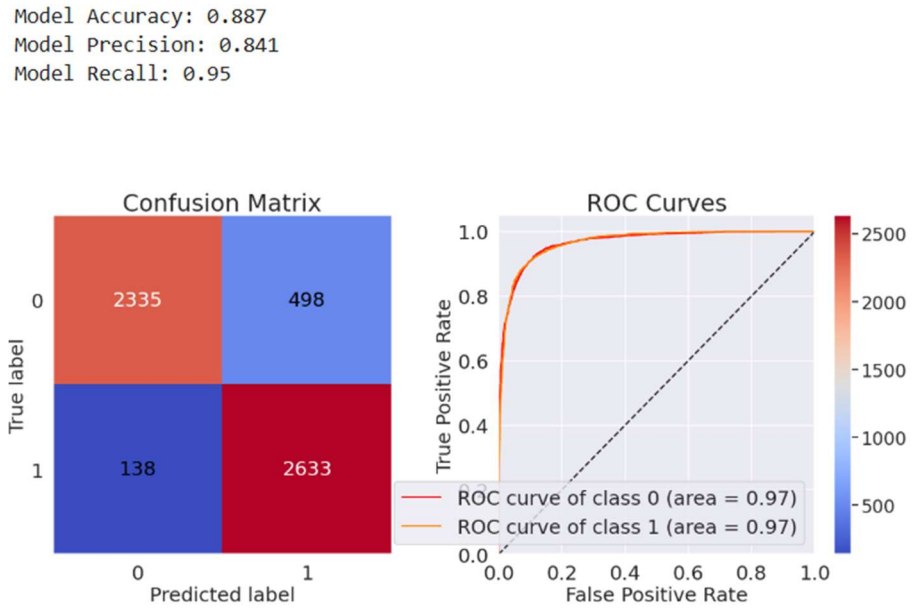

Рисунок 5.1 – Матриця невідповідностей та крива ROC

Код здійснює автоматизований аналіз текстових публікацій користувача для виявлення ознак посттравматичного стресового розладу (ПТСР). Після виводу повідомлення про початок аналізу постів користувача u/{username},

система проходиться по 50 останніх публікаціях типу selftext. Для кожного посту, що містить текст, викликається функція predict_text, яка класифікує його як "PTSD" або "Non-PTSD" на основі навченої моделі Наївного Баєса.

Якщо у пості виявлено ознаки ПТСР, значення лічильника ptsd_count збільшується. У консоль виводиться номер поста, перші 300 символів його тексту та результат класифікації. Після завершення обробки всіх дописів розраховується співвідношення постів з ознаками ПТСР до загальної кількості проаналізованих. Якщо не вдалося знайти жодного текстового посту, виводиться повідомлення про відсутність даних для аналізу. Якщо частка позитивних класифікацій становить менше 25%, виводиться повідомлення про відсутність ПТСР (лістинг 5.3). Інакше повідомляється про можливу наявність психологічних проблем. Цей код дозволяє швидко оцінити психоемоційний стан користувача на основі відкритих текстів, що дає змогу застосувати алгоритми машинного навчання в соціальному моніторингу.

Лістинг 5.3 – Оцінка постів на ПТСР

```
      # Using Naive Bayers method
      print(f"\n--- Analyze user's posts u/{username} --- using Naive
Bayers\n")

for i, post in enumerate(user.submissions.new(limit=50), start=1):
    text = post.selftext.strip()
    if not text:
        continue
    total_checked += 1
    prediction = predict_text(text)
    if prediction == "PTSD":
        ptsd_count += 1

      print(f"Post #{i}")
    print("Text      :", text[:300])
    print("Prediction:", prediction)
    print("-" * 60)

# Check for all posts
if total_checked == 0:
    print("There are no text posts for analysis.")
else:
    ptsd_ratio = ptsd_count / total_checked
    if ptsd_ratio < 0.25:
        print(f"\nResult: PTSD not detected.")
    else:
        print(f"\nResult: Possible signs of PTSD.")
```

Для зразка було взято акаунт u/GovSchwarzenegger, дописи якого було проаналізовано на виявлення ознак посттравматичного стресового розладу (ПТСР). Після старту скрипту виводиться сповіщення про початок перевірки. Кожен із наявних текстових постів користувача обробляється за допомогою функції predict_text, котра видає передбачений клас — "Positive" (висока вірогідність ПТСР) або "Negative" (низька ймовірність), а також відповідну ймовірність класифікації.

У розглянутому прикладі всі повідомлення були класифіковані як "Negative", з імовірністю наявності ПТСР в межах від ~11% до ~43%. Це свідчить про відсутність ознак тривожності, пригніченого емоційного стану чи небезпечних висловлювань. Зміст публікацій здебільшого має позитивний, мотиваційний чи інформативний характер: користувач ділиться новинами, запрошує до бесіди, презентує власні ініціативи й заохочує підписників до участі. На закінчення аналізу система видає висновок, що ПТСР не виявлено, оскільки жоден із постів не класифіковано як такий, що містить потенційні ознаки психологічного розладу (рисунок 5.2). Це демонструє стабільність емоційного фону автора контенту та ефективність запропонованої моделі у визначенні психоемоційного стану на основі тексту.

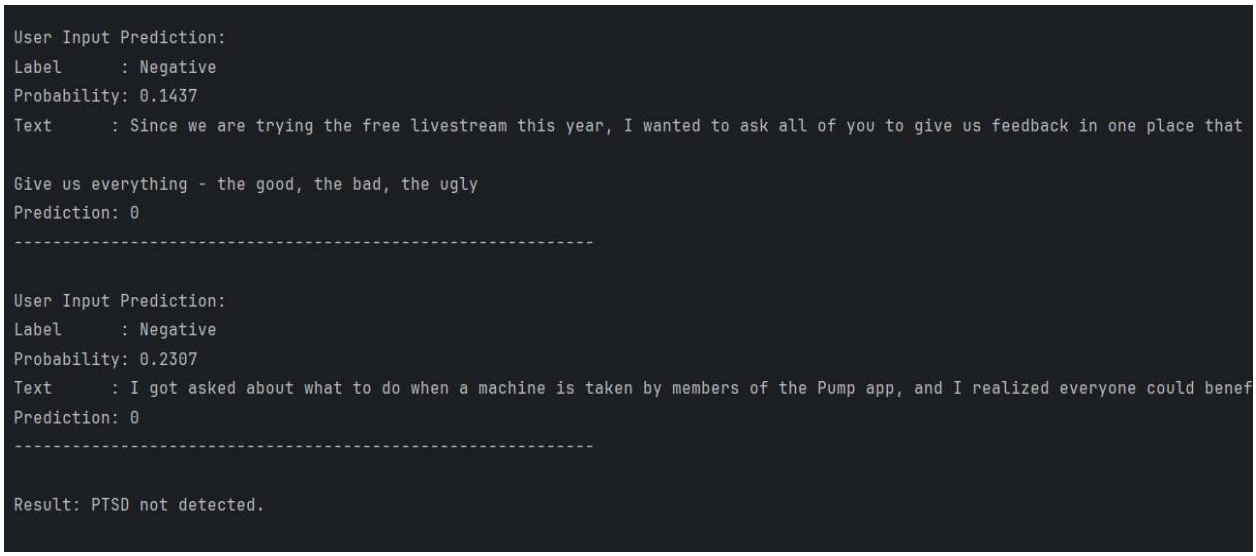

Рисунок 5.2 – Результати роботи: негативний результат ПТСР

Для порівняння також було проаналізовано інший акаунт, у якому виявлено повідомлення з ознаками ПТСР. В результаті система класифікувала контент як такий, що містить потенційні прояви психологічного розладу (рисунок 5.3).

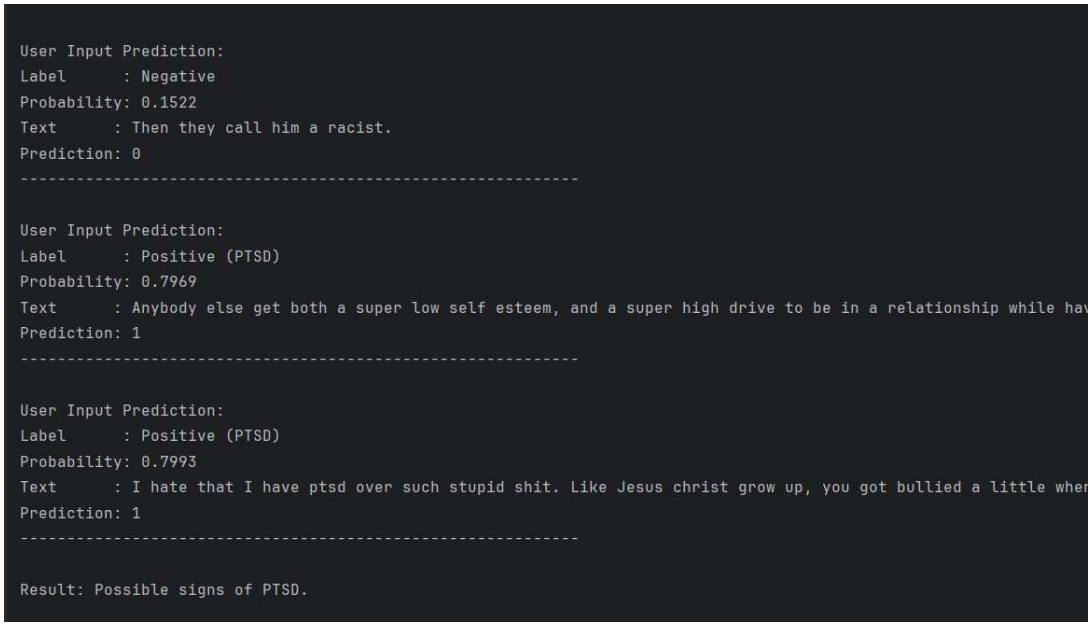

Рисунок 5.3 – Результати роботи: позитивний результат ПТСР

5.2 LSTM

5.2.1 Теоритичні відомості про наївний класифікатор LSTM

Довга короткочасна пам’ять (LSTM) — це тип архітектури штучної рекурентної нейронної мережі (RNN), яка широко застосовується в галузі глибокого навчання. Вона була спеціально розроблена для ефективного опрацювання та прогнозування даних, що мають часову або послідовну природу, таких як текст, аудіо або часові ряди. На відміну від звичайних нейронних мереж прямого поширення (feedforward networks), LSTM мають зворотні зв’язки, що дозволяє їм зберігати та використовувати інформацію про попередні елементи послідовності під час обробки нових даних.

Однією з головних проблем, з якими стикаються традиційні RNN, є зникнення або розрив градієнтів під час тривалого навчання, що ускладнює запам’ятовування довготривалих залежностей у даних. LSTM вирішує цю проблему завдяки особливій структурі комірки пам’яті, яка дозволяє зберігати інформацію протягом тривалого часу та ефективно керувати нею за допомогою спеціалізованих механізмів — вентилів (або шлюзів).

Кожна комірка LSTM містить три основні вентилі вхідний вентиль, вентиль забування та вихідний вентиль.

- вхідний вентиль визначає, яка частина нової інформації має бути додана до комірки пам’яті. Це дозволяє мережі навчатись на нових даних, не втрачаючи важливої інформації з попередніх кроків;

- вентиль забування регулює, яка частина збереженої інформації повинна бути видалена або збережена. Він надзвичайно важливий для контролю обсягу пам’яті та уникнення накопичення зайвих або застарілих даних;

- вихідний вентиль вирішує, яка частина вмісту комірки пам’яті буде використана для формування прихованого стану (hidden state), який передається далі по мережі.

Завдяки цим механізмам, LSTM можуть динамічно адаптуватися до різних типів послідовних залежностей, вибірково зберігаючи корисну інформацію та ігноруючи нерелевантну. Це дозволяє досягати високої точності в таких задачах, як розпізнавання мовлення, машинний переклад, генерація тексту, класифікація настроїв і прогнозування часових рядів [10].

Структура мережі LSTM складається з послідовного з’єднання LSTM-комірок, де кожна обробляє певний часовий крок вхідної послідовності. Вихід однієї комірки передається як вхід до наступної, формуючи потік обробки, який охоплює всю послідовність. Комірка пам’яті всередині кожної LSTM-одиниці дозволяє зберігати контекст попередніх часових кроків, що є критичним для аналізу довготривалих залежностей (рисунок 5.4).

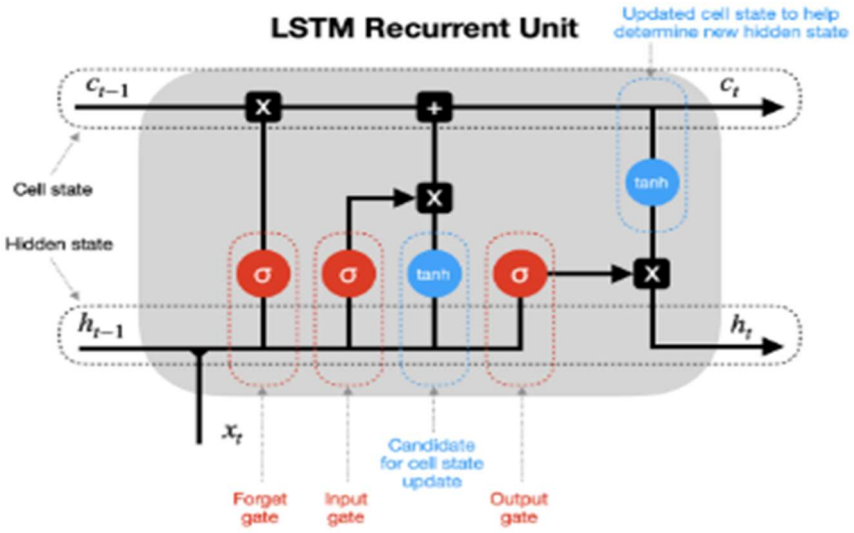

Рисунок 5.4 - Структура мережі LSTM

Таким чином, мережі LSTM є потужним інструментом для обробки складних часових структур, оскільки поєднують гнучкість нейронних мереж із можливістю запам’ятовування важливих подій у часі. Їх ефективність зробила LSTM стандартом де-факто в багатьох практичних застосуваннях, пов’язаних із послідовною обробкою інформації.

5.2.2 Машинне навчання за допомогою LSTM

Для початку необхідно почати з підготовки текстових даних, яка включає токенізацію та додавання заповнень. Вхідний текст перетворюється на послідовності цілих чисел за допомогою токенізатора, який враховує до 5000 найпоширеніших слів. Текст переводиться в нижній регістр і розбивається на слова за пробілами. Потім послідовності вирівнюються до однакової довжини (100 символів) за допомогою заповнень наприкінці. Функція повертає готові послідовності та об'єкт токенізатора для подальшого використання (лістинг 5.4).

Лістинг 5.4 – Код для токенізації

```
max_words = 5000
max_length = 100

def preprocess_text(input_text):
    """
    This function tokenizes the input text into sequences of integers
    and pads each sequence to a uniform length.
    """
    # Create a tokenizer with the specified number of words
    text_tokenizer=Tokenizer(num_words=max_words, lower=True, split=' ')
    # Fit the tokenizer on the input text
    text_tokenizer.fit_on_texts(input_text)
    # Convert text into sequences of integers
    sequences = text_tokenizer.texts_to_sequences(input_text)
    # Pad the sequences to ensure uniform length
    padded_sequences    =    pad_sequences(sequences,    padding='post',
maxlen=max_length)
    # Return the padded sequences and the tokenizer
    return padded_sequences, text_tokenizer
```

Далі потрібно підготовити нейронну мережу до навчання, налаштовуючи її функцію втрат, оптимізатор і метрики оцінки. Функція втрат binary_crossentropy використовується для бінарної класифікації, що допомагає моделі навчитися зменшувати розбіжності між передбаченими й реальними мітками. Оптимізатор SGD (Стохастичний градієнтний спуск) оновлює ваги моделі під час навчання, сприяючи поступовому зменшенню втрат. Метрики включають Recall, який оцінює точність передбачення позитивних випадків, та accuracy, яка вимірює загальну точність моделі. (лістинг 5.5).

Лістинг 5.5 – Код для оптимізації оцінки

```
model.compile(loss='binary_crossentropy',optimizer='SGD',
metrics=[tf.keras.metrics.Recall(),'accuracy'])
```

Далі потрібно налаштувати процес навчання нейронної мережі з використанням ранньої зупинки та визначення розміру пакетів даних. Об'єкт EarlyStopping з параметрами monitor=val_loss і patience=5 дозволяє зупинити навчання, якщо валідаційні втрати не зменшуються протягом п’яти епох поспіль. Це допомагає уникнути перенавчання та економить обчислювальні ресурси.

Навчання моделі виконується методом model.fit, де передаються тренувальні дані X_trn і y_trn, а також валідаційні дані X_vld і y_vld. Параметр

batch_size визначає, що модель оновлюватиме ваги після обробки кожного пакета з 64 прикладів. Загальна кількість епох задається змінною epochs, і процес виводить прогрес навчання завдяки параметру verbose. Додатково у параметрі callbacks передається об'єкт ранньої зупинки es, що контролює процес навчання. Отримана історія навчання зберігається в змінній history для подальшого аналізу (лістинг 5.6).

Лістинг 5.6 – Код для навчання моделі

```
es = EarlyStopping(monitor = 'val_loss', patience=5)
batch_size = 64
history = model.fit(X_trn, y_trn,
                    validation_data=(X_vld, y_vld),
                    batch_size=batch_size, epochs=epochs, verbose=1,
                    callbacks = [es])
```

Процес навчання зображено на рисунку 5.5

```
Epoch 1/50
245/245 [==============================] - 31s 108ms/step - loss: 0.6925 - recall: 0.2969 - accuracy: 0.5255 - val_loss: 0.6908 - val_recall: 0.3634 - val_accuracy: 0.61
67
Epoch 2/50
245/245 [==============================] - 23s 95ms/step - loss: 0.6897 - recall: 0.4451 - accuracy: 0.6027 - val_loss: 0.6880 - val_recall: 0.3300 - val_accuracy: 0.610
7
Epoch 3/50
245/245 [==============================] - 24s 99ms/step - loss: 0.6866 - recall: 0.4145 - accuracy: 0.6175 - val_loss: 0.6843 - val_recall: 0.5980 - val_accuracy: 0.672
8
Epoch 4/50
245/245 [==============================] - 24s 99ms/step - loss: 0.6817 - recall: 0.5485 - accuracy: 0.6389 - val_loss: 0.6775 - val_recall: 0.3884 - val_accuracy: 0.626
8
Epoch 5/50
245/245 [==============================] - 24s 98ms/step - loss: 0.6718 - recall: 0.5429 - accuracy: 0.6519 - val_loss: 0.6622 - val_recall: 0.7170 - val_accuracy: 0.679
8
Epoch 6/50
245/245 [==============================] - 23s 94ms/step - loss: 0.6470 - recall: 0.6540 - accuracy: 0.6674 - val_loss: 0.6279 - val_recall: 0.6787 - val_accuracy: 0.672
8
Epoch 7/50
245/245 [==============================] - 24s 98ms/step - loss: 0.6204 - recall: 0.6646 - accuracy: 0.6731 - val_loss: 0.6148 - val_recall: 0.6763 - val_accuracy: 0.671
6
Epoch 8/50
245/245 [==============================] - 24s 97ms/step - loss: 0.6158 - recall: 0.6726 - accuracy: 0.6736 - val_loss: 0.6123 - val_recall: 0.6748 - val_accuracy: 0.672
5
```

Рисунок 5.5 – Процес навчання моделі

Для перевірки навчання додано візуалізацію процесу навчання моделі, порівнюючи втрати та точність на тренувальних і валідаційних даних.

Спочатку задаються розміри області для графіків figsize і створюється дві підобласті для порівняння показників. У першому графіку відображаються втрати для тренувальних та валідаційних даних. Крива тренувальних втрат має штрихову лінію синього кольору, а валідаційних – пунктирну лінію червоного

кольору. Графік має підписані осі («Epochs» для кількості епох і «Loss» для втрат).

На другому графіку зображено точність моделі на тренувальних та валідаційних даних. Аналогічно до першого графіка, використовується синя штрихова крива для тренувальної точності та червона пунктирна для валідаційної. Осьові підписи вказують кількість епох та рівень точності, а легенда розташована в нижньому правому куті.

Параметр plt.tight_layout оптимізує простір між графіками, щоб уникнути накладання елементів. Функція plt.show відображає готовий графік для аналізу результату навчання моделі. Це дозволяє швидко оцінити, чи є модель стабільною, чи спостерігається перенавчання, і наскільки ефективно вона узгоджується з валідаційними даними. Код для створення двух графіків представлено в лістингу 5.7.

Лістинг 5.7 – Код для створення графіків

```
# Set up the plot dimensions and layout
plt.figure(figsize=(14, 5))

# Plot the training and validation loss
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], linestyle='--', color='blue',
label='Training Loss')
plt.plot(history.history['val_loss'], linestyle=':', color='red',
label='Validation Loss')
plt.title('Training vs. Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(loc='upper right')


# Plot the training and validation accuracy
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], linestyle='--', color='blue',
label='Training Accuracy')
plt.plot(history.history['val_accuracy'], linestyle=':', color='red',
label='Validation Accuracy')
plt.title('Training vs. Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
# Adjust the layout and display the plot
plt.tight_layout()
plt.show()
```

Графік порівняння втрат для тренувальних та валідаційних даних, а також графік порівняння точності тренувальних та валідаційних даних зображено на рисунку 5.6.

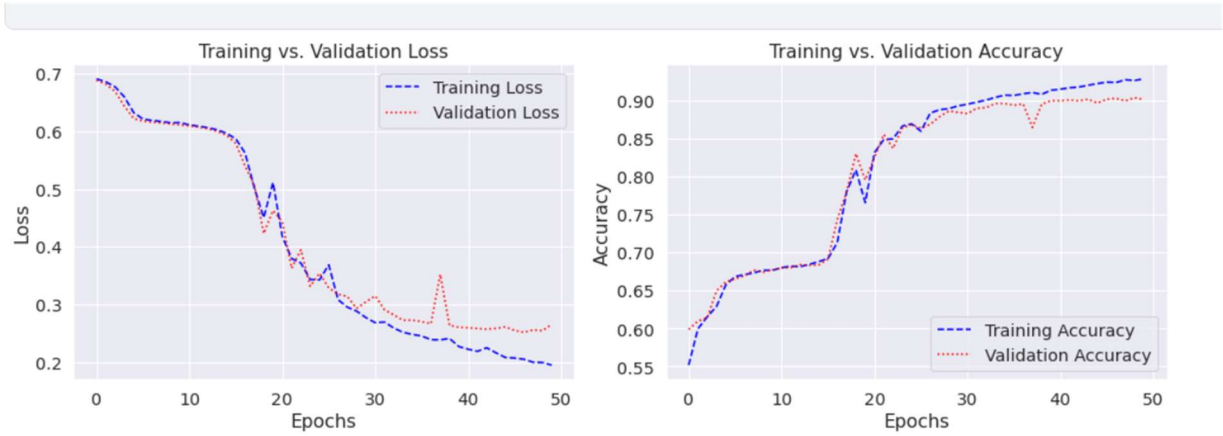

Рисунок 5.6 – Графіки порівняння тренувальних та валідаційних даних

Графік втрат демонструє поступове зменшення значень як для тренувальних, так і для валідаційних даних, що свідчить про ефективне наближення моделі до глобального мінімуму функції помилок. Незначні коливання валідаційних втрат у деяких епохах можуть бути пов’язані з адаптацією до нових даних або тимчасовим перенавчанням. Щодо точності класифікації, спостерігається чітка тенденція до її зростання в обох наборах даних. Наприкінці навчання модель демонструє точність понад 90%, що свідчить про здатність алгоритму ефективно навчатися та узагальнювати закономірності, не втрачаючи при цьому здатності до точного прогнозування на нових даних.

Функція preprocess_input_text призначена для перетворення введеного тексту у формат, який можна використовувати моделлю для передбачень. Текст перетворюється на послідовність чисел з використанням токенізатора tokenizer, який був навчений на тренувальних даних. Кожен токен (слово) замінюється відповідним числовим індексом. Якщо текст не містить токенів, які були у навчальних даних, повертається помилка з повідомленням: «Текст

не містить відомих токенів.» Послідовність заповнюється до зазначеної довжини max_len з допомогою методу pad_sequences. Заповнення додається до кінця послідовності (лістинг 5.8).

Лістинг 5.8 – Код для токенізації введеного тексту

```
def preprocess_input_text(input_text, tokenizer, max_len):
    sequences = tokenizer.texts_to_sequences([input_text])
    if len(sequences[0]) == 0:
        raise ValueError("Текст не містить відомих токенів.")
    padded_sequences = pad_sequences(sequences, padding='post',
maxlen=max_len)
    return padded_sequences
```

Функція predict_ptsd призначена для прогнозування можливості наявності посттравматичного стресового розладу на основі введеного тексту. Використовує функцію preprocess_input_text для перетворення вхідного тексту на формат, який підходить для моделі. Прогноз моделі передбачає:

- Передає оброблений текст модель для отримання передбачення;
- Повертає можливість наявності ПТСР, яка міститься в першому елементі передбачення;
- Обробка помилок.

Якщо у процесі виникає помилка (наприклад, текст не містить відомих токенів), виводить повідомлення про помилку та повертає None (лістинг 5.9).

Лістинг 5.9 – Код для обробки можливих помилок

```
def predict_ptsd(input_text, model, tokenizer, max_len):
    try:
  processed_text = preprocess_input_text(input_text, tokenizer, max_len)
        prediction = model.predict(processed_text)
        return prediction[0][0]
    except Exception as e:
        print(f"Error while processing: {e}")
        return None
```

Код виконує аналіз постів користувача Reddit із використанням LSTM-моделі для виявлення ознак ПТСР. Кожен текст обробляється токенізатором і передається до функції predict_ptsd, яка повертає ймовірність наявності розладу. Якщо вона перевищує 0.5, пост вважається «Positive». Інакше — «Negative». У разі помилки виводиться повідомлення «Prediction is not done».

Після обробки всіх постів розраховується частка позитивних результатів. Якщо вона менша за 25%, ПТСР не виявлено. Інакше — можливі ознаки ПТСР (лістинг 5.10).

Лістинг 5.10 – Код для обробки постів

```
print(f"\n--- Analyze user's posts u/{username} --- using LSTM\n")

for i, post in enumerate(user.submissions.new(limit=50), start=1):
    text = post.selftext.strip()
    if not text:
        continue
    total_checked += 1
    prediction = predict_ptsd(input_text, model, tokenizer, max_len)
    if prediction > 0.5:
        ptsd_count += 1
    print(f"Post #{i}")
    print("Text      :", text[:300])
    print("Prediction:", prediction)
    print("-" * 60)

if total_checked == 0:
    print("There are no text posts for analysis.")
else:
    ptsd_ratio = ptsd_count / total_checked
    if ptsd_ratio < 0.25:
        print(f"\nResult: PTSD not detected.")
    else:
        print(f"\nResult: Possible signs of PTSD.")
```

У межах експерименту було проаналізовано акаунт для виявлення можливих ознак ПТСР. Після запуску програми всі текстові публікації користувача оброблялися за допомогою функції predict_text, імовірність наявності розладу. У цьому випадку система не виявила ознак ПТСР. Це свідчить про відсутність у текстах тривожних або пригнічених настроїв. Зміст повідомлень мав переважно оптимістичне, пізнавальне або закличне спрямування: користувач ділиться оновленнями, ініціативами, звертається до підписників. Система сформувала висновок про відсутність симптомів ПТСР (рисунок 5.7), що свідчить про стабільний емоційний стан автора та підтверджує працездатність побудованої моделі.

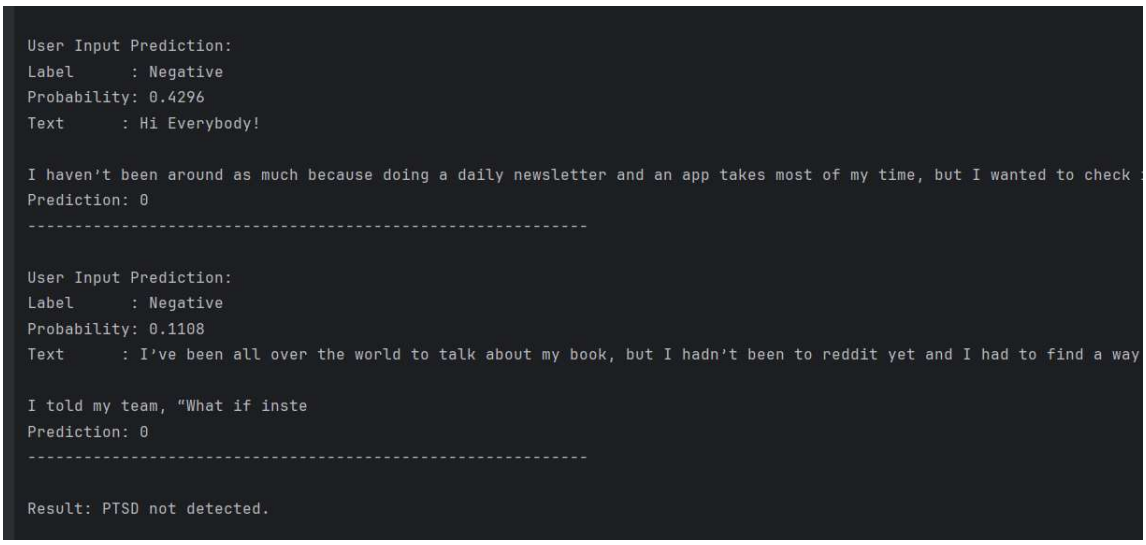

Рисунок 5.7 – Результати роботи: негативний результат ПТСР

Для порівняння результатів було також проаналізовано інший профіль користувача, публікації якого продемонстрували підвищений рівень емоційної напруги та тривожності. Система виявлення ПТСР класифікувала частину дописів як такі, що можуть містити ознаки психологічної дестабілізації, зокрема через вживання лексики, пов'язаної з травматичним досвідом, песимістичними настроями або самоідентифікацією з негативними емоційними станами. Це підтверджується графічним представленням результатів класифікації на рисунку 5.8.

Рисунок 5.8 – Результати роботи: позитивний результат ПТСР

ВИСНОВКИ

Діагностика посттравматичного стресового розладу на основі текстових даних є актуальним напрямом досліджень, що широко вивчається за допомогою сучасних алгоритмів машинного навчання, моделей обробки тексту та інструментів аналізу. Результати численних досліджень свідчать про високу ефективність автоматизованих підходів у виявленні психічних розладів, що створює можливості для впровадження інноваційних технологій у сферу психологічної допомоги.

У процесі роботи було розглянуто найбільш поширені алгоритми машинного навчання для аналізу тексту, зокрема наївний байєсівський класифікатор та рекурентні нейронні мережі типу LSTM. Модель наївного Байєса демонструє хороші результати у задачах класифікації тексту завдяки простоті реалізації, високій швидкості навчання та здатності працювати з великим обсягом розмірностей. Проте її ефективність може знижуватись у випадку залежності ознак або незбалансованості класів у навчальній вибірці. Натомість LSTM-моделі дозволяють враховувати контекст і послідовність слів у тексті, що робить їх особливо корисними для аналізу складних мовних патернів, характерних для проявів ПТСР, хоча вони потребують значних обчислювальних ресурсів та часу на навчання.

Таким чином, результати проведеного дослідження підтверджують доцільність використання методів машинного навчання для виявлення ознак посттравматичного стресового розладу в текстах і закладають основу для створення ефективних систем моніторингу психоемоційного стану користувачів.

Подальші дослідження повинні бути спрямовані на оптимізацію таких моделей з урахуванням специфіки вхідних даних, збалансованості вибірки та інтерпретованості результатів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Pagel J. F. Post-Traumatic Stress Disorder. Cham : Springer International Publishing, 2021 – 165 p.

2. Information Theory and Machine Learning / ed. by L. Zheng, C. Tian. MDPI, 2022. URL: https://doi.org/10.3390/books978-3-0365-5308-5 (date of access: 15.06.2025).

3. Ramos-Lima, L. F., Waikamp, V., Antonelli-Salgado, T., Passos, I. C. & Freitas, L. H. M. The use of machine learning techniques in trauma-related disorders: a systematic review. J. Psychiatric Res. 2021 – 159-172pp.

4. NSDG. Найкраща мова програмування для машинного навчання. *Sharp Coder Blog*. URL: https://uk.sharpcoderblog.com/blog/best-programming-language-for-machine-learning (дата звернення: 16.06.2025).

5. The Python Tutorial. *Python documentation*. URL: https://docs.python.org/3.13/tutorial/index.html (date of access: 16.06.2025).

6. Campbell A. Python Guide: Clear Introduction to Python Programming and Machine Learning. Independently Published, 2020.

7. Dataset Shift in Machine Learning / A. Schwaighofer et al. MIT Press, 2022.

8. IBM. What Are Naïve Bayes Classifiers? | IBM. *IBM - United States*. URL: https://www.ibm.com/think/topics/naive-bayes (date of access: 14.06.2025).

9. Multinomial Naive Bayes Explained. *Great Learning Blog: Free Resources what Matters to shape your Career!*. URL: https://www.mygreatlearning.com/blog/multinomial-naive-bayes-explained/ (date of access: 14.06.2025).

10. Dobilas S. LSTM Recurrent Neural Networks – How to Teach a Network to Remember the Past. *Towards data science*. URL: https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e/ (date of access: 14.06.2025).