

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Дослідження методів обробки та аналізу есе
англійською мовою з використанням алгоритмів
машинного навчання»**

Виконав студент групи ІПЗм-23-1
ОЛЕКСІЙЧУК Микита Валерійович

Керівник роботи:
ЧИЖМОТРЯ Олексій Володимирович

Рецензент:
к.т.н., доцент МОРОЗОВ Андрій Васильович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»
Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«01» жовтня 2024 р

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ОЛЕКСІЙЧУК Микита Валерійович**

Керівник роботи: **ЧИЖМОТРЕ Олексій Володимирович**

Тема роботи: **«Дослідження методів обробки та аналізу есе англійською мовою з використанням алгоритмів машинного навчання»**

Затверджена Наказом закладу вищої освіти від «01» жовтня 2024 р., №520/с

Вихідні дані для роботи: **дослідження галузі машинного навчання, розробки методів обробки текстових даних та алгоритму навчання моделей машинного навчання, вимірювання метрик ефективності роботи; програмні бібліотеки та методи роботи з файлами табличних даних та текстами, попередньої обробки даних, методи машинного навчання та окремі моделі.**

Керівники / консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Керівник/консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.В.	01.10.2024р.	01.10.2024р.
2	Чижмотря О.В.	01.10.2024р.	01.10.2024р.
3	Чижмотря О.В.	01.10.2024р.	01.10.2024р.

РЕФЕРАТ

Кваліфікаційна робота магістра складається з опису виконаного дослідження предметної області та алгоритмів, розробки ПЗ системи обробки вхідних даних і навчання моделей машинного навчання, та пояснювальної записки. Пояснювальна записка до випускної роботи містить текстову частину, викладену на 63 сторінках друкованого тексту, має 10 сторінок додатків. Список використаних джерел містить 44 найменування і займає 5 сторінок. В роботі наведено 28 рисунків та 1 таблицю. Загальний обсяг роботи – 73 сторінки.

Метою роботи є виконання дослідження алгоритмів машинного навчання і попередньої обробки текстових даних та розробка ПЗ з обробки вхідних даних, навчання моделей машинного навчання та порівняння ефективності роботи засобами мови програмування Python та бібліотек, модулів на основі зазначеної мови програмування, а також налаштування збереження навчених моделей у файлах для можливості подальшого використання.

У роботі описано процес дослідження джерел за предметною областю, зокрема історії галузей машинного навчання та обробки природної мови, парадигм, задач та моделей машинного навчання. Також було обрано технології розробки та обґрунтовано інструментальні засоби, що були використані в розробці, виконано проектування ПЗ та описано архітектуру моделі штучної нейронної мережі, проведено вимірювання метрик ефективності роботи моделей та їх аналіз, порівняння, підсумування результатів та можливостей покращення ПЗ.

КЛЮЧОВІ СЛОВА: МАШИННЕ НАВЧАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, ЕСЕ, ОБРОБКА ТЕКСТОВОЇ ІНФОРМАЦІЇ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

ABSTRACT

The master's thesis consists of a description of the completed research of the subject area and algorithms, software development of the system for processing input data and training machine learning models, and the paper. The paper to the graduation thesis contains a text part set out on 63 pages of printed text and has 10 pages of appendices. The list of references includes 44 items and covers 5 pages. The paper contains 28 figures and 1 table. The total volume of the thesis is 73 pages.

The purpose of the master's thesis is to study machine learning algorithms and text data preprocessing and to develop software for processing input data, training machine learning models, and comparing their performance using Python programming language, libraries and modules based on the specified programming language, as well as setting up the storage of trained models in files for further use.

The paper describes the process of researching sources in the subject area, including the history of machine learning and natural language processing, paradigms, tasks, and machine learning models. Also, the development technologies were selected and the tools used in the development were substantiated, the software was designed and the architecture of the artificial neural network model was described, the performance metrics of the models were measured and analyzed, compared, and the results and possibilities for improving the software were summarized.

KEYWORDS: MACHINE LEARNING, NATURAL LANGUAGE PROCESSING, ESSAY, TEXT INFORMATION PROCESSING, SOFTWARE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. ПОШУК, АНАЛІЗ ТА ДОСЛІДЖЕННЯ ДЖЕРЕЛ	
ІНФОРМАЦІЇ ЗА ПРЕДМЕТНОЮ ГАЛУЗЗЮ РОБОТИ	10
1.1. Постановка задачі	10
1.2. Дослідження галузі та пошук джерел за предметною областю	12
1.3. Дослідження алгоритмів попередньої обробки текстових даних	20
1.4. Аналіз та обґрунтування вибору інструментальних засобів	24
Висновки до першого розділу	26
РОЗДІЛ 2. ДЕТАЛЬНИЙ АНАЛІЗ ТЕХНОЛОГІЙ ТА ПРОЕКТУВАННЯ	
ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	28
2.1. Пошук та аналіз набору даних	28
2.2. Опис архітектури програмного забезпечення дослідження	29
2.3. Підготовка середовища розробки та попередня розробка ПЗ	34
Висновки до другого розділу	35
РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ,	
НАВЧАННЯ ТА ПОРІВНЯННЯ МОДЕЛЕЙ	36
3.1. Розробка алгоритмів попередньої обробки текстів та векторизації ...	36
3.2. Розробка та навчання моделей машинного навчання та моделі штучної нейронної мережі	42
3.3. Аналіз результатів навчання та порівняння ефективності роботи алгоритмів	47
Висновки до третього розділу	56
ВИСНОВКИ	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	59
ДОДАТКИ	64

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПЗ – Програмне забезпечення.

NLP – Natural Language Processing

ML – Machine Learning

IELTS – International English Language Testing System

SVM – Support Vector Machine

XGBoost – Extreme Gradient Boost

ReLU – Rectified Linear Unit

ВСТУП

Разом із розповсюдженням продуктів розвитку технологій, зокрема інформаційних, та доступу до мережі Інтернет підходи до розвитку давньої ідеї про створення штучного розуму в обчислювальній машині розвивались, видозмінювались та, врешті-решт, сформувались у те, що сьогодні відоме, як машинне навчання. Перехід від систем, заснованих на правилах, до систем, що керують даними, в умовах доступу до шаленої кількості інформації з будь-якої точки світу дозволило зробити прорив у галузі, разом із тим проклавши шлях для інших.

Разом із розвитком машинного навчання, відбувся зсув у такій галузі, як NLP, надавши друге дихання ідеям, що закладені в дану область. Поєднання сучасних інформаційних технологій, а саме нового статистичного підходу машинного навчання, дозволило не тільки обробляти числові дані для підприємств, а навіть навчити машину розуміти людську мову.

Такі системи, як ChatGPT, Claude, Gemini, невпинно покращуються та навчаються на більших вибірках даних, а завдяки розвитку розуміння людської мови та логіки навіть мають змогу виконувати задачі з математики, аналізу, написання робочого програмного коду тощо. Для розуміння яким чином було досягнуто таких результатів, необхідно виконати дослідження галузей, їх історії та розвитку, сучасних технологій розробки систем з машинного навчання та обробки природної мови і особливостей обробки текстових даних для навчання на них, а протягом дослідження – порівняння окремих перспективних алгоритмів між собою на обраних даних.

Мета даної роботи полягає у виконанні дослідження методів обробки та аналізу текстових даних у вигляді есе англійською мовою із використанням методів машинного навчання та розробки відповідного ПЗ на мові програмування Python та використанні таких бібліотек та інструментів, як NumPy, Pandas, Scikit-learn, NLTK, Gensim, Tensorflow тощо. Дослідження джерел має надати знання з галузі, основних задач машинного навчання, способів обробки текстової інформації у прийнятний для обчислювальних

машин формат, а розроблене на основі теорії ПЗ має забезпечувати функціонал з ефективною попередньою обробки текстів датасету, налаштування, навчання та збереження моделей машинного навчання, порівняння метрик ефективності їх роботи.

Поставлена мета обумовлює необхідність виконання наступних завдань:

- пошук та аналіз вітчизняних та зарубіжних джерел інформації по даних предметній галузі інформаційних технологій;
- дослідження методів обробки даних текстового формату, способів перетворення текстів вільного розміру в дискретні цифрові значення;
- аналіз сучасних засобів розробки, бібліотек та окремих технологій та алгоритмів з галузі машинного навчання;
- виконання попереднього налаштування середовища роботи ПЗ, включно зі збереженням вихідних моделей та дотичних до них об'єктів;
- розробка ПЗ згідно виконаних досліджень та аналізу;
- навчання моделей машинного навчання та покращення результатів через коригування параметрів обробки текстів та самих моделей;
- вимірювання визначених згідно задачі моделей метрик ефективності роботи моделей;
- порівняння роботи моделей на незалежних від навчальної вибірки даних згідно метрик та підсумування результатів даного дослідження та розробленого ПЗ;

Об'єктом дослідження є процес розробки алгоритмів попередньої обробки текстової інформації, імплементації та налаштування моделей машинного навчання та подальшого навчання, вимірювання показників ефективності роботи та проведеного навчання алгоритмів.

Предметом дослідження є інструментарій, необхідний для розробки алгоритмів обробки, розробки архітектури моделі штучної нейронної мережі, та існуючі у використаних інструментальних засобах моделі машинного навчання.

Наукова новизна дослідження полягає в розробці та апробації методів аналізу текстів есе англійською мовою із застосуванням сучасних алгоритмів машинного навчання, таких як SVM, XGBoost та моделі штучних нейронних мереж. Запропоновано підхід, що інтегрує методи попередньої обробки текстових даних, включаючи лематизацію та векторизацію Doc2Vec, для підвищення точності аналізу. Застосований підхід також враховує унікальність датасету есе IELTS, що дозволяє розширити можливості автоматизованого оцінювання текстів у навчальному контексті.

Практичне значення роботи визначається створенням програмного забезпечення, яке дозволяє ефективно обробляти текстові дані, навчати моделі машинного навчання та зберігати їх для подальшого використання. Розроблене ПЗ може застосовуватись для автоматизованого оцінювання есе у системах навчання, таких як IELTS, з метою спрощення перевірки текстів та зменшення навантаження на викладачів. Крім того, результати дослідження можуть бути інтегровані у системи підтримки прийняття рішень та інші прикладні задачі з аналізу текстової інформації.

В роботі було використано існуючі статистичні алгоритми машинного навчання та окремі компоненти розробки штучних нейронних мереж у вигляді налаштовуваних шарів нейронів, функцій впливу на навчання штучної нейронної мережі, як callback-и, функції регуляризації, активаційні функції тощо.

Розроблене ПЗ може бути покращене при виконанні подальшого дослідження з урахуванням поточних результатів, а саме алгоритм обробки даних може бути допрацьовано, а моделі машинного навчання – доналаштовані гіперпараметрами. Зокрема, архітектура моделі штучної нейронної мережі може бути покращена або суттєво змінена разом із використанням інших підходів до обробки даних для отримання кращих результатів.

Апробація. За темою кваліфікаційної магістерської роботи було опубліковано 3 тези конференцій [1][2][3].

РОЗДІЛ 1. ПОШУК, АНАЛІЗ ТА ДОСЛІДЖЕННЯ ДЖЕРЕЛ ІНФОРМАЦІЇ ЗА ПРЕДМЕТНОЮ ГАЛУЗЗЮ РОБОТИ

1.1. Постановка задачі

Головною метою кваліфікаційної роботи є дослідження методів обробки тексту та технологій машинного навчання на сучасному етапі їх розвитку у виконанні аналізу текстових творчих робіт та порівняння ефективності використання обраних технологій між собою відносно їх складності. Виконане дослідження дозволить отримати теоретичні та практичні знання, що дозволять грамотно аналізувати можливості різних технологій галузі, методів машинного навчання та доцільність їх застосування, а також сильні та слабкі сторони, специфіку моделювання та розробки.

На основі результатів глибокого дослідження джерел інформації за предметною областю та об'єктами дослідження, необхідно реалізувати у програмному коді систему для виконання задач із вище згаданого аналізу текстів. Загалом, дане дослідження можна розділити на наступні етапи:

1. Виконання теоретичного дослідження технологій:
 - 1.1. Аналіз наукових та інформаційних джерел за темою дослідження та технологіями.
 - 1.2. Дослідження окремих алгоритмів до подальшої імплементації.
 - 1.3. Вибір доцільних технологій розробки з урахуванням їх можливостей.
2. Організація вхідних даних, проектування та опис архітектури:
 - 2.1. Виконання пошуку та аналізу датасету до подальшого використання в навчанні.
 - 2.2. Опис архітектури програмного забезпечення з аналізом обраних засобів розробки з описаного інструментарію.
 - 2.3. Попередня розробка коду для підготовки середовища до роботи алгоритмів обробки даних та моделей.

3. Розробка програмного забезпечення та аналіз результатів:

- 3.1. Написання програмного коду обробки вхідних даних згідно результатів аналізу та проектування.
- 3.2. Написання програмного коду навчання моделей та отримання метрик навчання.
- 3.3. Аналіз отриманих на незалежних даних метрик та порівняння результатів роботи обраних моделей між собою.

При виконанні теоретичного дослідження технологій, а саме аналізі наукових джерел інформації, буде виконаний пошук серед вітчизняних та іноземних, міжнародних наукових джерел за предметами дослідження для отримання та поглиблення існуючих знань про технології та способи вирішення поставлених задач та визначеної проблематики. Зокрема, буде виконано аналіз інструментів розробки, що здатні до виконання задач із областей машинного навчання та обробки природної мови, та їх порівняння для вибору найефективніших методів та засобів із урахуванням можливої необхідності створення взаємодії між компонентами, які використовують різні стеки технологій.

В результаті пошуку та аналізу потенційних датасетів та проектування системи, опису архітектури, буде створено чіткі бачення майбутнього програмного забезпечення та план по його реалізації у програмному коді з використанням попередньо проаналізованих та обраних технологій розробки. Зокрема, будуть використані додаткові засоби для ефективного виконання аналізу та моделювання системи.

Програмна реалізація продукту на основі всіх попередніх кроків має в результаті дати практичний досвід роботи з технологіями та реалізації запланованого функціоналу згідно результатів теоретичного дослідження. Програмний продукт має відповідати поставленим вимогам та базуватись на результатах моделювання, опису архітектури системи, виконувати всі поставлені задачі. Для досягнення цього будуть виконуватись аналіз

проміжних та кінцевих метриках навчених моделей для їх налаштування для найкращої можливої ефективності роботи в умовах поставлених їм задач.

1.2. Дослідження галузі та пошук джерел за предметною областю

Реалізація методів обробки текстових робіт та дослідження ефективності різних алгоритмів вимагає глибокого дослідження таких галузей штучного інтелекту, як машинне навчання (ML) та обробка природної мови (NLP). Дані галузі штучного інтелекту набули широкого поширення через ефективність їх роботи при аналізі даних для бізнесів, наукових досліджень, та виконанні запитів на пошук інформації для користувачів.

Машинне навчання фокусується на використанні даних та алгоритмів для імітації людського інтелекту та процесів навчання [4][5][6]. Їх застосовують для вирішення складних завдань тим самим шляхом, яким ці завдання виконує людський мозок [4]. Використання статистичних методів у навчанні алгоритмів дозволяє виконувати передбачення даних або їх класифікацію для отримання важливої інформації по ключовим метрикам бізнесу та спрощення процесів прийняття рішень на їх основі [5][6].

Обробка природної мови дозволяє комп'ютерам розуміти, генерувати та маніпулювати людською мовою через здібність аналізувати дані, отримані з природної мови у вигляді тексту чи голосу [7]. Даною технологією користується велика кількість людей завдяки віртуальним асистентам на кшталт Siri, Alexa тощо та функціям фільтрації спаму поштових систем, перевірки граматики та орфографії текстів, автоматичного перекладу чи підсумування документів.

Обидві галузі штучного інтелекту отримали початок у середині минулого століття із появою та популяризації кібернетики та комп'ютерних наук, досліджень людського мозку, нейронів та подальших ідей про можливість створення нейронних мереж на основі цих досліджень у апаратно-програмному вигляді. Хоча сьогоденні системи використовують результати нещодавніх досягнень, витоки пов'язаних з ними наук можна

прослідкувати до 1940х років [8]. Робота “Логічне числення ідей, властивих нервовій діяльності” Уолтера Піттса та Уоррена Маккалока 1943го року пропонувала першу модель нейронної мережі, а книга “Організація поведінки: нейропсихологічна теорія” Дональда Гебба 1949го року містить теорії про те, як поведінка пов’язана з нейронними мережами та мозковою діяльністю [8][9]. Перша комп’ютерна програма з навчанням була створена та вперше термін “машинне навчання” було згадано у 1952му році Артуром Семюелем, а в 1957му році Френк Розенблат розробив першу нейронну мережу для комп’ютерів – перцептрон. Через недостатню ефективність тогочасних технологій подальші дослідження сповільнились, допоки у 1990х роках робота над галуззю не перейшла від підходу керування знаннями до підходу керування даними завдяки виникненню мережі Інтернет та спрощенню доступу до інформації. Програми замість використання наявних знань почали робити власні висновки із великих обсягів даних, що їм надали, та орієнтуватись на ці висновки у майбутньому.

Основа обробки природної мови була закладені ще на початку минулого століття Фердинаном де Сосюром з появою ідеї про те, що будь-яка людська мова є системою, а сенс того, що мова передає, будується на взаємозв’язках та контрастах [10]. Зі смертю вченого, його ідеї підхопили Альбер Сеше та Шарль Баллі, що у 1916му році опублікували книгу “Курс загального мовознавства”, започаткувавши структурну лінгвістику. На її основі були виконані перші спроби створення ефективних систем перекладу та розуміння тексту через зміну структури речень та їх переклад у зрозумілий для машини формат. Меморандум “Переклад” Воррена Вівера 1949го року започаткував ідею використання комп’ютерів для перекладу та термін “машинний переклад” [11]. У 1958му році було створено мову програмування LISP Джоном Маккарті, а у 1964му було написано програму ELIZA для імітації психіатра, хоч і без розуміння запитів, та створено Консультаційний комітет з автоматичної обробки мови ALPAC. У 1966му році даний комітет оголосив про зупинку підтримки та фінансування

досліджень з галузі на фоні дорогої вартості машинного перекладу незважаючи на 12 років досліджень та витрачені 20 мільйонів доларів США на них [10][11]. Лише на початку 1990х років з розвитком галузі машинного навчання ідеї машинного перекладу отримали друге дихання, замінивши рукописні правила для систем на статистичні моделі.

Сучасні технології машинного навчання розділяють на різноманітні парадигми з різними підходами до навчання моделей та можливостей застосування в залежності від поставлених задач та умов, і зазвичай саме дані та задачі визначають вибір парадигми. З основних парадигм наразі виділяють контрольоване, неконтрольоване навчання та навчання з підкріпленням [12]. Окрім даних парадигм, існують більш специфічні для виконання нішевих завдань, наприклад, із постійним потоком даних або як комбінація основних парадигм.

Спільними для всіх парадигм машинного навчання для їх ефективного застосування є виконання наступних 7 етапів [6]:

1. Збір даних: будь-яка система потребує великої кількості даних.
2. Підготовка даних: зібрані сирі дані необхідно обробити, нормалізувати, очистити від помилок та дублювань.
3. Вибір моделі: в залежності від задачі, яку необхідно виконати, та природи даних необхідно обрати парадигму та модель.
4. Навчання моделі: обрану модель чи моделі необхідно налаштувати на використання даних та запустити процес навчання.
5. Оцінка моделі: результати роботи моделі протягом навчання кожного кроку, зокрема і на кінцевому етапі, треба вимірювати спеціальними метриками із порівнянням попередніх висновків.
6. Налаштування параметрів: у цей етап входить точне налаштування протягом покрокового навчання параметрів та гіпер-параметрів, таких як кількість кроків роботи алгоритму, швидкість навчання, розподіл та інші в залежності від моделі.

7. Виконання передбачень: навчену та налаштовану модель застосовують для вирішення питань та отримання відповідей на нових даних.

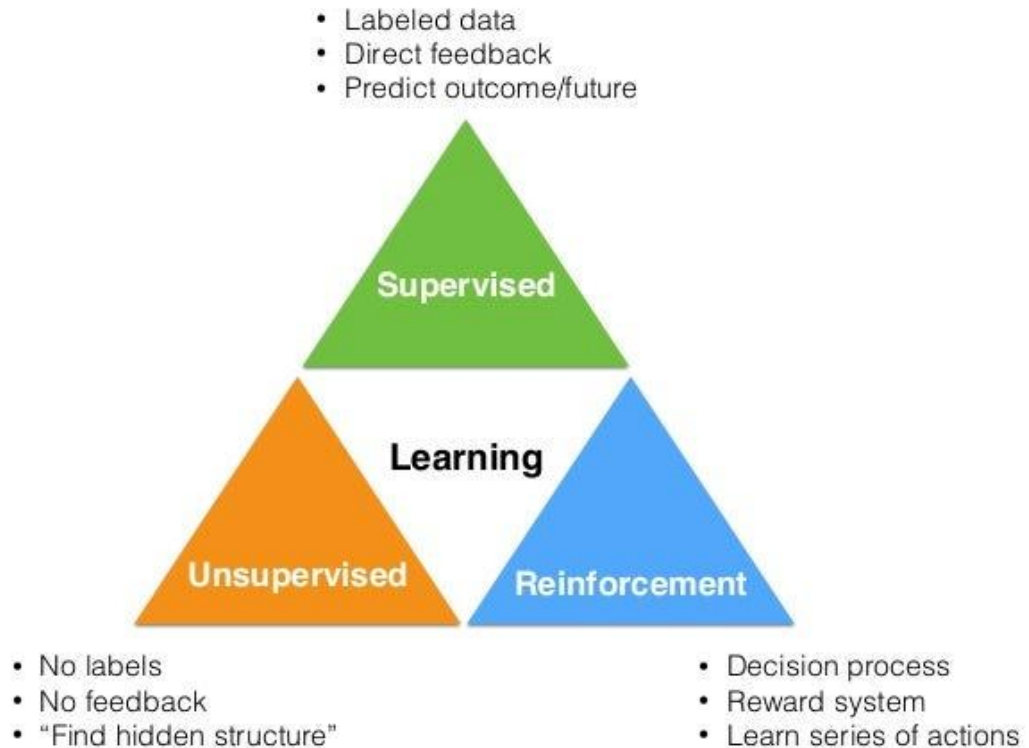


Рис.1.1 – Графічний опис основних парадигм машинного навчання

Контрольоване навчання є найпоширенішою парадигмою машинного навчання, що використовує позначені набори даних у вигляді пар вхідної та вихідної інформації, або інакше відомих як “незалежних” та “залежної змінної” або “ознак” та “цілі” [13] відповідно. Ознаки окремої одиниці набору даних є вектором ознак. Метою контрольованого навчання є створення та тренування на основі навчальних даних моделі для передбачення вихідних значень для нових, ще незнайомих для моделі, ознак. До контрольованого навчання входять завдання регресії та класифікації даних [12][13].

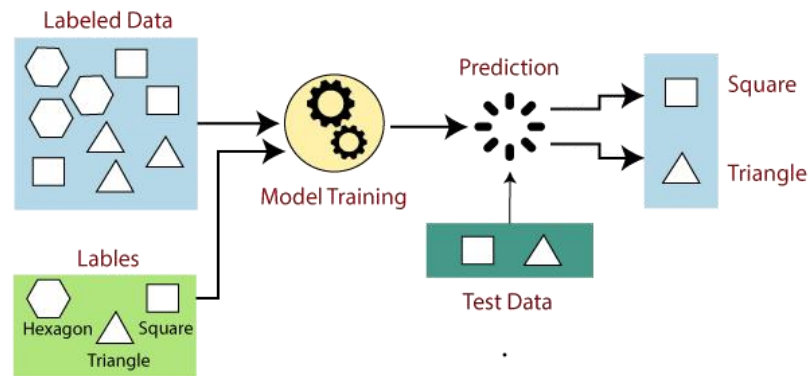


Рис.1.2 – Загальна модель роботи алгоритмів контрольованого навчання

Регресія має на меті передбачати безперервні цифрові значення, як вага, вартість, температура тощо на основі векторів ознак набору даних. Прикладом регресії може бути передбачення зарплатні працівника згідно стажу роботи та посади. Для виконання завдань регресії були створені різноманітні алгоритми машинного навчання, найпоширенішими з яких є [13][14]:

- лінійна регресія;
- поліноміальна регресія;
- регресія деревом рішень;
- регресія випадковим лісом;
- регресія методом k-найближчих сусідів;
- регресія нейронними мережами;
- регресія опорно-векторними машинами.

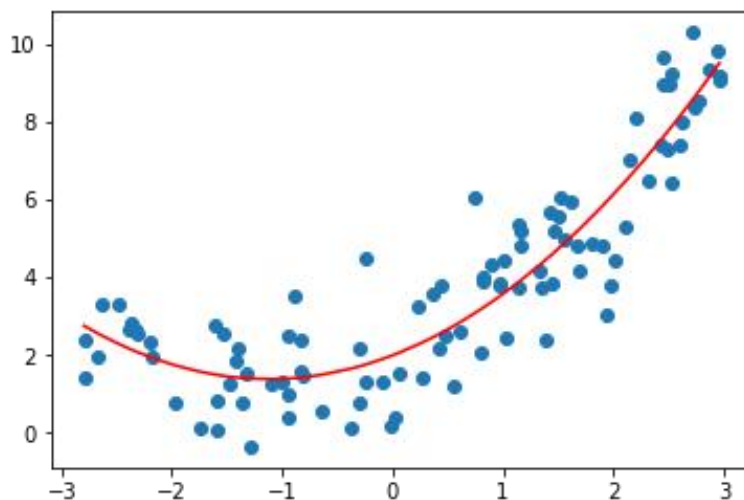


Рис.1.3 – Приклад візуалізації поліноміальної регресії

Класифікація виконує передбачення належності даних чи результатів роботи до наперед визначених категорій. Прикладом завдання класифікації можна вважати статистичне передбачення хвороби за певними медичними показниками або підвиду рослини за характеристиками листя та квітки. Серед найпоширеніших алгоритмів класифікації в машинному навчанні виділяють наступні [13][15]:

- логістична регресія;
- наївний баєсів класифікатор;
- класифікація методом k-найближчих сусідів;
- класифікація деревом рішень;
- класифікація випадковим лісом;
- класифікація опорно-векторними машинами;
- класифікація нейронними мережами.

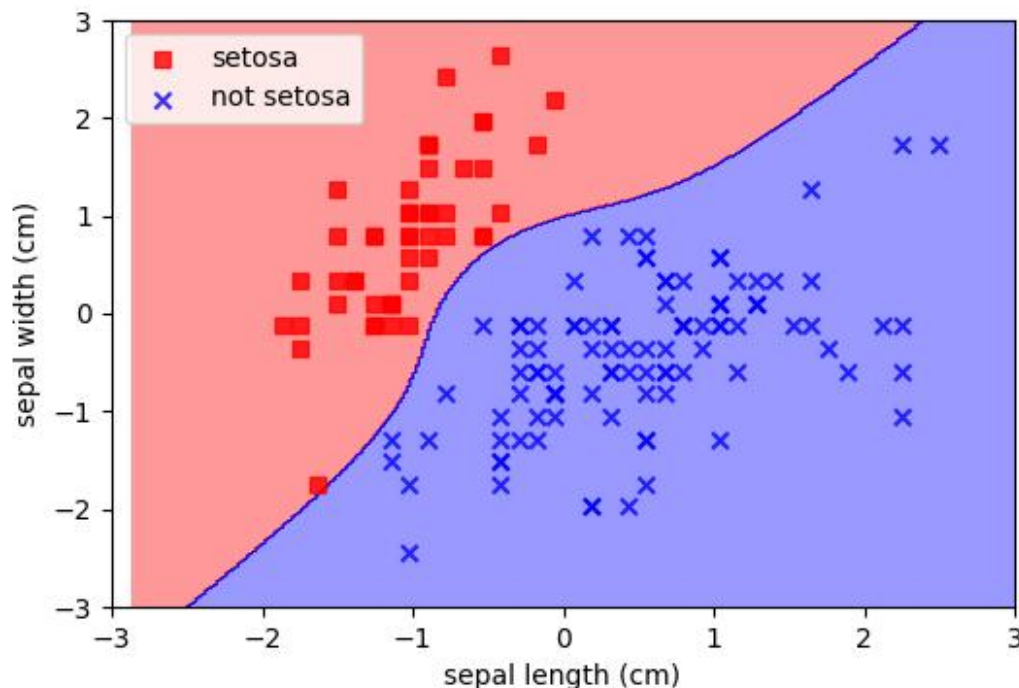


Рис.1.4 – Приклад візуалізації нелінійної класифікації

Парадигма неконтрольованого навчання є наступною за популярністю після контрольованого зі суттєвою різницею в структурі вхідних даних, а саме відсутність залежних змінних, на які система може орієнтуватись у своєму навчанні та подальшій роботі [16]. Натомість, завданням даної

парадигми необхідно самостійно знайти зв'язок між одиницями даних та виявити часом приховані патерни у вхідній інформації, при цьому жертвуючи точністю та надійністю. Серед завдань даної парадигми головними є завдання кластеризації, пошуку асоціативних правил та зменшення розмірності.

Кластеризація ґрунтується на теоретичній точці зору, що елементи або екземпляри однієї групи володіють схожими властивостями, і тому підлягають об'єднанню за ними у кластери [16][17][18]. Прикладом виконання кластеризації може бути пошук груп серед користувачів по їх доходам, активності на сервісі, віку, статі та іншим параметрам, що дозволяє створення різних стратегій. В залежності від природи даних, методи кластеризації діляться на жорстку, де елемент може бути лише в одній групі, та м'яку, де елемент може бути членом різних кластерів, а також по різних типам, основними з яких є [17][18] кластеризації розділів, на основі щільності, на основі моделі розподілу, ієрархічна та нечітка кластеризація (м'яка).

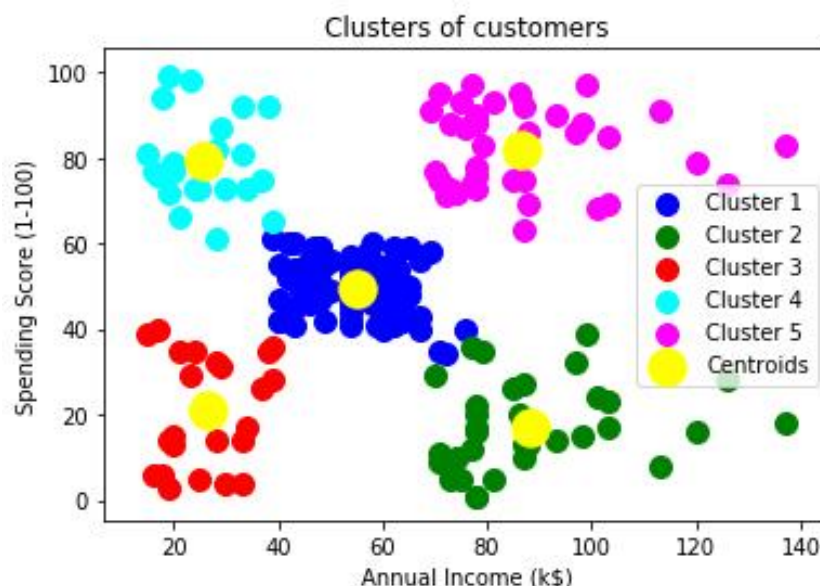


Рис.1.5 – Приклад візуалізації кластеризації розділів методом k–середніх

Навчання з підкріпленням є подібною до керованого навчання парадигмою в тому, що обидва підходи керуються вихідною інформацією в процесі навчання. У парадигмі навчання з підкріпленням, проте, вихідна

інформація замість правильних відповідей та даних включає у себе нагороди та покарання за певні дії системи, яку в рамках парадигми звуть агентом, та аналіз поточного стану з урахуванням нагород чи покарань [19][20]. Таким чином, системи машинного навчання цієї парадигми ставлять за мету максимізацію кінцевого результату, при цьому для побудови оптимальної стратегії перед агентом постає дилема дослідження нових станів та максимізації результату вже відомими методами [19].

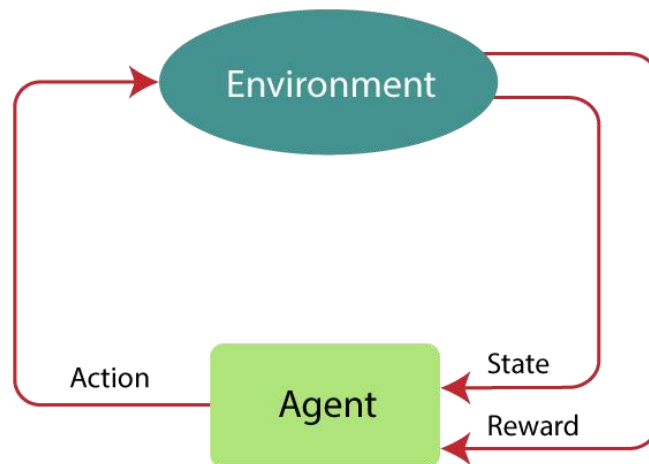


Рис.1.6 – Модель роботи алгоритмів навчання з підкріпленням

Підкріплення у навчанні поділяється на 2 типи: позитивне та негативне [20]. Позитивне підкріплення від певної поведінки збільшує частоту дотримання даної поведінки, при цьому завелика кількість нагород може суттєво знизити якість результатів. Негативне у свою чергу накладає штрафи за поведінку, зменшуючи “бажання” системи до виконання подій, що призводять до цього, але дане підкріплення дозволяє лише встановити мінімальну межу. Для якісного навчання необхідне ефективне використання обох типів та можливу варіацію нагороди/штрафів від окремих дій. Дана парадигма може застосовуватись у робототехніці на підприємствах для автоматизації або регулювання параметрів роботи заводів у режимі реального часу [20].

Для вирішення поставленої задачі в даній роботі необхідно встановити методи обробки даних та алгоритми, які будуть навчатись на них. Зокрема, в процесі дослідження датасету необхідно встановити спосіб очистки даних

для уніфікації стилів текстів та виправлення можливих помилок форматування залежних та незалежних змінних.

Після виконання пошуку необхідних алгоритмів машинного навчання буде виконано подальше моделювання системи для дослідження та порівняння результатів їх роботи. У процесі пошуку алгоритмів необхідно брати до уваги формат вхідних даних, адже саме від них залежить яку роботу має виконувати система, що у свою чергу вказує на відповідні парадигму, задачу та моделі машинного навчання.

1.3. Дослідження алгоритмів попередньої обробки текстових даних

Специфіка типової роботи систем машинного навчання полягає у тому, що прийнятним форматом для моделей є цифровий, адже класичні моделі та алгоритми побудовані на використанні математичних формул для передбачення результатів на основі дискретних значень. Зокрема, у випадку роботи із форматами даних, що не є цифровими, виконуються операції з категоризації.

У процесі аналізу текстів вхідні дані здебільшого мають неструктурований або напівструктурований характер [21], оскільки текстові дані відзначаються гнучкістю довжини та неоднорідністю. Тому для застосування алгоритмів машинного навчання необхідно структурувати дані шляхом їх попередньої обробки.

До методів попередньої обробки тексту включають такі операції, як нормалізація тексту, а саме приведення до єдиного регістру, очищення від розділових знаків, очищення від стоп-слів, токенизація, тобто розділення тексту на токени, роль яких виконують слова, і на основі цього: добування коренів слів (stemming) [22][23][24], лематизація, векторизація.

Проведення нормалізації та очищення від розділових знаків є можливим зі стандартними функціями більшості мов програмування, у той час як для очищення тексту від стоп-слів є можливість використовувати спеціалізовані засоби та бібліотеки. За своєю суттю стоп-слова – це слова, які не несуть великого смислового навантаження у тексті, як сполучники, частки,

через що для покращення якості остаточних результатів та економії ресурсів їх варто очищати [25].

Після проведення токенизації попередньо обробленого тексту, стає можливим використання одного з двох підходів приведення слів до їх базової форми, зменшуючи кількість словоформ зі збереженням змісту: стеммінг та лематизація.

Стеммінг – метод обробки слова, що прибирає з нього варіативні частини, як префікси, суфікси та закінчення, залишаючи основу слова. Таким чином, одне слово, що у різних формах повторюватиметься у тексті, уніфікується, зменшуючи кількість токенів. Даний метод має недолік у тому, що оброблені токени можуть перетворитись на неіснуючі слова, наприклад, прибираючи закінчення “-ing”, у цей же час не оброблюючи дієслово “is”. Окрім цього, можливе виникнення проблем надмірної чи недостатньої обробки, наприклад, зводячи неспоріднені слова до однієї основи та, навпаки, незведення до однієї основи слів споріднених, відповідно.

У порівнянні з цим, лематизація є методом обробки слова, що також прибирає варіативні частини, але не “обрізаючи” їх, а встановлюючи базову форму слова. Даний метод дозволяє обробити текст, залишаючи нормалізовані версії, або леми, слів, що існують за словниками, і відповідно. Недоліком даного методу є витрата обчислювальних ресурсів для перевірки слів згідно зі встановленими словниками, що накладає обмеження та підвищує вимоги до системи [26].

Stemming vs Lemmatization



Рис.1.7 – Порівняння стеммінгу та лематизації на прикладі однакових слів

Після попередньої обробки тексту, його необхідно перетворити у формат, зрозумілий для моделей машинного навчання, а саме у фіксованого розміру числовий набір даних. У той час, як тип есе та оцінка вже є числового типу, для тексту необхідно виконати його векторизацію. Векторизація – процес перетворення неоднорідних текстових даних на однорідний вектор. Існує велика кількість методів векторизації, які мають власні переваги та недоліки, та відповідно власні сфери застосування в залежності від вхідних даних. [27]

Векторизація керується словником, створеному по всім унікальним словам корпусу, тобто усім окремим текстам датасету, які звуться документами. Найпростіші методи векторизації, такі як Bag-of-Words та засновані на ньому BM25 та TF-IDF, виконують створення розрідженого вектора, від довжини якого залежить кількість індивідуальних, найбільш частих слів зі словника (та одна позиція під решту, рідкісні), по яких буде вестись підрахунок. Останні методи також виконують додаткові операції, як зваження слів відносно частоти слів у всьому корпусі та врахування довжини

документу відповідно [27]. Явною перевагою даних методів є простота імплементації та логіки роботи, недоліками ж є втрата порядку та контексту слів разом зі створенням великих масивів даних у 10-20 тисяч елементів, більшість з яких можуть бути рівними 0, тобто розріджені матриці.

Для вирішення проблеми втрати контексту можливе використання програм при токенизації, тобто окремі токени є комбінаціями слів у тому порядку, в якому вони зустрічаються у документі. Кількість слів у токенах (n) дозволяє частково зберігати порядок слів при векторизації та навчанні моделей, але так само не вирішує відсутність контексту [28].

Відомими техніками, що дозволяють взяти до уваги контекст навколишніх слів, є засновані на використанні нейронних мереж та Deep Learning, наприклад Word2Vec та Doc2Vec. Обидва підходи використовують парадигму неконтрольованого навчання, що дозволяє навчати моделі без наявності залежних змінних.

Word2Vec дозволяє створити модель векторизації текстів через врахування навколишніх слів при навчанні на окремих словах документів та включає в себе 2 основні моделі: Continuous Bag-of-Words (CBOW) та Skip-gram. Перша модель тренується передбачати слова за наданим контекстом у вигляді вектора слів, у той час як друга навпаки – тренується передбачати контекст за наданим словом. Розмір вектору слів, які використовуються у власних алгоритмах, залежить від параметру розміру вікна контексту. CBOW є швидшою моделлю та має високу ефективність при роботі із невеликими датасетами, навчаючись на локальних паттернах але програючи в збереженні глобальної інформації. Skip-gram, попри повільніше навчання, є більш гнучкою та ефективнішою із великими наборами даних. [27][28]

Doc2Vec є розширенням Word2Vec, але замість вкладання слів (word embedding) фокусується на цілих документах. При виконанні векторизації Doc2Vec стає можливим порівняння документів між собою та загальної роботи із датасетами великих текстів. Основними моделями підходу є Distributed Memory (DM) та Distributed Bag-of -Words (DBOW). В архітектурі

DM В архітектурі DM нейронна мережа приймає два типи вхідних даних: контекстні слова, що використовуються для передбачення цільового слова, та унікальний ідентифікатор документа, необхідний для фіксації загального значення документа. Мережа складається з двох основних компонентів: проекційного шару для створення векторів слів і векторів документів і вихідного шару. DBOW є простішою моделлю, що фокусується на розумінні розподілення слів у тексті замість їх значення, при цьому вона не враховує порядок слів у документі. Таким чином, DM дозволяє аналізувати зміст документів та дозволяє захопити семантичне значення документа, у той час як DBOW – лише структуру то розподіл слів [29].

1.4. Аналіз та обґрунтування вибору інструментальних засобів

Для створення системи машинного навчання для виконання поставлених задач, а саме аналізу текстів різними способами та алгоритмами, було вирішено обрати такі засоби розробки, як мову програмування Python 3.12 та інтегроване середовище розробки на мові Python JetBrains PyCharm, бібліотеки обраної мови NumPy, Scikit-learn, NLTK, spaCy, Pandas, систему керування версіями git та платформу для спільної розробки та збереження репозиторіїв коду GitHub.

Python версії 3.12 – інтерпретована мова програмування високого рівня загального призначення зі суворою динамічною типізацією, розроблена у 1990-му році Гвідо ван Россумом. Філософія дизайну мови наголошує на зручності читання коду з використанням значних відступів. Дана мова програмування підтримує різні парадигми програмування, включаючи структуроване, об'єктно-орієнтоване, функціональне. На мові Python написано бібліотеки для веб-розробки, розробки користувацького інтерфейсу, адміністрування систем та розробки наукового ПЗ, зокрема систем машинного навчання.

Google Colaboratory – сервіс Jupyter Notebook, який знаходиться на хостингу компанії Google та надає власні обчислювальні ресурси, включно із GPU та TPU варіантами, для запуску програмного коду, зокрема для

проведення навчання моделей машинного навчання. Даний сервіс є зручним для проведення досліджень у сфері машинного навчання, data science тощо, і має безкоштовний доступ до певної кількості ресурсів для досліджень.

NumPy – бібліотека з відкритим вихідним кодом для мови програмування Python, що дає підтримку роботи з великими, багаторозмірними масивами та матрицями і математичними функціями, що оперують ними. На використанні даної бібліотеки спираються більшість інших дослідницьких бібліотек та додатків.

Pandas – бібліотека програмного забезпечення на мові програмування Python для маніпулювання даними та їх аналізу через надання структур даних та операцій з керування числових таблиць та часових рядів. Розроблена поверх бібліотеки NumPy, Pandas вводить багато порівнянних функцій роботи з DataFrames, які були встановлені в мові програмування R.

NLTK, аббревіатура від Natural Language Toolkit – набір бібліотек і програм для статистичної обробки природної мови для англійської мови на мові програмування Python. Даний набір містить засоби для класифікації, токенізації, лематизації, синтаксичного аналізу, семантичного обґрунтування, зокрема розроблені на основі технологій машинного навчання.

Scikit-learn – безкоштовна бібліотека для розробки систем машинного навчання на мові Python, що містить різноманітні алгоритми та моделі для виконання регресії, класифікації, кластеризації включно з опорно-векторними машинами, деревами рішень та випадковими лісами, нейронними мережами тощо, а також алгоритми для попередньої обробки даних. Дана бібліотека призначена для взаємодії з чисельними та науковими бібліотеками NumPy та SciPy.

Tensorflow – бібліотека з відкритим вихідним кодом, розроблена Google, для розробки систем машинного навчання, зокрема попередньої обробки даних, побудови та тренування моделей штучних нейронних мереж. Влаштована бібліотека **Keras** надає широкий спектр алгоритмів, налаштувань, алгоритмів та шарів нейронів до використання в розробці.

Gensim – бібліотека з відкритим вихідним кодом для неконтрольованого моделювання тем, індексування документів, пошуку за схожістю та інших функцій обробки природної мови з використанням сучасного статистичного машинного навчання.

pickle – модуль серіалізації та десеріалізації об'єктів, зокрема, навчених моделей машинного навчання та налаштованих об'єктів обробки даних, що дозволяє виконувати збереження прогресу навчання, виконання навчання окремо від платформ та пристроїв кінцевого використання та резервного копіювання.

Matplotlib – бібліотека на мові програмування Python для створення статичних, анімованих та інтерактивних візуалізацій даних, зокрема у вигляді графіків, матриць тощо. Завдяки візуалізації сприйняття результатів навчання стає простішим, через що використання даної бібліотеки є важливою складовою аналізу результатів даного дослідження.

Окрім описаного інструментарію, можливе використання додаткових сторонніх бібліотек, які доповнюють вказані та додають функціонал або надають інші моделі машинного навчання до використання.

Висновки до першого розділу

Протягом проведення аналізу предметної області було визначено головні задачі щодо виконання подальшого дослідження за темою та сформовано завдання.

Протягом дослідження наукових вітчизняних та міжнародних джерел за предметною областю та темою дослідження було вивчено історію розвитку технологій машинного навчання та обробки природної мови, зокрема появи сучасного підходу до створення алгоритмів та моделей на їх основі, детально розглянуто головні парадигми машинного навчання та задачі, які вони виконують, та можливості з виконання аналізу текстів засобами машинного навчання.

Було виконано пошук та огляд, аналіз існуючих аналогів, а саме проаналізовано виконання аналізів текстів інструментами, що надають обрані

до огляду сервіси. Завдяки виконаному аналізу було доповнено орієнтири щодо подальших етапів імплементації власної системи.

Дослідження засобів розробки програмного забезпечення за темою та предметною областю дозволило виділити окремі необхідні інструменти, програмне забезпечення та бібліотеки для виконання розробки необхідного програмного забезпечення згідно можливостей оглянутих бібліотек.

РОЗДІЛ 2. ДЕТАЛЬНИЙ АНАЛІЗ ТЕХНОЛОГІЙ ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Пошук та аналіз набору даних

Для виконання дослідження методів машинного навчання та попередньої обробки тексту необхідно мати набір даних, який підійде для використання навчання системи та матиме велику кількість даних. Для пошуку датасету, який можна використати в процесі дослідження, було використано такі сервіси, як Papers With Code, Hugging Face та Kaggle. В результаті було знайдено датасет есе, написаних до міжнародної системи тестування англійської мови (IELTS) [30].

IELTS є однією з найбільших систем тестування, яку використовують для оцінювання знань англійської мови при переселенні до США, Великобританії, Канади, Австралії та Нової Зеландії, а також для вступу з-за кордону у заклади вищої освіти у цих країнах. Дана система має кілька типів тестування, з яких відзначаються Академічний (Academic, Task 2) та Загальної підготовки (General Training, Task 1).

Обраний датасет містить есе типу Академічний (Task 2) та пов'язану із ними інформацію, як питання до опрацювання, есе по темі питання, характеристика есе, створена оцінювачем від організації IELTS, і підсумкова оцінка, що містить значення від 9 та 8.5 до 4 та "<4", де "<4" включає у себе всі есе, що відображають слабкий рівень знань автора.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	prompt,essay,evaluation,band																
2	Interviews form the basic criteria for most large companies. However, some people think that the interview is not a reliable method of choosing whom to employ and there are other better methods. To w																
3	To begin with, an interview enables the recruiter to know the kind of person he or she is recruiting. It helps the employer to see the personality traits of the employee such as how he answers questions, h																
4	Furthermore, recruiters also assess the applicant's ability to solve problems when they arise. A good idea generated or how one handles situations can bring great development to the company. For instanc																
5	On the other hand, other methods such as CVs, cover letters, the use of only certificates and many more are not a suitable step to recruit an applicant due to the fact that it does not allow the recruiter to se																
6	To sum up, I think an interview is still the most reliable practice of hiring employees rather than using other methods. So I suggest managers use only interviews as a means of sourcing workers for their cor																
7	The essay effectively addresses the given task. The candidate clearly states their position in the introduction and provides relevant arguments and evidence to support their stance. However, the essay lack																
8	**Coherence and Cohesion: [7.5]**																
9	The essay is well-organized and easy to follow. Transitions between sentences and paragraphs are clear, and connecting words and phrases are used effectively to maintain a smooth progression of ideas. H																
10	**Lexical Resource: [7]**																
11	The candidate demonstrates a good range of vocabulary. The majority of the words are used accurately, although there are a few instances of repetition and overused phrases. The vocabulary is appropriat																
12	**Grammatical Range and Accuracy: [7]**																
13	The essay exhibits a variety of sentence structures, including complex and compound sentences. However, there are a few grammatical errors, such as incorrect verb tenses and missing articles. The use of																
14	**Overall Band Score: [7.5]**																
15	The essay effectively addresses the given task, demonstrates good coherence and cohesion, and uses a range of vocabulary. However, there are some grammatical errors and a lack of depth in exploring the																
16	**Feedback and Additional Comments:**																

Рис.2.1 – Необроблений вигляд даних файлу test.csv датасету

При обробці подальшій даного датасету необхідно визначення елементів до видалення, а саме виконати пошук рядків із пустими або дубльованими значеннями, та виконати загальну очистку та обробку. В подальшому, ці дані будуть використані для навчання моделей після виконання необхідної підготовки.

2.2. Опис архітектури програмного забезпечення дослідження

Для розробки та виконання коду буде використано сервіс Google Colaboratory, який надає ресурси, такі як GPU/TPU, ОЗП та передвстановлені бібліотеки із можливістю дозавантаження нових, для виконання обчислень моделями машинного навчання. Завдяки цьому сервісу, стає можливим розробка за відсутності потужного апаратного забезпечення, необхідного для роботи алгоритмів.

У процесі попередньої обробки документів вхідних даних, будуть використані описані у попередньому розділі алгоритми та методи обробки. Зокрема, з урахуванням описаних недоліків стеммінгу та лематизації, для дослідження буде використано метод лематизації через можливість часткового вирішення проблеми потребив обчислювальних ресурсах завдяки наявності великої кількості ресурсів у розпорядженні при користуванні засобом розробки Google Colab.

В рамках даного дослідження найефективнішим методом векторизації можна вважати Doc2Vec, модель DM, яку і буде використано в подальшій розробці. При цьому, для ефективного використання ресурсів та інтеграції теми як контексту есе зі самим есе, ці незалежні змінні датасету будуть конкатеновані за принципом “Тема” + “ “ + “Есе” та векторизовані разом.

Вибір моделей машинного навчання залежить від цілі, яку має виконувати система. У даному випадку, датасет містить залежні дані у вигляді оцінки есе по системі IELTS, що звужує вибір до використання парадигми контрольованого навчання. Зокрема, через наявність значень від 9 до 4 із проміжними (таких, як 8.5, 5.5, 6.5) та окремого класу для оцінок нижче ніж 4, логічним є використання моделей класифікації.

Зокрема, для дослідження ефективності різних моделей машинного навчання, буде використано моделі SVM та XGBoost та спроектовано модель штучної нейронної мережі та рекурентної нейронної мережі.

SVM – метод машинного навчання з учителем, який використовує метод опорних векторів для визначення якомога більшого проміжку між найближчими одне до одного одиницями даних різних класів і згідно виконаного визначення – прокласти чітке розділення. Окрім цього, завдяки використанню ядрових методів стає можливим виконання ядрового трюку для даних, що не можуть бути розділені лінійною залежністю, а саме він дозволяє без обчислення координат даних у неявному просторі ознак (більшому в розмірності на 1 у порівнянні з розмірністю вхідних даних) виконувати всі обчислення у початковому просторі, спрощуючи задачу моделі. Одними з найпопулярніших ядер є поліноміальне, сигмоїдальне та Гаусове ядро або ядро радіальної базисної функції (RBF kernel) [31].

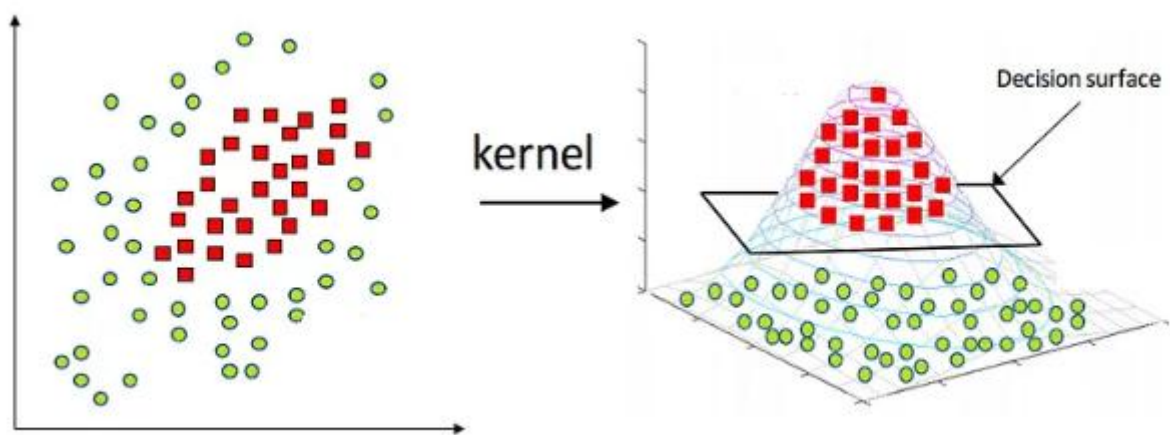


Рис.2.2 – Візуалізація роботи класифікатора ядрового SVM на датасеті з 2х незалежних змінних

Extreme Gradient Boost, або XGBoost, це ансамблева модель машинного навчання, яка заснована на системі градієнтного підсилення (gradient boost) та відома своєю гнучкістю, швидкістю роботи та ефективністю. Алгоритм підсилення будує окремі внутрішні слабкі моделі, які вчаться на часткових даних та разом виконують передбачення на вхідних даних у роботі

усереднюючи результати по окремим моделям, таким чином підсилюючи кінцевий результат. При ініціації моделі встановлюється кількість градієнтно-підсилених дерев, що побудовані паралельно, а під час роботи виконує регуляризацию L1 та L2 на складніших моделях [32].

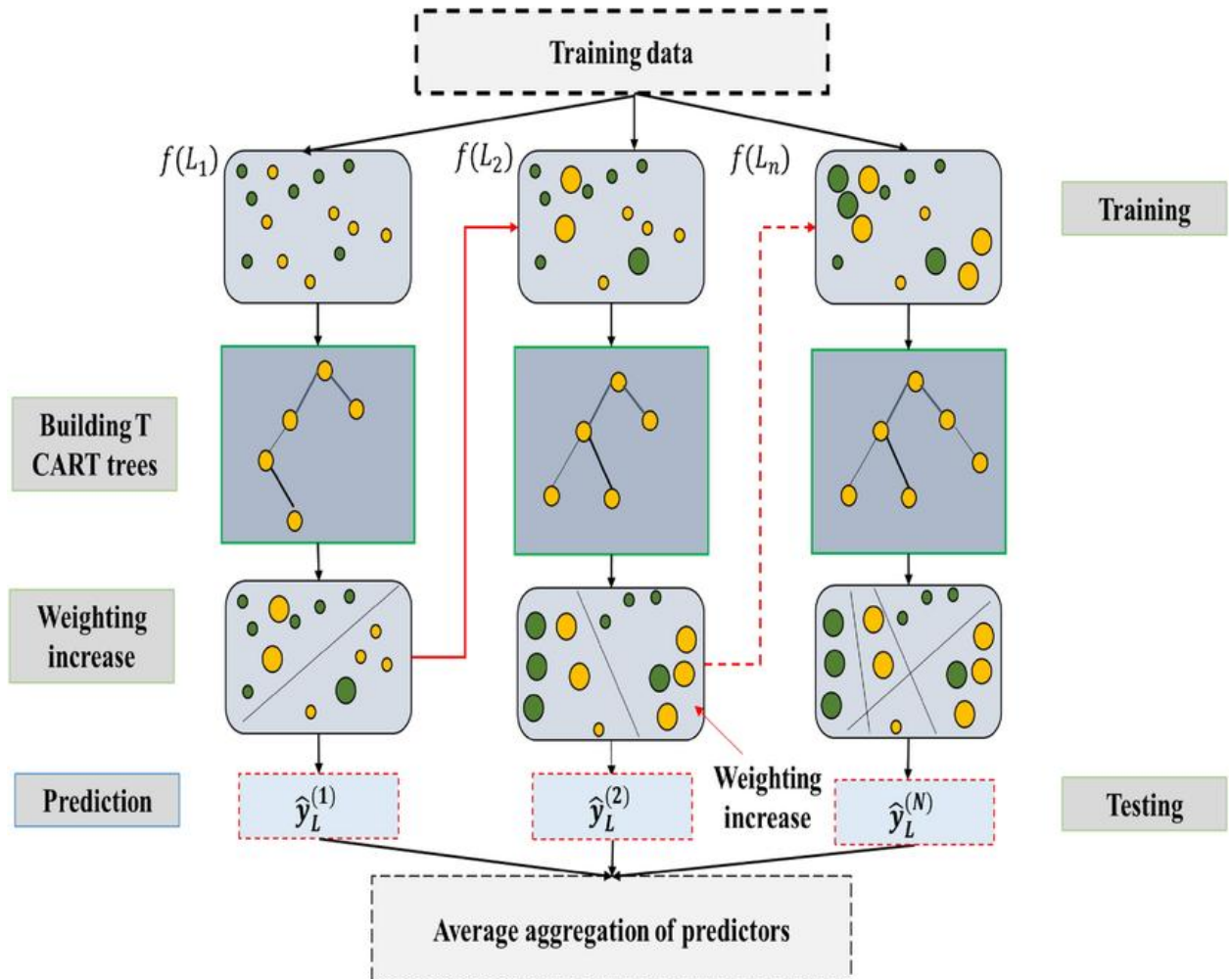


Рис.2.3 – Візуалізація роботи моделі XGBoost

Регуляризація L1 та L2, відомі як Lasso (Least Absolute Shrinkage and Selection Operator) та Ridge дозволяють вирішити, або спростити разом із іншими методами, проблему перенавчання моделей, тобто коли модель не здатна надавати позитивний результат на нових відносно навчальних даних [33]. Вони використовуються для вхідних даних із великою кількістю незалежних змінних, яких у даному дослідженні буде багато через виконання векторизації у розмірі від однієї до кількох сотень елементів векторів.

При побудові штучних нейронних мереж, буде використано бібліотеку Keras (вбудовану в бібліотеку Tensorflow), а саме модель Sequential, тобто послідовну модель шарів мережі, та такі шари, як:

Dense – шар повнозв’язних нейронів, що отримують дані з усіх нейронів попереднього шару.

Batch Normalization – шар нормалізації вхідних даних [34], який використовується для збереження масштабування даних одне відносно одного.

Dropout – шар випадково відсікання вхідних даних, таким чином і самих нейронів, з попереднього шару; використання даного шару адресується до проблеми перенавчання та є одним з рішень або комбінованих методів.

Зокрема, буде встановлено функції регуляризації даних L2 та використано активаційну функцію ReLU або її підвид за потреби вирішення проблеми “помираючого ReLU” [35][36], що полягає у відмові оновлення ваг нейронів або активації певних нейронів.

Основними метриками для задач класифікації, а відповідно і до результатів створених моделей є Accuracy, Precision, Recall та F1-score [37]. Такі метрики, як Precision, Recall та F1-score, використовуються в першу чергу для двійкових класифікацій із використанням True Positive, False positive, True Negative та False Negative значень, але у випадку багатокласової задачі можливе виконання усереднення значень по окремих класах. Використання Accuracy дозволяє визначити відношення між вірними передбаченнями та справжніми значеннями. Precision вказує на відношення TP (True Positive, вірні “позитивні” вгадування причетності до класу) до суми всіх “позитивних” вгадувань (TP + FP). Recall вказує на відношення TP до суми всіх “позитивних” реальних значень. F1-score обраховується за формулою:

$$F1 \text{ Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

і вважається однією з кращих метрик для визначення ефективності моделі.

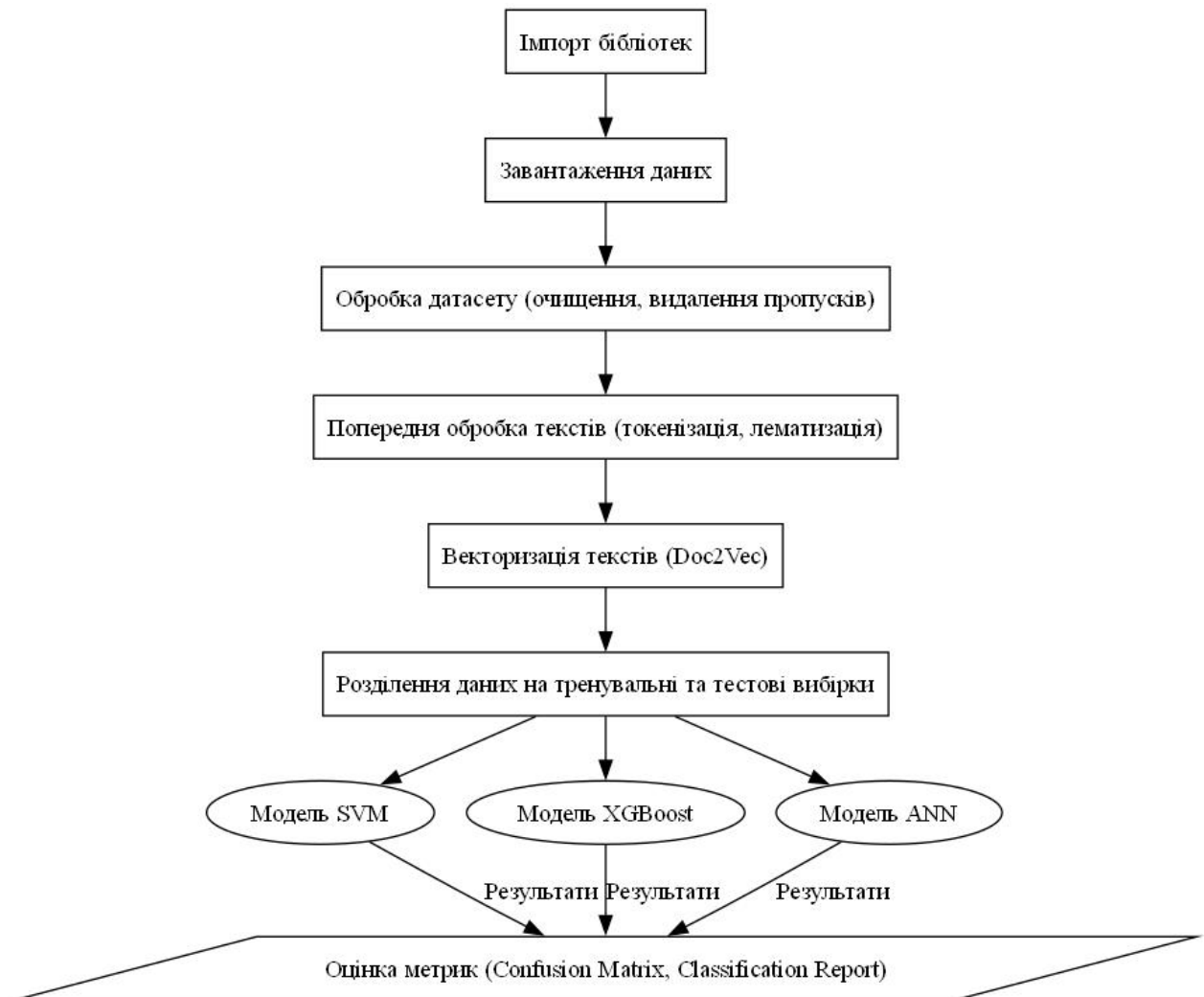


Рис.2.4 – Блок-схема роботи програмного забезпечення згідно опису архітектури ПЗ

Згідно виконаного дослідження алгоритмів попередньої обробки текстових даних та описаної архітектури ПЗ, зокрема моделей до використання, майбутнє програмне забезпечення виконуватиме роботу послідовно згідно блок-схеми на рис.2.4. Кінцевий етап порівняння результатів обраних до дослідження моделей виконуватиметься на основі отриманих метрик класифікації окремих моделей.

2.3. Підготовка середовища розробки та попередня розробка ПЗ

Першим етапом виконання підготовки до розробки ПЗ є імпортування необхідних програмних компонентів та бібліотек. Окрім описаних бібліотек до використання, для зручності виконання навчання, збереження моделей та встановлення однієї точки завантаження/вивантаження, було використано Google Drive та відповідну функцію бібліотеки Google Colab для доступу до хмарного сховища.

Код імпортування бібліотек:

```
from google.colab import drive
import os.path
import numpy as np
import pandas as pd
from tqdm import tqdm
import matplotlib.pyplot as plt

import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

import gensim
from gensim.parsing.preprocessing import strip_tags, \
strip_multiple_whitespaces, \
strip_punctuation, remove_stopwords
from gensim.models.doc2vec import Doc2Vec, TaggedDocument

from sklearn.metrics import accuracy_score, classification_report, \
confusion_matrix, ConfusionMatrixDisplay, roc_curve, auc
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.utils.class_weight import compute_class_weight
from sklearn.svm import SVC

from xgboost import XGBClassifier

import tensorflow as tf

import string
import re
import pickle
```

Зокрема, серед додаткових

Для коректної обробки текстів необхідно завантаження пакунків бібліотеки NLTK. Перед завантаженням даних з файлів датасету через бібліотеку Pandas ініціюватиметься доповнення завдяки TQDM, що додає до функціоналу Pandas можливість виведення прогресу виконання функцій [38][39].

Код завантаження пакунків та активацію додаткових функцій TQDM:

```

nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
nltk.download('wordnet')
nltk.download('tagsets')
nltk.download('universal_tagset')

tqdm.pandas()

```

В подальшому буде розроблено основний функціонал програмного забезпечення дослідження, а саме буде добуто дані з датасету, виконано попередню обробку та векторизацію, розділення датасету на набори даних для тренування, валідації та тестування, а також імплементовано код навчання та перевірки моделей машинного навчання згідно описаних метрик.

Висновки до другого розділу

Протягом виконання другого розділу було проаналізовано вхідні дані системи (датасет), а на основі виконаного аналізу – встановлено відповідну парадигму та задачу алгоритмів машинного навчання, обрано моделі та описано попередній етап розробки.

Пошук датасету для подальшого використання в навчанні моделей було виконано на таких сервісах та ресурсах, як Papers With Code, Hugging Face та Kaggle. При пошуку датасету, головними критеріями до пошуку були однорідність даних, величина датасету та незалежність даних між собою. Наявність залежної змінної дозволила обрати як парадигму контрольоване навчання, а структура даних та унікальні значення змінної – задачу класифікації.

Згідно аналізу датасету, було описано початкові етапи роботи програмного забезпечення, а саме обрано алгоритми обробки датасету, описано моделі машинного навчання до дослідження їх ефективності в даній задачі, зокрема описано архітектуру моделі штучної нейронної мережі.

Виконане проектування та попередня розробка імпортування бібліотек, налаштування пакунків дозволяє перейти до розробки основного функціоналу програмного забезпечення та аналізу результатів навчання моделей.

	prompt	essay	evaluation	band
count	9833	9833	9833	9833
unique	3468	8985	9833	252
top	Some people believe that teenagers should be r... Over the past decades, environmental problems ...	### Task Achievement: 6.5\n- The essay adequat...		7
freq	158	6	1	271

Рис.3.3 – Опис датасету функцією describe()

Згідно отриманої інформації, було визначено, що рядків чи стовпців із пропущеними даними немає, у той час як через наявність неунікальних есе важливим є видалення повних дублікатів (включно з порівнянням есе у них). Окрім цього, для уніфікації даних для подальшої роботи необхідно очистити стовпець band від упущених символів \n та \r.

Код попередньої обробки датасету:

```
dataset = dataset.drop_duplicates()
dataset = dataset.replace(r'(\r*\n*)*', '', regex=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9833 entries, 0 to 9832
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  -
0   prompt      9833 non-null   object
1   essay       9833 non-null   object
2   evaluation  9833 non-null   object
3   band        9833 non-null   object
dtypes: object(4)
memory usage: 307.4+ KB
```

Рис.3.4 – Інформація по датасету після очистки

	prompt	essay	evaluation	band
count	9833	9833	9833	9833
unique	2934	8809	9833	12
top	Some people believe that teenagers should be r... Over the past decades, environmental problems ...	### Task Achievement: 6.5- The essay adequatel...		7
freq	162	6	1	1358

Рис.3.5 – Опис датасету після очистки

Помітною зміною в датасеті є суттєве зменшення унікальних значень стовпця оцінки як залежної змінної, який тепер містить 12 замість 252 значень. Зокрема, ці значення становлять оцінки від 9 та 8.5 до 4 та менше 4 (“<4”).

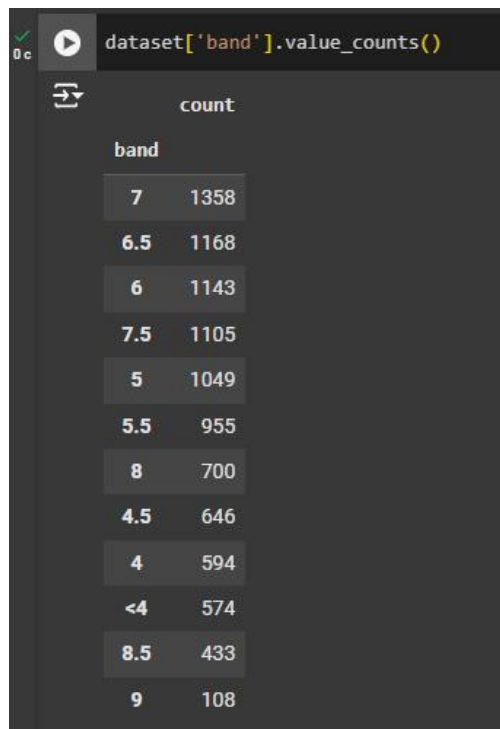


Рис.3.6 – Підрахування кількості даних згідно унікальних значень залежної змінної

Після виконаної попередньої обробки датасету, наступним кроком є обробка самих текстових даних для їх адаптації під єдиний сприйнятний для моделей машинного навчання формат – вектори чисел із незмінними розмірами. Такі етапи обробки, як зменшення регістру, видалення всіх символів окрім літер латинського алфавіту, необхідні для коректного виконання виключення стоп-слів та лематизації. Лематизація виконується завдяки алгоритму WordNetLemmatizer [40], що заснований на використанні лексичної бібліотеки WordNet Принстонського університету [41].

Код функції попередньої обробки текстів:

```
lemmatizer = WordNetLemmatizer()
def preprocess(text: str, tokenize=True, no_stopwords=True, lemmatize=True) -
> str:
    text = text.strip()
    text = strip_tags(text)
    text = strip_multiple_whitespaces(text)
    text = re.sub(r'\.\s*([A-Z])', r'. \1', text)

    if (not tokenize):
        return text # Returns cleaned text

    text = text.lower()
    text = strip_punctuation(text)

    if (no_stopwords):
        tokens = remove_stopwords(text)
```

```

tokens = word_tokenize(text)

if (lemmatize):
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

return ' '.join(tokens) # Returns preprocessed by conditions sequence of
words from text

```

Дана функція отримує на вході окремий текст та умови у випадку необхідності виконання часткової обробки (за замовчуванням виконуються усі етапи), повертаючи оброблений текст. Метод обробки текстів застосовується до всього текстового датасету, зокрема для цього було об'єднано стовпці питань та есе. Це дозволяє включити контекст есе до текстів та в подальшому натренувати спільну модель векторизації. Оброблені тексти зберігаються разом зі вхідним датасетом.

Код обробки текстових даних датасету:

```

texts = dataset['prompt'] + dataset['essay']
preprocessed_texts = texts.progress_apply(preprocess)
dataset.insert(2, "preprocessed_text", preprocessed_texts)
data = np.array(preprocessed_texts)

```

Приклад об'єднаних текстових даних після виконання обробки:

```

interview form the basic criterion for most large company however some people
think that the interview is not a reliable method of choosing whom to employ
and there are other better method to what extent do you agree or disagree it
is believed by some expert that the traditional approach of recruiting
candidate which is interviewing is the best way whereas others think
different method such a exam writing cv cover letter or application letter
and many more are good i strongly agree with the statement interview is the
most reliable approach to recruit worker because this method assist the
recruiter to know the person and his ability to do the work and their problem
solving ability to begin with an interview enables the recruiter to know the
kind of person he or she is recruiting it help the employer to see the
personality trait of the employee such a how he answer question his facial
mannerism and also his communication skill that is whether introvert or
extrovert also his teamwork skill is measured during the dialogue for
instance job like sale personnel require good communication skill to be able
to do the work effectively and efficiently so interview allow the manager to
assess whether or not the applicant qualifies for the job furthermore
recruiter also assess the applicant s ability to solve problem when they
arise a good idea generated or how one handle situation can bring great
development to the company for instance pressure can put fear into an
employee which can make him make a wrong decision that can bring loss to the
company while some too can take pressure in a calm action and make a good
decision on the other hand other method such a cv cover letter the use of
only certificate and many more are not a suitable step to recruit an
applicant due to the fact that it doe not allow the recruiter to see the full
potential of the candidate information found in the cv or cover letter may
not be true because people lie to obtain what they desire in the same way a
candidate can also lie to acquire the position to sum up i think an interview
is still the most reliable practice of hiring employee rather than using
other method so i suggest manager use only interview a a mean of sourcing

```

worker for their company

Останнім етапом перед виконанням векторизації необхідно повторно токенізувати тексти та виконати встановлення тегів методом `TaggedDocument`, що є необхідним для створення словника слів моделі векторизації, навчання моделі.

Код токенизації та тегування по номеру тексту в масиві:

```
for i, row in enumerate(data):
    data[i] = word_tokenize(row)
tagged_texts = [TaggedDocument(words=text, tags=[i]) for i, text in
enumerate(data)]
```

Після виконання обробки тексту, відбувається векторизація текстів із навчанням та застосуванням моделі `Doc2Vec` [42]. Зокрема, відбувається збереження даної моделі для можливості подальшого використання в системі без необхідності повторного навчання. У процесі дослідження даної моделі було виявлено оптимальний розмір векторів у 300 одиниць, файл із моделлю зберігає цю специфікацію в назві.

Код створення, навчання та збереження або завантаження навченої моделі векторизації:

```
vector_size = 300
if os.path.isfile(path + f'vectorizer_{vector_size}.pkl'):
    with open(path + f'vectorizer_{vector_size}.pkl', 'rb') as file:
        model_doc_to_vec = pickle.load(file)

    print(f"Vectorizer was loaded from file 'vectorizer_{vector_size}.pkl'")
else:
    print("Vectorizer created anew")

    model_doc_to_vec = Doc2Vec(vector_size=vector_size,
                               min_count=2, epochs=50)
    model_doc_to_vec.build_vocab(tagged_texts)
    model_doc_to_vec.train(tagged_texts,
                           total_examples=model_doc_to_vec.corpus_count,
                           epochs=model_doc_to_vec.epochs)

    with open(path + f'vectorizer_{vector_size}.pkl', 'wb') as file:
        pickle.dump(model_doc_to_vec, file)
    print(f"Vectorizer was saved into file 'vectorizer_{vector_size}.pkl'")
```

Код використання моделі на оброблених текстах навчальної вибірки:

```
document_vectors = np.array([model_doc_to_vec.infer_vector(
    word_tokenize(document)) for document in dataset.preprocessed_text])
```

Приклад виконаної векторизації першого тексту датасету винесено у додаток А.

Отримані вектори текстів представляють собою вхідними даними для майбутніх моделей машинного навчання, тобто є наборами незалежних змінних, що відповідають залежним змінним у вигляді оцінок до цих текстів. Для зручності їх використання, створюються стандартні змінні X та y , зокрема необроблені залежні змінні проходять процес встановлення ярликів до них для створення коректних для навчання значень у проміжку від 0 до 11, в сумі 12 ярликів. Для виконання цього застосовується метод `LabelEncoder` бібліотеки `scikit-learn`.

Код присвоєння стандартним змінним значень та встановлення ярликів:

```
X = document_vectors.copy()
y = np.array(dataset.band)
le = LabelEncoder()
y = le.fit_transform(y)
```

Відповідні дії щодо завантаження, обробки та підготовки даних на навчених моделях та підготованих методах відбуваються і для даних із файлу тестування моделей `test.csv`.

Код підготовки тестувальних даних:

```
test = pd.read_csv(path + "test.csv")
test = test.replace(r'(\r*\n*)*', '', regex=True)
test['band'] = test['band'].replace(r'.0', '', regex=True)
test = test.dropna(axis=1)

test_texts = test['prompt'] + test['essay']
test_texts_prep = test_texts.progress_apply(preprocess)
data_test = np.array(test_texts_prep)
test.insert(2, "preprocessed_text", test_texts_prep)

test_vectors = np.array([model_doc_to_vec.infer_vector(
word_tokenize(document)) for document in test.preprocessed_text])
X_test = test_vectors.copy()
y_test = le.transform(np.array(test.band))
```

Для отримання навчання моделей машинного навчання вхідні дані розділяються на власне тренувальну та валідаційну вибірки, де валідаційна вибірка складає 10% від вхідних.

Код розділення вхідних даних на тренувальну та валідаційну вибірки:

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.1)
```

У результаті підготовки даних до роботи, маємо 3 вибірки: тренувальні дані у розмірі 8849, валідаційні у розмірі 984 та тестувальні у розмірі 941 записів.

3.2. Розробка та навчання моделей машинного навчання та моделі штучної нейронної мережі

Для порівняння ефективності роботи різних методів машинного навчання було вирішено використати такі моделі, як SVM для класифікації та XGBoost, а також розробити послідовну нейронну мережу із використанням повнозв'язних шарів та додаткових засобів для зменшення можливості перенавчання системи.

Створення моделі SVM для класифікації використовує бібліотеку `scikit-learn` та модель `SVC` з ядром `RBF`, збереження та завантаження моделі відбувається шляхом використання бібліотеки `pickle`.

Код створення, навчання та збереження або завантаження моделі `SVC`:

```
svm_path = path + f'classifier_svm_{vector_size}.pkl'

if os.path.isfile(svm_path):
    print(f"SVC model was loaded from file '{svm_path}'")
    with open(svm_path, 'rb') as file:
        svm_classifier = pickle.load(file)
else:
    print("SVC model created anew")
    svm_classifier = SVC(kernel='rbf')
    svm_classifier.fit(X_train, y_train)

    with open(svm_path, 'wb') as file:
        pickle.dump(svm_classifier, file)
    print(f"SVC model was saved into file '{svm_path}'")
```

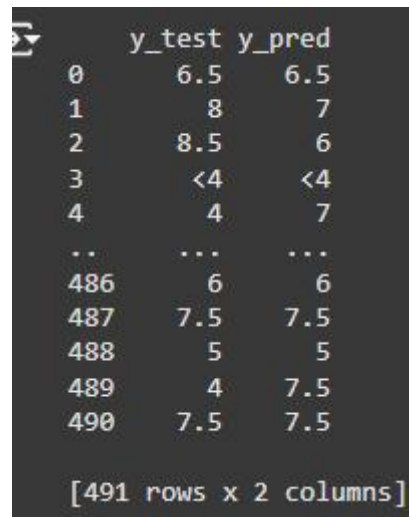
Модель використовує оброблені незалежні та залежні змінні з тренувальної вибірки для навчання, після чого зберігається в окремому файлі. У випадку існування файлу моделі, виконується її завантаження та присвоєння відповідній змінній для подальшої роботи. Всі параметри зберігаються через серіалізацію даних, при цьому для відновлення моделі через десеріалізацію необхідно, щоб програма підтримувала відповідні бібліотеки, що використано. При створенні файлу з моделлю враховується розмір векторів, що вказано в назві, адже при налаштуванні моделі вона здатна приймати лише параметри із тими самими характеристиками, що і при навчанні.

Для отримання метрик ефективності моделі, виконаємо передбачення на основі тестових даних та виконаємо порівняння результатів із відповідними справжніми даними.

Код передбачення результатів моделлю SVC та їх порівняння:

```
y_pred = svm_classifier.predict(X_test)

df_class = pd.DataFrame({"y_test": le.inverse_transform(y_test),
                        "y_pred": le.inverse_transform(y_pred)})
print(df_class)
```



	y_test	y_pred
0	6.5	6.5
1	8	7
2	8.5	6
3	<4	<4
4	4	7
..	...	...
486	6	6
487	7.5	7.5
488	5	5
489	4	7.5
490	7.5	7.5

[491 rows x 2 columns]

Рис.3.7 – Порівняння справжніх даних та результатів передбачення моделлю SVC

Аналіз метрик стає можливим із використанням методів `classification_report`, що надає у вигляді тексту метрики за окремими класами та усереднені, та `confusion_matrix`, `ConfusionMatrixDisplay`, які можна використати для графічного відображення результатів. Відповідний аналіз буде зображено у наступному розділі разом із порівнянням моделей.

Наступною моделлю до створення, навчання та аналізу є модель `XGBClassifier`. Основними параметрами до даної моделі є кількість дерев усередині (`n_estimators`), ціль, де визначається якого типу виконуватиметься класифікація: бінарна чи багатокласова, і швидкість навчання, яка регулює зміну ваг та параметрів протягом навчання. Дана модель має можливість використання додаткових датасетів для валідації, у даному випадку використано валідаційну вибірку.

Код створення, навчання та збереження або завантаження моделі XGBClassifier:

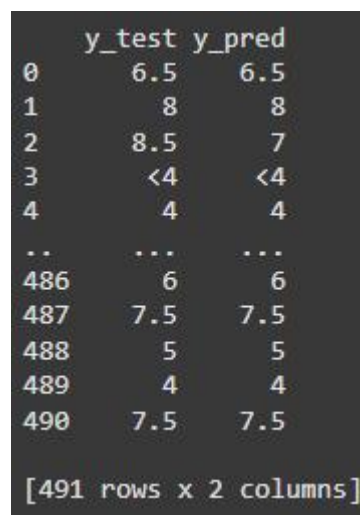
```
xgboost_path = path + f'classifier_xgb_{vector_size}.pkl'

if os.path.isfile(xgboost_path):
    print(f"XGB model was loaded from file '{xgboost_path}'")
    with open(xgboost_path, 'rb') as file:
        xgb_classifier = pickle.load(file)
else:
    print("XGB model created anew")

    xgb_classifier = XGBClassifier(n_estimators=75, objective="multi:softprob",
learning_rate=0.2)
    xgb_classifier.fit(X_train, y_train, eval_set=[(X_val, y_val)])

    with open(xgboost_path, 'wb') as file:
        pickle.dump(xgb_classifier, file)
    print(f"XGB model was saved into file '{xgboost_path}'")
```

Відповідно до попереднього прикладу, так само виконується передбачення результатів, порівняння і подальше відображення метрик.



	y_test	y_pred
0	6.5	6.5
1	8	8
2	8.5	7
3	<4	<4
4	4	4
..	...	...
486	6	6
487	7.5	7.5
488	5	5
489	4	4
490	7.5	7.5

[491 rows x 2 columns]

Рис.3.8 – Порівняння справжніх даних та результатів передбачення моделлю XGBClassifier

Для ефективного використання ресурсів та якісного навчання нейронна мережа окрім параметрів L2 регуляризації та шарів Dropout використовує callback-и, а саме:

EarlyStopping: дозволяє закінчувати процес навчання за виявлення плато на валідаційних даних, тобто за відсутності суттєвого розвитку чи регулярне погіршення метрик.

ModelCheckpoint: зберігає найкращі ваги в окремій моделі після кожної епохи та дозволяє завантажувати їх після навчання.

Окрім цього, використовується планувальник експоненційного занепадання, що зменшує швидкість навчання за певним коефіцієнтом по встановленим параметрам, та оптимізатор AdamW, що є версією оптимізатора Adam з додатковим значенням занепадання ваг нейронів. Використання цих методів оптимізації та налаштування нейронної мережі дозволяє адресувати проблему перенавчання разом із використаними шарами нейронів та регуляризациєю.

Код налаштування callback-ів та створення моделі нейронної мережі:

```
early_stopping_callback = tf.keras.callbacks.EarlyStopping(patience = 10,
restore_best_weights = True)

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=0.002,
    decay_steps=500,
    decay_rate=0.97
)

model_checkpoint_callback =
tf.keras.callbacks.ModelCheckpoint('ann_best_model.keras',
save_best_only=True)

ann_path = path + f'classifier_ann_{vector_size}.keras'

if os.path.isfile(ann_path):
    ann_model = tf.keras.models.load_model(ann_path,
custom_objects=tf.keras.layers.ELU)
    print(f"ANN model was loaded from file '{ann_path}'")
else:
    print("ANN model created anew")
    ann_model = tf.keras.models.Sequential([
        tf.keras.layers.Dense(256, activation=tf.keras.layers.ELU(alpha=0.01),
            input_dim=X_train.shape[4],
            kernel_regularizer=tf.keras.regularizers.l2(0.016)
        ),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(256, activation=tf.keras.layers.ELU(alpha=0.01),
            kernel_regularizer=tf.keras.regularizers.l2(0.016)
        ),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Dropout(0.4),
        tf.keras.layers.Dense(64, activation=tf.keras.layers.ELU(alpha=0.01),
            kernel_regularizer=tf.keras.regularizers.l2(0.016)
        ),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Dropout(0.4),
        tf.keras.layers.Dense(32, activation=tf.keras.layers.ELU(alpha=0.01),
            kernel_regularizer=tf.keras.regularizers.l2(0.016)
        ),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(len(np.unique(y_train)), activation='softmax')
    ])

```

```
ann_model.compile(optimizer=tf.keras.optimizers.AdamW(learning_rate=lr_schedule, weight_decay=0.01),
                  loss='categorical_crossentropy',
                  metrics=['AUC',
                           tf.keras.metrics.CategoricalAccuracy(name='accuracy'),
                           tf.keras.metrics.Precision(name='precision'),
                           tf.keras.metrics.Recall(name='recall'),
                           tf.keras.metrics.F1Score(name='f1_score')])
```

Збереження та завантаження моделі виконується влаштованими методами бібліотеки keras.

Модель використовує активаційну функцію softmax і таку метрику втрат, як категоріальна кросентропія, яка обраховує втрати в навчанні для багатокласових моделей, через що на вхід моделі необхідно подавати залежні змінні у вигляді розрідженого масиву категорій класів. Приведення до цієї форми виконується функцією `to_categorical`.

Передбачення, які робить модель, матимуть таку саму форму, але вміщуючи дробові шанси належності вхідних даних до тієї чи іншої категорії. Найбільший зі вказаних шансів вважається найімовірнішою відповіддю та зберігається в окремій змінній.

Окрім цього, для підвищення якості навчання виконується обрахунок ваг частоти тих чи інших класів. Налаштування кількості епох та розмірів партій вхідних даних впливає на максимальний час навчання та якість навчання.

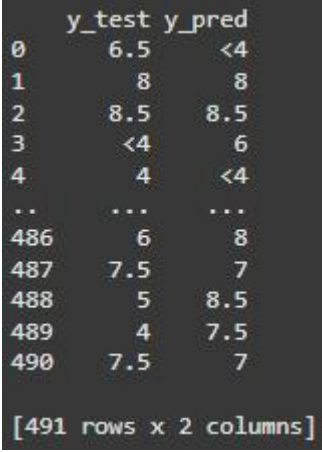
Код ініціалізації навчання моделі нейронної мережі із додатковими параметрами.

```
class_weights = compute_class_weight("balanced", classes=np.unique(y_train),
y=y_train)
if not os.path.isfile(ann_path):
    history = ann_model.fit(
        X_train, tf.keras.utils.to_categorical(y_train),
        validation_data=(X_val, tf.keras.utils.to_categorical(y_val)),
        class_weight=dict(enumerate(class_weights)),
        epochs=50, batch_size=64, shuffle=True,
        # callbacks=[early_stopping_callback, lr_scheduler_callback,
model_checkpoint_callback])
        # callbacks=[early_stopping_callback, reduce_lr_on_plateau_callback,
model_checkpoint_callback])
        callbacks=[early_stopping_callback, model_checkpoint_callback])
    ann_model.load_weights('ann_best_model.keras')

# with open(path + 'classifier_ann.pkl', 'wb') as f:
#     pickle.dump(ann_model, f)
ann_model.save(ann_path)
```

```
print(f"ANN model was saved into file '{ann_path}')
```

У процесі навчання відображення графіків дозволяє аналізувати модель на стан перенавчання, коли модель здатна ефективно працювати лише з даними навчальної вибірки та погано на інших даних. Якщо при збільшенні метрик на навчальній вибірці спостерігається так зване “плато” або навіть погіршення результатів – модель перенавчена та потребує доопрацювання [43].



	y_test	y_pred
0	6.5	<4
1	8	8
2	8.5	8.5
3	<4	6
4	4	<4
..	...	...
486	6	8
487	7.5	7
488	5	8.5
489	4	7.5
490	7.5	7

[491 rows x 2 columns]

Рис.3.9 – Порівняння справжніх даних та результатів передбачення моделлю штучної нейронної мережі

Виконавши навчання всіх застосованих моделей, маємо можливість виконати подальший аналіз результатів роботи на тестовій вибірці, а отримані метрики – порівняти між собою.

3.3. Аналіз результатів навчання та порівняння ефективності роботи алгоритмів

Для навчених моделей класифікації необхідно визначити їх метрики на незалежних даних, до яких відносяться дані тестової вибірки, досі не відомі моделям. Завдяки такому розділенню, перевірка якості навчання та роботи засобів машинного навчання набуває об’єктивності, необхідної для впевненості при порівнянні моделей.

Для отримання метрик класифікації на тестових даних, можна скористатися функцією `scikit-learn classification_report`. Дана функція дозволяє виконати обрахунок основних метрик класифікації будь-якого типу, як бінарного чи багатокласового. У випадку аналізу багатокласової

класифікації, вказані метрики обраховуються у формі “один-проти-всіх”, а підсумкові результати моделі по всіх класах обраховуються як середні значення метрик окремих класів [37]. Для отримання метрик навчених моделей, використаємо функцію на їх результатах передбачень та справжніх даних тестової вибірки.

Код обрахунку та виведення метрик навчених моделей машинного навчання:

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.86	0.55	0.67	33
1	0.94	0.53	0.68	30
2	0.63	0.83	0.72	35
3	0.80	0.62	0.70	58
4	0.61	0.66	0.63	58
5	0.67	0.67	0.67	67
6	0.49	0.83	0.62	76
7	0.55	0.68	0.61	41
8	1.00	0.43	0.60	35
9	1.00	0.44	0.61	16
10	1.00	0.50	0.67	4
11	0.90	0.71	0.79	38
accuracy			0.66	491
macro avg	0.79	0.62	0.66	491
weighted avg	0.72	0.66	0.66	491

Рис.3.10 – Виведення метрик роботи моделі SVC

Результати обрахування метрик моделі SVC показують, що в середньому модель здатна ефективно передбачати результати із вірогідністю 66-70%. Зокрема, метрики мають наступні значення:

- accuracy: 0.66 (66%);
- precision: середнє = 0.79, зважене = 0.72;
- recall: середнє = 0.62, зважене = 0.66;
- F1-score: середнє = 0.66, зважене = 0.66.

Дані результати вказують на необхідність тонкого налаштування моделі під вимоги задачі та датасет. Згідно поточних результатів, модель не є оптимальною для роботи.

Іншим способом візуалізації результатів навчання є отримання даних матриці невідповідностей. Для цього можливе використання функції `sklearn confusion_matrix`, а для графічної візуалізації – бібліотеки `matplotlib` та функції іншої функції `ConfusionMatrixDisplay`. У випадку бінарної класифікації отриману матрицю невідповідностей можна виокремити у значення True Positive, False Positive, True Negative, False Negative. У випадку багатокласової класифікації надається матриця вгаданих значень відносно окремих справжніх. Візуалізація дозволяє кольором на графіку вказати відносну частоту значень, а також надає можливість аналізу відхилень у навчанні моделі, як необ'єктивна робота та переважання одних передбачених значень над іншими.

Код створення та візуалізації матриці невідповідностей:

```
cm = confusion_matrix(y_test, y_pred)

fig, ax = plt.subplots(figsize=(8, 6))
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(ax=ax)

plt.title('Confusion Matrix SVC')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
```

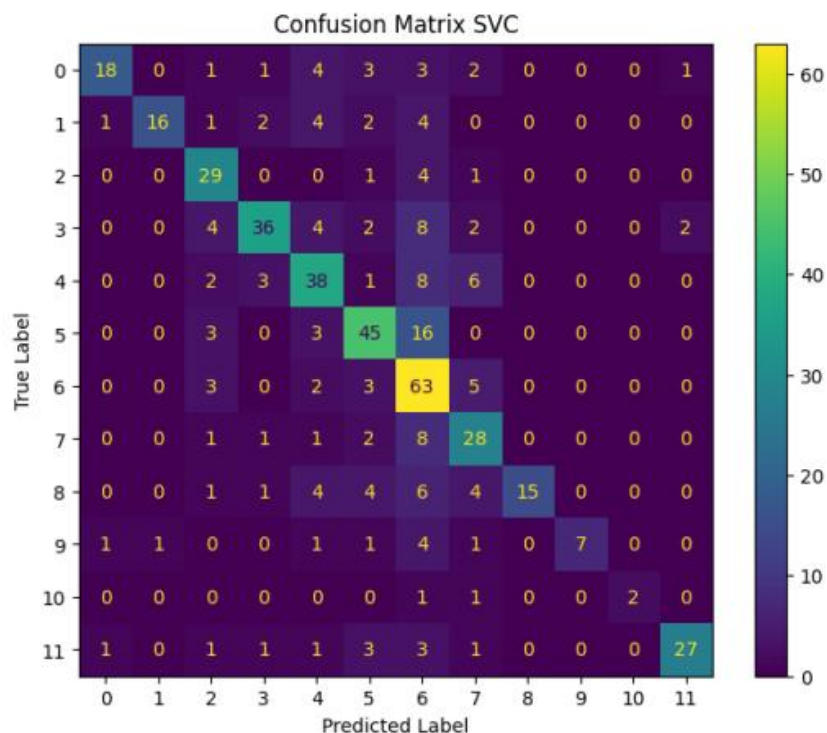


Рис.3.11 – Матриця невідповідностей для моделі SVC

Згідно графіку вище, можна побачити розподіл значень передбачень моделлю SVC. За ідеальних умов, де модель передбачає зі 100% якістю, значення матриці були б нульовими у всіх клітинках окрім головної діагоналі, що позначало б відсутність помилкових передбачень. У даному випадку, можемо спостерігати значну частку невірних передбачень та класи, які були передбачено натомість.

Наступною моделлю для аналізу є ансамблева модель XGBClassifier. Природа ансамблевих моделей дозволяє попередньо сподіватись на кращі результати за правильних налаштувань та підготовки даних через специфіку ансамблів – незалежне навчання менших моделей всередині більшої для усереднення їх результатів та спільного навчання або послідовне покращення результатів попередніх етапів.

	precision	recall	f1-score	support
0	0.96	0.79	0.87	33
1	0.96	0.73	0.83	30
2	0.76	0.91	0.83	35
3	0.89	0.86	0.88	58
4	0.83	0.84	0.84	58
5	0.83	0.87	0.85	67
6	0.84	0.88	0.86	76
7	0.71	0.85	0.78	41
8	1.00	0.89	0.94	35
9	1.00	0.88	0.93	16
10	1.00	0.50	0.67	4
11	0.92	0.92	0.92	38
accuracy			0.86	491
macro avg	0.89	0.83	0.85	491
weighted avg	0.87	0.86	0.86	491

Рис.3.12 – Виведення метрик роботи моделі XGBClassifier

Згідно отриманих метрик, дана модель надає набагато кращі результати роботи як по окремим класам, так і в середньому. Метрики даної моделі становлять:

- accuracy: 0.86;
- precision: середнє = 0.89, зважене = 0.87;
- recall: середнє = 0.83, зважене = 0.86;
- F1-score: середнє = 0.85, зважене = 0.86.

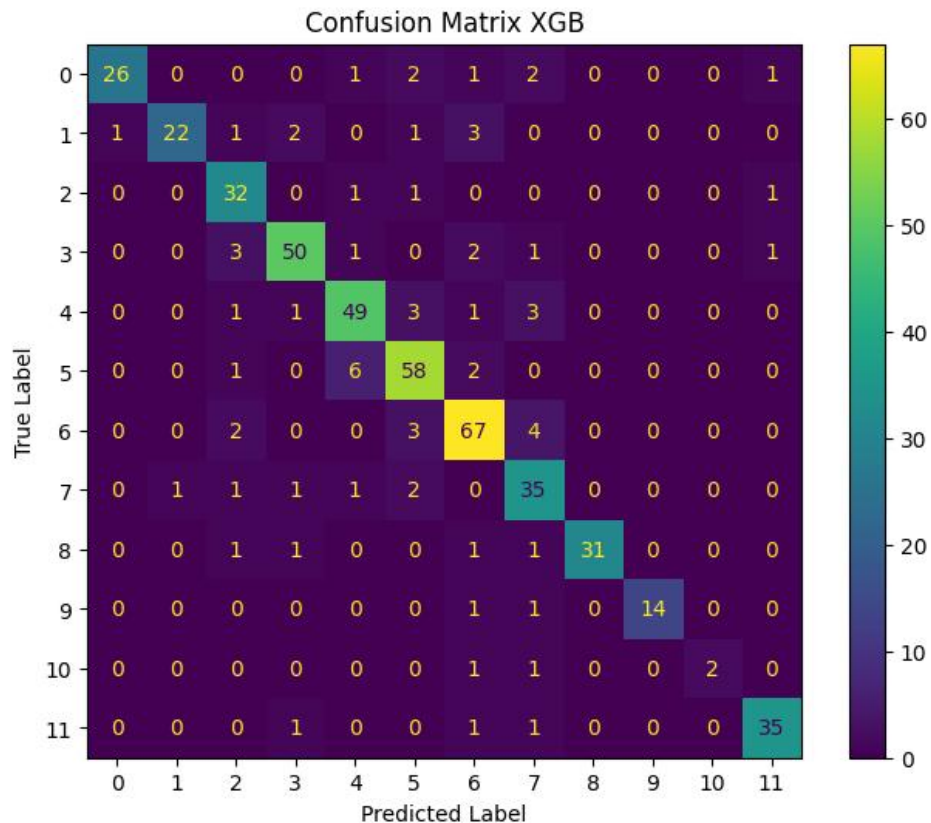


Рис.3.13 – Матриця невідповідностей для моделі XGBClassifier

Отримані метрики та відображена матриця невідповідностей вказують на високу ефективність використання ансамблевих методів машинного навчання на прикладі методу XGB. Більш тонке налаштування моделі дозволить їй отримати середню ефективність у 0.9 і більше, що вважається результатом добре налаштованої та натренованої моделі машинного навчання.

Останньою моделлю до аналізу є модель штучної нейронної мережі. Складності зі створення штучних нейронних мереж полягає у необхідності тонкого налаштування, проведення експериментів з параметрами та аналізі вихідних проміжних метрик протягом часто довгих процесів повторного навчання. Через широкі можливості використання різних оптимізаторів, активаційних функцій, шарів нейронів та параметрів до всіх зазначених елементів, нейронні мережі є складними у підготовці до якісної роботи, але потенційно набагато гнучкішими та потужнішими у виконанні задач. Зокрема, для штучних нейронних мереж можливе використання шарів із

функціоналом, що повністю замінює окремі етапи попередньої обробки даних або виконує операції з даними, що складно імплементувати звичайними способами.

Раніше було згадано можливість огляду прогресу навчання штучної нейронної мережі через відображення графіків навчальної кривої по окремим метрикам, а саме у випадку наявності плато або погіршенню результатів модель вважається перенавченою [43]. У процесі навчання за налаштування виводу прогресу можна вести моніторинг метрик та відповідне виведення валідаційних варіантів для порівняння між собою та отримання розуміння до кінця навчання.

```
Epoch 1/50
/usr/local/lib/python3.10/dist-packages/tensorflow/python/data/ops/structured_function.py:258: UserWarning: Even though the `tf.config.experimental_run_functions_eagerly`
warnings.warn(
139/139 ----- 39s 237ms/step - AUC: 0.5048 - accuracy: 0.0829 - f1_score: 0.0775 - loss: 13.2716 - precision: 0.0812 - recall: 0.0104 - val_AUC: 0.4776 - va
Epoch 2/50
139/139 ----- 35s 235ms/step - AUC: 0.5247 - accuracy: 0.0900 - f1_score: 0.0888 - loss: 10.2227 - precision: 0.0771 - recall: 0.0023 - val_AUC: 0.4839 - va
Epoch 3/50
139/139 ----- 41s 234ms/step - AUC: 0.5370 - accuracy: 0.1006 - f1_score: 0.0985 - loss: 8.0011 - precision: 0.1573 - recall: 0.0018 - val_AUC: 0.5128 - val
Epoch 4/50
139/139 ----- 41s 237ms/step - AUC: 0.5702 - accuracy: 0.1059 - f1_score: 0.1016 - loss: 6.3065 - precision: 0.1693 - recall: 0.0024 - val_AUC: 0.5491 - val
Epoch 5/50
139/139 ----- 41s 237ms/step - AUC: 0.5828 - accuracy: 0.1093 - f1_score: 0.1009 - loss: 5.1153 - precision: 0.2326 - recall: 0.0067 - val_AUC: 0.5684 - val
Epoch 6/50
139/139 ----- 33s 234ms/step - AUC: 0.6099 - accuracy: 0.1198 - f1_score: 0.1073 - loss: 4.3025 - precision: 0.2739 - recall: 0.0106 - val_AUC: 0.5982 - val
Epoch 7/50
139/139 ----- 33s 237ms/step - AUC: 0.6307 - accuracy: 0.1321 - f1_score: 0.1216 - loss: 3.6878 - precision: 0.3174 - recall: 0.0158 - val_AUC: 0.6099 - val
Epoch 8/50
139/139 ----- 41s 237ms/step - AUC: 0.6435 - accuracy: 0.1375 - f1_score: 0.1263 - loss: 3.3773 - precision: 0.3844 - recall: 0.0213 - val_AUC: 0.6352 - val
Epoch 9/50
139/139 ----- 41s 239ms/step - AUC: 0.6504 - accuracy: 0.1353 - f1_score: 0.1264 - loss: 3.1699 - precision: 0.3480 - recall: 0.0198 - val_AUC: 0.6356 - val
Epoch 10/50
139/139 ----- 33s 234ms/step - AUC: 0.6599 - accuracy: 0.1529 - f1_score: 0.1423 - loss: 2.9869 - precision: 0.4086 - recall: 0.0231 - val_AUC: 0.6397 - val
Epoch 11/50
139/139 ----- 42s 243ms/step - AUC: 0.6728 - accuracy: 0.1549 - f1_score: 0.1421 - loss: 2.8661 - precision: 0.4517 - recall: 0.0266 - val_AUC: 0.6492 - val
Epoch 12/50
139/139 ----- 32s 234ms/step - AUC: 0.6788 - accuracy: 0.1680 - f1_score: 0.1582 - loss: 2.8741 - precision: 0.4319 - recall: 0.0273 - val_AUC: 0.6446 - val
Epoch 13/50
139/139 ----- 33s 240ms/step - AUC: 0.6816 - accuracy: 0.1591 - f1_score: 0.1496 - loss: 2.8282 - precision: 0.4415 - recall: 0.0311 - val_AUC: 0.6518 - val
Epoch 14/50
139/139 ----- 32s 232ms/step - AUC: 0.6775 - accuracy: 0.1524 - f1_score: 0.1403 - loss: 2.8361 - precision: 0.4091 - recall: 0.0295 - val_AUC: 0.6597 - val
Epoch 15/50
139/139 ----- 41s 232ms/step - AUC: 0.6858 - accuracy: 0.1605 - f1_score: 0.1513 - loss: 2.7785 - precision: 0.4469 - recall: 0.0339 - val_AUC: 0.6452 - val
```

Рис.3.14 – Проміжні метрики епох навчання моделі штучної нейронної мережі

Для візуалізації кривих результати навчання зберігаються у змінній history, всередині якої знаходяться значення метрик протягом епох навчання. Відповідно, їх можна візуалізувати через використання matplotlib та отримуючи доступ до значень за іменами метрик, що налаштовано при створенні моделі.

Код виконання візуалізації навчальних кривих по метриках:

```
# Plot loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.show()

# Plot accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
```

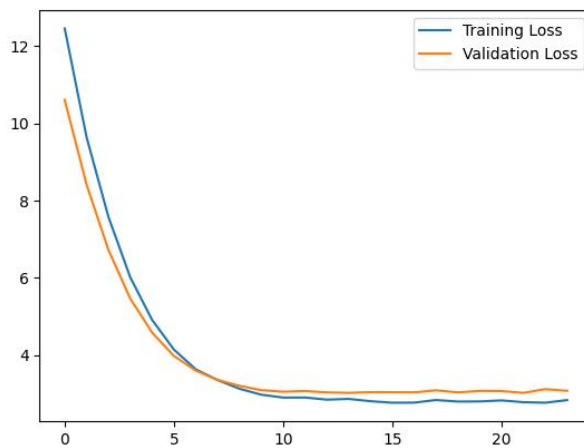
```
plt.show()

# Plot F1
plt.plot(history.history['f1_score'][-1], label='Training F1-score')
plt.plot(history.history['val_f1_score'][-1], label='Validation F1-score')
plt.legend()
plt.show()

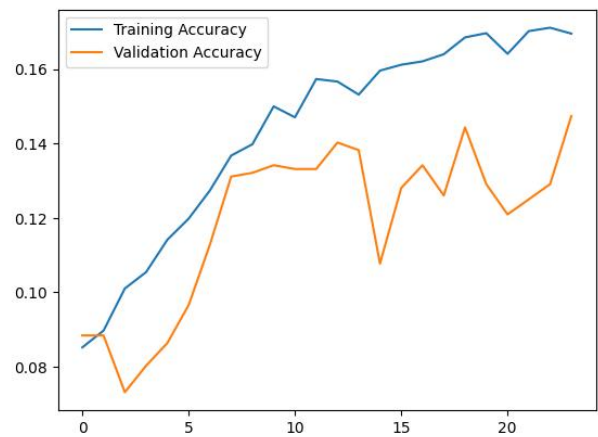
# Plot precision
plt.plot(history.history['precision'], label='Training Precision')
plt.plot(history.history['val_precision'], label='Validation Precision')
plt.legend()
plt.show()

# Plot recall
plt.plot(history.history['recall'], label='Training Recall')
plt.plot(history.history['val_recall'], label='Validation Recall')
plt.legend()
plt.show()

# Plot AUC
plt.plot(history.history['AUC'], label='Training AUC')
plt.plot(history.history['val_AUC'], label='Validation AUC')
plt.legend()
plt.show()
```



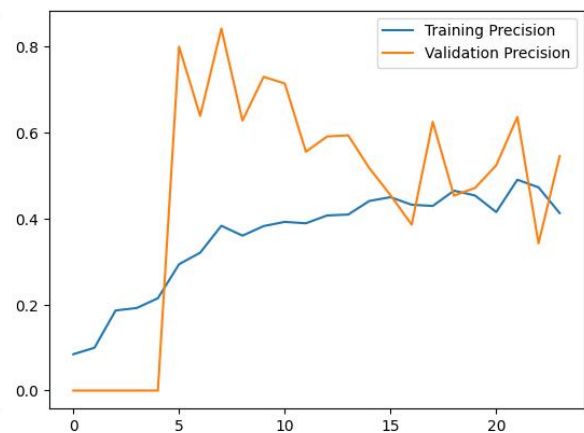
А) Графік втрат



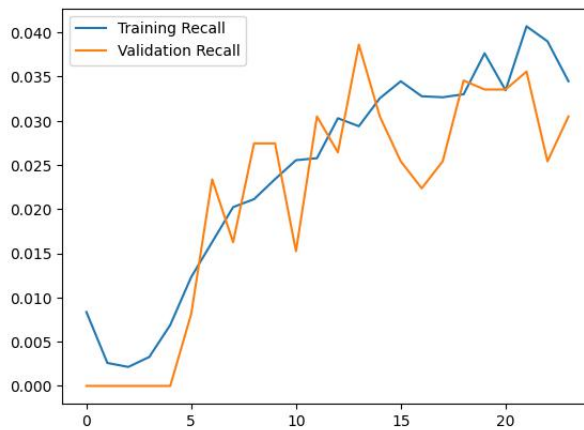
Б) Графік метрики Ассурасу



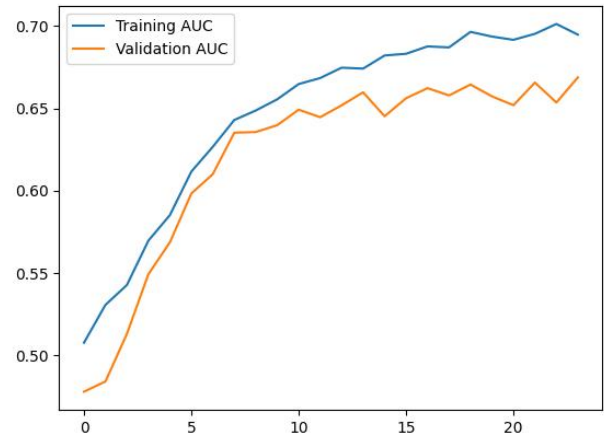
В) Графік F1-Score



Г) Графік Precision



Д) Графік Recall



Е) Графік AUC

Рис.3.15 – Криві навчання моделі штучної нейронної мережі

Візуалізація кривих навчання по метрикам дозволяє зробити висновок, що модель не перенавчилась, але при цьому значення самих метрик є незадовільними.

	precision	recall	f1-score	support
0	0.32	0.33	0.33	33
1	0.14	0.37	0.21	30
2	0.00	0.00	0.00	35
3	0.00	0.00	0.00	58
4	0.19	0.26	0.22	58
5	0.00	0.00	0.00	67
6	0.33	0.08	0.13	76
7	0.13	0.17	0.15	41
8	0.20	0.23	0.21	35
9	0.22	0.75	0.34	16
10	0.17	1.00	0.30	4
11	0.30	0.82	0.44	38
accuracy			0.21	491
macro avg	0.17	0.33	0.19	491
weighted avg	0.16	0.21	0.16	491

Рис.3.16 – Виведення метрик роботи моделі штучної нейронної мережі

Згідно виведених метрик, підтверджується попередньо візуалізовані низькі результати навчання, цього разу підтверджені на тестових даних. Значення метрик:

- accuracy: 0.21;
- precision: середнє = 0.17, зважене = 0.16;
- recall: середнє = 0.33, зважене = 0.21;
- F1-score: середнє = 0.19, зважене = 0.16.

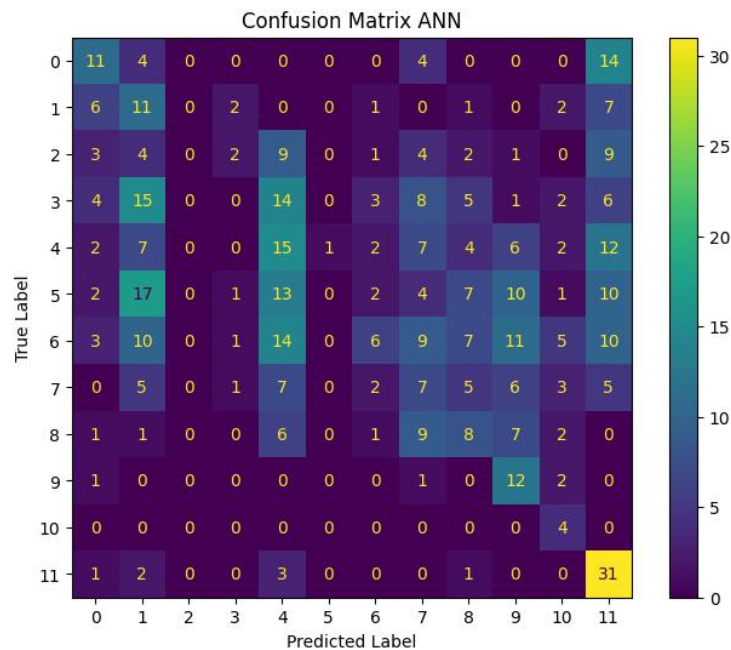


Рис.3.17 – Матриця невідповідностей для моделі штучної нейронної мережі

Згідно візуалізованої матриці невідповідностей, можна впевнено визнати провал даної архітектури нейронної мережі. Якість моделі є низькою, через що необхідним є її оптимізація, аналіз слабких місць та їх виправлення або створення нової моделі, переосмислення архітектури в цілому.

Таблиця 3.1

Порівняльна таблиця метрик моделей

Модель	Accuracy	Precision	Recall	F1-score
SVM	0.66	0.72	0.66	0.66
XGB	0.86	0.87	0.86	0.86
ANN	0.21	0.16	0.21	0.16

Згідно порівняльної таблиці, найефективнішою є модель ансамблевого типу XGB, найслабшою є модель штучної нейронної мережі із поточною архітектурою.

Попри високий потенціал у вирішенні задач подібного типу, поточна нейронна мережа є далекою від оптимальної та потребує глибокої доробки. Можливим шляхом розвитку є використання технологій рекурентних нейронних мереж, таких як LSTM або варіації та GRU [44], створення

ансамблевих моделей штучних нейронних мереж, що використовують декілька моделей різних архітектур в послідовності або паралельно тощо.

У той самий час, ансамблева модель XGBoost показала гарні результати попри відсутність тонкого налаштування моделі при встановлених стандартних параметрах, що може доказує ефективність використання методу ансамблів для вирішення складних задач. При подальшому виконанні тонкого налаштування та загального покращення системи, дана модель має реалістичний потенціал мати високу ефективність роботи з аналізу есе по системі IELTS.

Повний код програми буде зазначено у додатку Б.

Висновки до третього розділу

Протягом виконання третього розділу було розроблено програмний код методів обробки інформації датасету есе системи IELTS, розроблено та навчено, проаналізовано метрики роботи таких методів машинного навчання, як SVM, XGBoost та штучна нейронна мережа власної архітектури.

При розробці функціоналу попередньої обробки текстів та їх векторизації, було використано підхід об'єднання тем есе зі самими роботами для збереження контексту та виконано детальну обробку даних через їх очистку, видалення стоп-слів, токенізацію та лематизацію, та подальшу векторизацію текстів засобами моделі Doc2Vec.

Оброблені дані використовуються у навчанні розроблених моделей контрольованого машинного навчання для класифікації у вигляді моделей SVC, XGBClassifier та послідовної нейронної мережі зі повнозв'язними шарами нейронів із використанням нормалізації даних, регуляризації L2, відсікання нейронів та інших способів контролю над перенавчанням.

Метрики класифікації, отримані при оцінці якості роботи моделей на незалежних тестових даних, було використано для порівняння моделей між собою. Виконане порівняння дозволило виконати оцінку доцільності використання обраних моделей та зокрема якість поточної архітектури нейронної мережі.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи магістра було виконано дослідження предметної області, спроектовано та реалізовано програмне забезпечення дослідження із навчанням моделей машинного навчання для порівняння, для чого було вирішено наступні завдання:

1. Виконано аналіз та дослідження предметної області, а саме проведено пошук та аналіз джерел інформації серед вітчизняних та зарубіжних ресурсів, проаналізовано та досліджено алгоритми обробки текстових даних, проаналізовано доступний інструментарій та описано бібліотеки, модулі до використання в розробці.

2. Виконано аналіз окремих алгоритмів серед використаних технологій та проектування, програмного забезпечення. За результатами моделювання та проектування було отримано майбутнє бачення структури програмного забезпечення та його коду.

3. Виконано програмну реалізацію системи згідно виконаного проектування та досліджених, проаналізованих технологій. Було описано процес розробки та окремі використані компоненти в процесі розробки моделі штучної нейронної мережі.

4. Виконано оцінку метрик ефективності роботи моделей на незалежних даних та порівняння метрик обраних моделей між собою.

Реалізований програмний продукт відповідає поставленим до нього вимогам та реалізує весь функціонал. За час роботи над проектом було виконано поставлені задачі та створено програмне забезпечення за обраною тематикою.

Протягом дослідження джерел інформації, аналізу технологій та розробки програмного забезпечення було покращено наявні та отримано нові знання, вміння та навички з пошуку достовірних та об'єктивних джерел щодо сучасних технологій, розробки систем обробки даних та навчання моделей машинного навчання, зокрема отримано досвід роботи з класичними,

ансамблевими та моделями штучних нейронних мереж з аналізу обробленої текстової інформації по задачі класифікації парадигми контрольованого навчання. Окрім цього, було отримано розуміння сильних та слабких сторін окремих моделей, визначено найгіршу та найкращу за метриками модель у даному дослідженні.

Виконане дослідження показує високий потенціал у використанні методів машинного навчання в роботі з текстовою інформацією, зокрема з її обробки та виконання класифікації. Дана робота показала, що використання відповідних до задач технологій, тонке налаштування та експериментальні дослідження різних моделей, архітектур штучних нейронних мереж дозволяють створювати ефективні моделі та потенційно модернізувати галузі, раніше недоступні для обчислювальних машин, спрощуючи робочі процеси або автоматизовуючи їх.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Олексійчук М. В., Чижмотря О. В. Етичні проблеми розвитку і використання технологій машинного навчання та NLP та підходи до їх вирішення. Комп'ютерні технології: інновації, проблеми, рішення: тези VI Всеукраїнської науково-технічної конференції, м. Житомир, 29-30 листопада 2023 р. – Житомир: Державний університет "Житомирська політехніка", 2023. – С. 176-177.
2. Oleksiichuk M.V., Chyzhmotria O.V. Ensemble methods of Machine Learning: Opportunities and prospects. Комп'ютерні технології: інновації, проблеми, рішення: тези VII Всеукраїнської науково-технічної конференції, м. Житомир, 02-03 грудня 2024 р. – Житомир: Державний університет «Житомирська політехніка», 2024
3. Oleksiichuk M.V., Chyzhmotria O.V. Sentiment analysis of texts using NLP and Machine Learning. Комп'ютерні технології: інновації, проблеми, рішення: тези VII Всеукраїнської науково-технічної конференції, м. Житомир, 02-03 грудня 2024 р. – Житомир: Державний університет «Житомирська політехніка», 2024
4. Brown S. Machine learning, explained [Електронний ресурс] / Sara Brown // MIT Sloan – 2021р. – Режим доступу до ресурсу: <https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained>
5. What is machine learning? [Електронний ресурс] // IBM – 2023р. – Режим доступу до ресурсу: <https://www.ibm.com/topics/machine-learning>
6. Kaseb K. Understand Machine Learning [Електронний ресурс] / Kayvan Kaseb // Medium – 2020р. – Режим доступу до ресурсу: <https://medium.com/kayvan-kaseb/understand-machine-learning-e140fc93bdf1>
7. What Is Natural Language Processing (NLP)? [Електронний ресурс] // Oracle – 2021р. – Режим доступу до ресурсу: <https://www.oracle.com/ua/artificial-intelligence/what-is-natural-language-processing/>

8. Firican G. The history of Machine Learning [Электронный ресурс] / George Firican // Lights On Data – Режим доступа до ресурсу: <https://www.lightsondata.com/the-history-of-machine-learning/>
9. Foote K. A Brief History of Machine Learning [Электронный ресурс] / Keith D. Foote // Dataversity – 2021p. – Режим доступа до ресурсу: <https://www.dataversity.net/a-brief-history-of-machine-learning/>
10. Foote K. A Brief History of Natural Language Processing – [Электронный ресурс] / Keith D. Foote // Dataversity – 2023p. – Режим доступа до ресурсу: <https://www.dataversity.net/a-brief-history-of-natural-language-processing-nlp/>
11. Louis A. A Brief History of Natural Language Processing — Part 1 – [Электронный ресурс] / Antoine Louis // Medium – 2020p. – Режим доступа до ресурсу: <https://medium.com/@antoine.louis/a-brief-history-of-natural-language-processing-part-1-ffbc937ebce>
12. Etienne B. Introduction to Machine Learning / Bernard Etienne., 2021. – 424 с. – (Wolfram Media, Inc.)
13. Ali M. Supervised Machine Learning – [Электронный ресурс] / Moez Ali // DataCamp – 2022p. – Режим доступа до ресурсу: <https://www.datacamp.com/blog/supervised-machine-learning>
14. 10 popular regression algorithms in Machine Learning [Электронный ресурс] // Online Manipal – 2023p. – Режим доступа до ресурсу: <https://www.onlinemanipal.com/blogs/popular-regression-algorithms-in-machine-learning>
15. Choosing the Best Machine Learning Classification Model and Avoiding Overfitting. Chapter 1: Choosing the Right Classification Model – [Электронный ресурс] // MathWorks – Режим доступа до ресурсу: <https://www.mathworks.com/campaigns/offers/next/choosing-the-best-machine-learning-classification-model-and-avoiding-overfitting.html>

16. Pykes K. Introduction to Unsupervised Learning – [Электронный ресурс] / Kurtis Pykes // DataCamp – 2024г. – Режим доступа до ресурсу: <https://www.datacamp.com/blog/introduction-to-unsupervised-learning>
17. Clustering in Machine Learning [Электронный ресурс] // JavaTpoint – Режим доступа до ресурсу: <https://www.javatpoint.com/clustering-in-machine-learning>
18. Priy S. Clustering in Machine Learning [Электронный ресурс] / Surya Priy // GeeksforGeeks – 2023г. – Режим доступа до ресурсу: <https://www.geeksforgeeks.org/clustering-in-machine-learning/>
19. Bhatt S. Reinforcement Learning 101 [Электронный ресурс] / Shweta Bhatt // Medium – 2018г. – Режим доступа до ресурсу: <https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292>
20. Bajaj P. Reinforcement learning – [Электронный ресурс] / Prateek Bajaj // GeeksforGeeks – 2023г. – Режим доступа до ресурсу: <https://www.geeksforgeeks.org/what-is-reinforcement-learning/>
21. Shacklett M. Structured vs Unstructured Data: Key Differences [Электронный ресурс] / Mary Shacklett // Datamation. – 2023. – Режим доступа до ресурсу: <https://www.datamation.com/big-data/structured-vs-unstructured-data/>
22. jacobperalta. Text Preprocessing in Python [Электронный ресурс] / jacobperalta // GeeksforGeeks. – 2024. – Режим доступа до ресурсу: <https://www.geeksforgeeks.org/text-preprocessing-in-python-set-1/>
23. jacobperalta. Text Preprocessing in Python | Set 2 [Электронный ресурс] / jacobperalta // GeeksforGeeks. – 2024. – Режим доступа до ресурсу: <https://www.geeksforgeeks.org/text-preprocessing-in-python-set-2/>
24. Chai C. Comparison of text preprocessing methods / Christine P. Chai // Natural Language Engineering, vol. 29, no. 3 – Кембридж: Cambridge University Press, 2023. – С. 509–553
25. Removing stop words with NLTK in Python [Электронный ресурс] // GeeksforGeeks. – 2024. – Режим доступа до ресурсу: <https://www.geeksforgeeks.org/removing-stop-words-nltk-python/>

26. Murel J. What are stemming and lemmatization? [Электронный ресурс] / J. Murel Ph.D., E. Kavlakoglu // IBM. – 2023. – Режим доступа до ресурсу: <https://www.ibm.com/topics/stemming-lemmatization>
27. Sineglazov, V. M.. Comparative Analysis of Text Vectorization Methods / Sineglazov, V. M., Savenko, I. M. // Electronics and Control Systems №2(76) – Kyiv: National Aviation University, 2023. – С. 21–27
28. Khadka P. Text Vectorization [Электронный ресурс] / Pranjal Khadka // Medium. – 2024. – Режим доступа до ресурсу: <https://medium.com/@pranjalkhadka/text-vectorization-de690f5edb5e>
29. ram9119. Doc2Vec in NLP [Электронный ресурс] / ram9119 // GeeksforGeeks. – 2023. – Режим доступа до ресурсу: <https://www.geeksforgeeks.org/doc2vec-in-nlp/>
30. The world's most trusted English test [Электронный ресурс] // IELTS – 2024. – Режим доступа до ресурсу: <https://ielts.org/>
31. Suthaharan S. Support Vector Machine / Shan Suthaharan // Machine Learning Models and Algorithms for Big Data Classification / Shan Suthaharan. – Бостон, Массачусетс, США: Springer, Boston, MA, 2016. – С. 207–235
32. XgBoost [Электронный ресурс] // IT Wiki. – 2023. – Режим доступа до ресурсу: <https://itwiki.dev/data-science/ml-reference/ml-glossary/xgboost>
33. Nagpal A. L1 and L2 Regularization Methods [Электронный ресурс] / Anuja Nagpal // Towards Data Science. – 2017. – Режим доступа до ресурсу: <https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c>
34. Saxena S. Introduction to Batch Normalization [Электронный ресурс] / Shipra Saxena // Analytics Vidhya. – 2024. – Режим доступа до ресурсу: <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-batch-normalization/>
35. Krishnamurthy B. An Introduction to the ReLU Activation Function [Электронный ресурс] / B. Krishnamurthy, B. Whitfield // Built In. – 2024. – Режим доступа до ресурсу: <https://builtin.com/machine-learning/relu-activation-function>

36. Buhl N. Activation Functions in Neural Networks: With 15 examples [Електронний ресурс] / Nikolaj Buhl // Encord. – 2023. – Режим доступу до ресурсу: <https://encord.com/blog/activation-functions-neural-networks/>
37. Accuracy, precision, and recall in multi-class classification [Електронний ресурс] // Evidently AI. – 2024. – Режим доступу до ресурсу: <https://www.evidentlyai.com/classification-metrics/multi-class-metrics>
38. Udeh O. 2 apply() Method Alternatives You Should Try [Електронний ресурс] / Onyedikachi Udeh // Medium. – 2021. – Режим доступу до ресурсу: <https://medium.com/analytics-vidhya/2-apply-method-alternatives-you-should-try-909f3f06cec6>
39. TQDM [Електронний ресурс] // GitHub. – 2024. – Режим доступу до ресурсу: <https://github.com/tqdm/tqdm>
40. nltk.stem.WordNetLemmatizer [Електронний ресурс] // NLTK. – 2 січня 2023. – Режим доступу до ресурсу: <https://www.nltk.org/api/nltk.stem.WordNetLemmatizer.html>
41. Princeton University "About WordNet." [Електронний ресурс] // Princeton University. – 2010-2024. – Режим доступу до ресурсу: <https://wordnet.princeton.edu/>
42. Řehůřek R. Doc2Vec Model [Електронний ресурс] / Radim Řehůřek // gensim. – 2024. – Режим доступу до ресурсу: https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html
43. Brownlee J. How to Identify Overfitting Machine Learning Models in Scikit-Learn [Електронний ресурс] / Jason Brownlee // Machine Learning Mastery. – 27 листопада 2020. – Режим доступу до ресурсу: <https://machinelearningmastery.com/overfitting-machine-learning-models/>
44. Hybrid RNN Based Text Classification Model for Unstructured Data / [P. Sunagar, B. Sowmya, D. Pruthviraja та ін.] // SN Computer Science, Volume 5, Article number 726, 2024

ДОДАТКИ

Приклад векторизації тексту

Document 1 : interview form the basic criterion for most large company however some people think that the interview is not a reliable method of choosing whom to employ and there are other better method to what extent do you agree or disagree it is believed by some expert that the traditional approach of recruiting candidate which is interviewing is the best way whereas others think different method such a exam writing cv cover letter or application letter and many more are good i strongly agree with the statement interview is the most reliable approach to recruit worker because this method assist the recruiter to know the person and his ability to do the work and their problem solving ability to begin with an interview enables the recruiter to know the kind of person he or she is recruiting it help the employer to see the personality trait of the employee such a how he answer question his facial mannerism and also his communication skill that is whether introvert or extrovert also his teamwork skill is measured during the dialogue for instance job like sale personnel require good communication skill to be able to do the work effectively and efficiently so interview allow the manager to assess whether or not the applicant qualifies for the job furthermore recruiter also assess the applicant s ability to solve problem when they arise a good idea generated or how one handle situation can bring great development to the company for instance pressure can put fear into an employee which can make him make a wrong decision that can bring loss to the company while some too can take pressure in a calm action and make a good decision on the other hand other method such a cv cover letter the use of only certificate and many more are not a suitable step to recruit an applicant due to the fact that it doe not allow the recruiter to see the full potential of the candidate information found in the cv or cover letter may not be true because people lie to obtain what they desire in the same way a candidate can also lie to acquire the position to sum up i think an interview is still the most reliable practice of hiring employee rather than using other method so i suggest manager use only interview a a mean of sourcing worker for their company

Vector: [-1.0916919 1.5454077 -0.43409622 0.11394569 0.13053334
0.17674327

1.6218566	-1.3534964	-1.1093826	1.3297606	0.00671802	-1.2463135
-0.8747974	-0.8569886	0.02930998	-0.0270223	-0.23803891	0.2995371
-0.9370086	2.2058282	-1.1288146	0.02513292	-0.16134004	0.03681787
0.725869	-0.7423857	-0.05147475	-0.11150272	-0.43917218	-0.3007106
-0.22362639	-0.810368	1.5440625	1.4001259	-0.93216974	0.06567676
-0.8807401	-2.0749984	1.4076253	-1.1409135	0.41825676	1.4714515
-0.93425417	0.55394787	0.41418496	0.31503573	-1.6611172	0.33140084
-0.41054094	-0.29164398	-0.10988791	-1.3854544	0.26044363	-0.4024756
1.5488842	-0.5259267	-1.9673034	-1.2364414	-0.23497939	-0.8212332
2.4573128	-1.4698858	0.87030137	0.7414933	2.1919224	-0.4232342
0.64305115	-2.4897494	-0.49935144	1.624278	0.7702142	-1.6311599
0.17106514	-0.8982825	-1.617578	-0.9343929	0.26571184	-2.7948983
0.11183555	-1.8251857	0.2592528	-1.1181048	2.0293283	1.6460096
0.94843096	0.01287154	-0.53875595	-1.4909232	-0.14778349	0.9560051
1.3281661	0.50797975	0.04057613	-0.4446241	-1.2294954	-0.28213334
0.59641314	0.26681697	0.17061512	-0.43735337	0.41708615	-1.0885088
-1.7134961	-0.6965523	0.5866205	-0.02444898	-1.3023264	1.0919653
-0.6099261	-0.39529282	-1.3716125	2.2901473	0.10209296	-0.75803775
-0.05704032	-2.6857238	-0.7159961	-0.61297417	0.8398568	2.0903623
-0.14315905	-0.90926665	0.74984354	0.03688549	1.8745269	0.3079822
-0.20480107	0.40313393	-0.01531118	-0.05524451	-0.04587816	0.9502136
-1.7491692	0.5523515	-0.99586236	-0.17615129	1.0571772	1.3987124
-0.44675565	0.5666262	0.7541873	2.53608	-1.6566359	-1.4087701
-0.45179972	0.10632052	-1.4916121	0.7457822	0.30792183	0.18898396
2.1756127	-0.15707242	-0.5296332	-0.56897986	-0.631238	0.29263657
-0.7680968	1.9236397	-1.5069399	-0.21786128	0.29740122	0.6990409
-0.09056687	-0.8929159	-0.46151888	-1.0047789	-0.714762	-1.2737081

-1.8965735	1.4526329	1.2504337	0.01690427	0.1791292	-0.67365855
0.114768	0.9567824	-1.6151675	1.2399149	-0.6288655	1.3401972
-0.6417033	0.7296063	-1.4742122	-1.249713	1.0371662	1.0880128
-0.9319442	0.95366	-0.28322768	-1.2806529	0.33023834	-0.73044246
1.6140598	0.9171999	0.5466823	0.62600815	0.07076201	0.68651885
0.23780178	-0.10490654	-1.8388289	-0.7222166	-0.54522073	-1.68605
-1.2894144	-1.7072396	0.0144592	1.0380931	0.13467519	-1.6502755
-3.7625833	0.16101672	2.7871706	0.7835233	0.08478764	0.29416123
-0.939974	-2.0388758	0.25583944	-0.02068846	2.4077823	1.0462354
2.410426	-0.02007837	0.63924766	-0.5834951	0.2504959	1.0939162
-1.7834046	-0.7237686	1.4476275	0.12213387	0.32779577	0.18277958
0.3828326	-0.7953672	-0.8657656	0.04094954	-0.838005	-1.4097329
0.1749706	2.395019	-2.0722706	-0.963077	0.41159236	0.13980769
1.7124792	-0.25440192	-0.2610861	-1.4591508	-0.52067405	-1.1417356
-4.4177747	0.22449388	2.0678134	0.08301826	-1.3191979	0.13756664
-0.1974871	1.0078622	0.71931046	-1.3486915	-1.1395057	0.03500338
-0.39538628	0.5259185	0.68441945	0.59383494	-0.37748578	-0.24322416
0.47583258	0.48786756	1.2491776	0.44883457	1.2963759	-1.2514031
0.57557327	-0.96252775	-0.6803547	-0.40459317	-0.6246476	0.90415484
1.4255404	-2.0639277	1.1354085	-1.8657646	-1.9540448	-1.3081841
-0.9723516	1.5604023	-1.0753322	0.44452122	-0.6763772	1.1452585
-0.4477943	1.3050729	1.1819193	-0.74923277	-0.6798502	-1.3600149]

Повний код програмного забезпечення з файлу .ipynb

```

from google.colab import drive
import os.path
import numpy as np
import pandas as pd
from tqdm import tqdm
import matplotlib.pyplot as plt

import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

import gensim
from gensim.parsing.preprocessing import strip_tags,
strip_multiple_whitespaces, \
strip_punctuation, remove_stopwords
from gensim.models.doc2vec import Doc2Vec, TaggedDocument

from sklearn.metrics import accuracy_score, classification_report, \
confusion_matrix, ConfusionMatrixDisplay, roc_curve, auc
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.utils.class_weight import compute_class_weight
from sklearn.svm import SVC

from xgboost import XGBClassifier

import tensorflow as tf

import string
import re
import pickle

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
nltk.download('wordnet')
nltk.download('tagsets')
nltk.download('universal_tagset')
nltk.download('punkt_tab')

tqdm.pandas()

"""# Підготовка датасету

## Завантаження даних
"""

drive.mount('/content/drive/')

path = "drive/MyDrive/Colab Notebooks/Кваліфікаційна робота/"
!ls "{path}"

dataset = pd.read_csv(path + "train.csv")
dataset.head(5)

"""## Обробка датасету"""

dataset.info()
```

```

dataset.describe()

dataset.iloc[0].band

dataset = dataset.drop_duplicates()

dataset = dataset.replace(r'(\r*\n*)*', '', regex=True)
dataset['band'] = dataset['band'].replace(r'.0', '', regex=True)

dataset = dataset.dropna(axis=1)
dataset.head(5)

dataset.info()

dataset.describe()

len(dataset['prompt'].unique())

dataset['band'].unique()

dataset['band'].value_counts()

"""## Обработка текста"""

texts = dataset['prompt'] + dataset['essay']
texts.head(5)

lemmatizer = WordNetLemmatizer()
def preprocess(text: str, tokenize=True, no_stopwords=True, lemmatize=True) -
> str:
    text = text.strip()
    text = strip_tags(text)
    text = strip_multiple_whitespace(text)
    text = re.sub(r'\.\s*([A-Z])', r'. \1', text)

    if (not tokenize):
        return text # Returns cleaned text

    text = text.lower()
    text = strip_punctuation(text)

    if (no_stopwords):
        tokens = remove_stopwords(text)

    tokens = word_tokenize(text)

    if (lemmatize):
        tokens = [lemmatizer.lemmatize(word) for word in tokens]

    return ' '.join(tokens) # Returns preprocessed by conditions sequence of
words from text
    # return tokens

preprocess(texts[0])

preprocessed_texts = texts.progress_apply(preprocess)

dataset.insert(2, "preprocessed_text", preprocessed_texts)

dataset.head(5)

data = np.array(preprocessed_texts)

data[0]

```

```

"""## Векторизація"""

for i, row in enumerate(data):
    data[i] = word_tokenize(row)

tagged_texts = [TaggedDocument(words=text, tags=[i]) for i, text in
enumerate(data)]

vector_size = 300
if os.path.isfile(path + f'vectorizer_{vector_size}.pkl'):
    with open(path + f'vectorizer_{vector_size}.pkl', 'rb') as file:
        model_doc_to_vec = pickle.load(file)

    print(f"Vectorizer was loaded from file 'vectorizer_{vector_size}.pkl'")
else:
    print("Vectorizer created anew")

    model_doc_to_vec = Doc2Vec(vector_size=vector_size,
                                min_count=2, epochs=50)
    model_doc_to_vec.build_vocab(tagged_texts)
    model_doc_to_vec.train(tagged_texts,
                            total_examples=model_doc_to_vec.corpus_count,
                            epochs=model_doc_to_vec.epochs)

    with open(path + f'vectorizer_{vector_size}.pkl', 'wb') as file:
        pickle.dump(model_doc_to_vec, file)
    print(f"Vectorizer was saved into file 'vectorizer_{vector_size}.pkl'")

document_vectors = np.array([model_doc_to_vec.infer_vector(
    word_tokenize(document)) for document in dataset.preprocessed_text])

for i, record in dataset.head(5).iterrows():
    print("Document", i + 1, ":", record.preprocessed_text)
    print("Vector:", document_vectors[i])
    print()

document_vectors[0]

X = document_vectors.copy()
y = np.array(dataset.band)

X

y

le = LabelEncoder()
y = le.fit_transform(y)
y

"""## Підготовка тестових даних"""

test = pd.read_csv(path + "test.csv")
test = test.replace(r'(\r*\n*)*', '', regex=True)
test['band'] = test['band'].replace(r'.0', '', regex=True)
test = test.dropna(axis=1)
test.head(5)

test_texts = test['prompt'] + test['essay']
test_texts_prep = test_texts.progress_apply(preprocess)
data_test = np.array(test_texts_prep)

test.insert(2, "preprocessed_text", test_texts_prep)

```

```

test_vectors = np.array([model_doc_to_vec.infer_vector(
    word_tokenize(document)) for document in test.preprocessed_text])

X_test = test_vectors.copy()
y_test = le.transform(np.array(test.band))

"""## Розділення тренувальних даних на тренувальну та валідаційну вибірки"""

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.1)
np.unique(y_train, return_counts=True), np.unique(y_val, return_counts=True)
print(len(X_train), len(X_val), len(X_test))

"""# Робота моделі SVM для класифікації"""

svm_path = path + f'classifier_svm_{vector_size}.pkl'

if os.path.isfile(svm_path):
    print(f"SVC model was loaded from file '{svm_path}'")
    with open(svm_path, 'rb') as file:
        svm_classifier = pickle.load(file)
else:
    print("SVC model created anew")
    svm_classifier = SVC(kernel='rbf')
    svm_classifier.fit(X_train, y_train)

    with open(svm_path, 'wb') as file:
        pickle.dump(svm_classifier, file)
    print(f"SVC model was saved into file '{svm_path}'")

y_pred = svm_classifier.predict(X_test)

df_class = pd.DataFrame({"y_test": le.inverse_transform(y_test),
                        "y_pred": le.inverse_transform(y_pred)})
print(df_class)

print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)

fig, ax = plt.subplots(figsize=(8, 6))
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(ax=ax)

plt.title('Confusion Matrix SVC')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')

"""# Робота моделі XGBoost"""

xgboost_path = path + f'classifier_xgb_{vector_size}.pkl'

np.unique(y_train), np.unique(y_val)

if os.path.isfile(xgboost_path):
    print(f"XGB model was loaded from file '{xgboost_path}'")
    with open(xgboost_path, 'rb') as file:
        xgb_classifier = pickle.load(file)
else:
    print("XGB model created anew")

    xgb_classifier = XGBClassifier(n_estimators=75, objective="multi:softprob",
    learning_rate=0.2)

```

```

xgb_classifier.fit(X_train, y_train, eval_set=(X_val, y_val))

with open(xgboost_path, 'wb') as file:
    pickle.dump(xgb_classifier, file)
print(f"XGB model was saved into file '{xgboost_path}'")

y_pred = xgb_classifier.predict(X_test)

df_class = pd.DataFrame({"y_test": le.inverse_transform(y_test),
                        "y_pred": le.inverse_transform(y_pred)})
print(df_class)

print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)

fig, ax = plt.subplots(figsize=(8, 6))
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(ax=ax)

plt.title('Confusion Matrix XGB')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')

"""# Робота моделі штучної нейронної мережі"""

early_stopping_callback = tf.keras.callbacks.EarlyStopping(patience = 10,
restore_best_weights = True)

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=0.001,
    decay_steps=9000,
    decay_rate=0.96
)

model_checkpoint_callback =
tf.keras.callbacks.ModelCheckpoint('ann_best_model.keras',
save_best_only=True)

ann_path = path + f'classifier_ann_{vector_size}.keras'

if os.path.isfile(ann_path):
    ann_model = tf.keras.models.load_model(ann_path,
custom_objects=tf.keras.layers.ELU)
    print(f"ANN model was loaded from file '{ann_path}'")
else:
    print("ANN model created anew")
    ann_model = tf.keras.models.Sequential([
        tf.keras.layers.Dense(256, activation=tf.keras.layers.ELU(alpha=0.01),
                                input_dim=X_train.shape[4],
                                kernel_regularizer=tf.keras.regularizers.l2(0.016)
                                ),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(256, activation=tf.keras.layers.ELU(alpha=0.01),
                                kernel_regularizer=tf.keras.regularizers.l2(0.016)
                                ),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Dropout(0.4),
        tf.keras.layers.Dense(64, activation=tf.keras.layers.ELU(alpha=0.01),
                                kernel_regularizer=tf.keras.regularizers.l2(0.016)
                                ),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Dropout(0.4),

```

```

tf.keras.layers.Dense(32, activation=tf.keras.layers.ELU(alpha=0.01),
                      kernel_regularizer=tf.keras.regularizers.l2(0.016)
                      ),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Dropout(0.3),
tf.keras.layers.Dense(len(np.unique(y_train)), activation='softmax')
])

ann_model.compile(optimizer=tf.keras.optimizers.AdamW(learning_rate=lr_schedule, weight_decay=0.01),
                 loss='categorical_crossentropy',
                 metrics=['AUC',
                        tf.keras.metrics.CategoricalAccuracy(name='accuracy'),
                        tf.keras.metrics.Precision(name='precision'),
                        tf.keras.metrics.Recall(name='recall'),
                        tf.keras.metrics.F1Score(name='f1_score')])

tf.config.run_functions_eagerly(True)

class_weights = compute_class_weight("balanced", classes=np.unique(y_train),
y=y_train)

if not os.path.isfile(ann_path):
    history = ann_model.fit(
        X_train, tf.keras.utils.to_categorical(y_train),
        validation_data=(X_val, tf.keras.utils.to_categorical(y_val)),
        class_weight=dict(enumerate(class_weights)),
        epochs=50, batch_size=64, shuffle=True,
        # callbacks=[early_stopping_callback, lr_scheduler_callback,
model_checkpoint_callback])
        # callbacks=[early_stopping_callback, reduce_lr_on_plateau_callback,
model_checkpoint_callback])
        callbacks=[early_stopping_callback, model_checkpoint_callback])
    ann_model.load_weights('ann_best_model.keras')

    # with open(path + 'classifier_ann.pkl', 'wb') as f:
    #     pickle.dump(ann_model, f)
    ann_model.save(ann_path)
    print(f"ANN model was saved into file '{ann_path}'")

    # Plot loss
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.legend()
    plt.show()

    # Plot accuracy
    plt.plot(history.history['accuracy'], label='Training Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.legend()
    plt.show()

    # Plot F1
    plt.plot(history.history['f1_score'][-1], label='Training F1-score')
    plt.plot(history.history['val_f1_score'][-1], label='Validation F1-score')
    plt.legend()
    plt.show()

    # Plot precision
    plt.plot(history.history['precision'], label='Training Precision')
    plt.plot(history.history['val_precision'], label='Validation Precision')
    plt.legend()
    plt.show()

```

```

# Plot recall
plt.plot(history.history['recall'], label='Training Recall')
plt.plot(history.history['val_recall'], label='Validation Recall')
plt.legend()
plt.show()

# Plot AUC
plt.plot(history.history['AUC'], label='Training AUC')
plt.plot(history.history['val_AUC'], label='Validation AUC')
plt.legend()
plt.show()

y_pred = ann_model.predict(X_test)

y_pred_clear = np.argmax(y_pred, axis=1)

ann_model.evaluate(X_val, tf.keras.utils.to_categorical(y_val))

df_class = pd.DataFrame({"y_test": le.inverse_transform(y_test),
                        "y_pred": le.inverse_transform(y_pred_clear)})
print(df_class)

print(classification_report(y_test, y_pred_clear))

cm = confusion_matrix(y_test, y_pred_clear)

fig, ax = plt.subplots(figsize=(8, 6))
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(ax=ax)

plt.title('Confusion Matrix ANN')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')

```