МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2022 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА РОЗПІЗНАВАННЯ СЕМАНТИКИ ЕЛЕКТРОННОГО ЛИСТА

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810308

Виконав студент 4-го курсу, групи 402
_____В. В. Димо_____
«___» _____ 2022 р.

Керівник: *__д-р техн. наук, проф._*
_____О. П. Гожий__
«___» _____ 2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**

Галузь знань **122 «Інформаційні технології»**

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2021_р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 402 факультету комп'ютерних наук Димо Валерію Володимировичу.

1. Тема кваліфікаційної роботи «Інтелектуальна система розпізнавання семантики електронного листа».

Керівник роботи Гожий Олександр Петрович, д-р техн. наук, професор.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «7» грудня 2021 р. № 318

2. Строк представлення кваліфікаційної роботи студентом «27» червня 2022 р.

3. Вхідні (початкові) дані до роботи: тексти електронних листів на англійській мові; навчена модель нейронної мережі.

Очікуваний результат: класифікація електронних листів згідно семантики змісту, розпізнавання вмісту спаму у тексті.

4. Перелік питань, що підлягають розробці:

– аналіз сучасного стану задач обробки природньої мови;

– огляд існуючих методів та алгоритмів штучного інтелекту;

– технологічні та програмні рішення побудови, навчання та тестування алгоритмів машинного навчання;

– аналіз результатів застосування обраних методів машинного навчання для вирішення поставленої задачі;

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Аналіз вимог до організації заходів техніки безпеки та охорони праці під час роботи за комп’ютером»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	канд. техн. наук, доцент Алексєєва Анна Олександрівна	

Керівник роботи _____д-р техн. наук, проф. Гожий О.П._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання _____Димо В. В._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_20_» _____листопада____ 2021 р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: Інтелектуальна система розпізнавання семантики електронного листа

	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	26.10.2021	30.10.2021	Виконано
2	Отримання завдання на виконання БКР	20.11.2021	20.11.2021	Виконано
3	Складання календарного плану	07.12.2021	07.12.2021	Виконано
4	Отримання завдання на переддипломну практику	20.05.2022	20.05.2022	Виконано
5	Проходження переддипломної практики, написання практичної частини БКР	23.05.2022	04.06.2022	Виконано
6	Розробка звіту з переддипломної практики	04.06.2022	06.06.2022	Виконано
7	Початок виконання БКР: збір матеріалу та даних, написання аналітичної частини БКР	28.02.2022 та 06.06.2022	27.03.2022 та 19.06.2022	Виконано
8	Попередній захист БКР на засіданні комісії кафедри	30.05.2022	31.05.2022	Виконано
9	Розробка звіту з переддипломної практики	01.06.2022	03.06.2022	Виконано
10	Доробка та остаточне оформлення БКР	02.06.2022	20.06.2022	Виконано
11	Подання БКР рецензенту	16.06.2022	18.06.2022	Виконано
12	Подання БКР, електронні копії та інші необхідні документи до захисту	20.06.2022	20.06.2022	Виконано
13	Захист БКР перед екзаменаційною комісією	27.06.2022	27.06.2022	Виконано

Розробив студент Димо В. В. _____
(прізвище та ініціали) (*підпис*)

Керівник роботи д-р техн. наук, проф. Гожий О.П. _____
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

« 8 » грудня 2021 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента групи 402

ЧНУ ім. Петра Могили

Димо Валерія Володимировича

Тема: «Інтелектуальна система розпізнавання семантики електронного листа»

Об’єктом дослідження є процес обробки природньої мови за допомогою машинного навчання.

Предметом дослідження є методи та алгоритми класифікації в додатках, які використовують обробку природньої мови.

Мета дипломної роботи – класифікація змісту електронного листа на наявність спаму за допомогою методів машинного навчання.

Робота складається з фахового розділу і спеціальної частини з охорони праці. Пояснювальна записка складається зі вступу, п’яти розділів та висновків.

У першому розділі було розкрито задачу обробки природньої мови, проаналізовано сучасний стан проблеми та методи її рішення.

У другому розділі – наведено детальний опис технік NLP, приклади рішення задачі класифікації за допомогою машинного навчання.

Третій розділ описує можливості бібліотеки для машинного навчання Scikit-Learn, а також використання Gmail API, PyQt5 у створенні інтелектуальної системи.

У четвертому розділі описується реалізації інтелектуальної системи з використанням створеної моделі, та демонструються результати її роботи.

В результаті розроблено desktop–додаток мовою програмування Python з використанням бібліотек Scikit-Learn, Gmail API, PyQt5.

Бакалаврська кваліфікаційна робота містить 94 сторінок, 46 рисунків, 13 таблиць, 12 формул, 38 використаних джерел та 2 додатки.

Ключові слова: штучний інтелект, машинне навчання, Python, Scikit-Learn, PyQt5, NLP.

ABSTRACT

to the bachelor's qualification work by the student of group 402 of

Petro Mohyla Black Sea National University

Dymo Valerii Volodymyrovych

Topic: "Intelligent e-mail semantics recognition system"

The object of research is the process of processing natural language with the help of machine learning.

The subject of research is the methods and algorithms of classification in applications that use natural language processing.

The purpose of the thesis is to classify the content of the e-mail for the presence of spam using machine learning methods.

The research consists of a special part and a part on labor protection. The explanatory note consists of an introduction, five chapters and conclusions.

In the first section the problem of natural language processing was revealed, the current state of the problem and methods of its solution were analyzed.

The second section provides a detailed description of NLP techniques, examples of solving the problem of classification using machine learning.

The third section describes the capabilities of the Scikit-Learn machine learning library, as well as the use of the Gmail API, PyQt5 in developing an intelligent system.

The fourth section describes the implementation of an intelligent system using the created model, and demonstrates the results of its work.

As a result, a desktop application was developed in the Python programming language using the Scikit-Learn, Gmail API, and PyQt5 libraries.

The bachelor's thesis contains 94 pages, 46 figures, 13 tables, 12 formulas, 38 sources used and 2 appendices.

Keywords: artificial intelligence, machine learning, Python, Scikit-Learn, PyQt5, NLP.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ВДТ – візуальний дисплейний термінал

ЕОМ – електронна обчислювальна машина

ПК – персональний комп’ютер

ПЕОМ – персональні електронні обчислювальні машини

API – application programming interface

CNN – convolutional neural network

GRU – gated recurrent network

GUI – graphical user interface

IDE – integrated development environment

IDF – inversive document frequency

LSTM – long short-term memory

ML – machine learning

NLP – natural language processing

PDP – partial dependence plot

RNN – recurrent neural network

SQL – structured query language

TF – term frequency

Пояснювальна записка

до кваліфікаційної роботи

на тему:

«ІНТЕЛЕКТУАЛЬНА СИСТЕМА РОЗПІЗНАВАННЯ СЕМАНТИКИ ЕЛЕКТРОННОГО ЛИСТА»

Спеціальність 122 «Комп’ютерні науки»

122 – БКР – 402.21810308

Виконав студент 4-го курсу, групи 402
В. В. Димо
«___» _____ 2022 р.

Керівник: *д-р техн. наук, проф.*
О. П. Гожий
«___» _____ 2022 р.

Миколаїв – 2022

ВСТУП

Розробка комп’ютера та поява глобальної мережі Інтернет дали неймовірний поштовх у сфері інформаційних технологій, як корисних інструментів для автоматизації процесів у офісах та виробництві на підприємствах, так і для застосування у звичайному житті людиною.

Під час технологічного буму з’явилося багато нових можливостей, таких як передача даних, створення електронних документів, пошти тощо. З новими технологіями прийшли і нові задачі, вирішування яких стало одним із ключових моментів розвитку.

Обробка природної мови або NLP (англ. Natural Language Processing) – спеціальний напрям у математичній лінгвістиці, інформатиці та штучного інтелекту, який вивчає синтез природньої мови за допомогою комп’ютерного аналізу. Таким чином, використовуючи методи NLP комп’ютерна система має змогу «розуміти» зміст тексту та надавати йому сенс [1, 2].

Методи обробки природньої мови набрали великої популярності, оскільки взаємодія людини із комп’ютером завжди була актуальною. Практичне застосування методів NLP можна знайти у різноманітних задача: пошуку, редагування, діалогу, аналізу тексту, прогнозу тощо. Зі створенням мережі Інтернет та можливостю взаємодії людини з іншою людиною у мережі, сфера обробки природної мови розширилася – тепер стало можливо оброблювати запити користувачів, аналізувати публікації користувачів, в тому числі електронну пошту.

У наші часи складно знайти людину, яка б користувалася комп’ютером, і не користувалася електронною поштою – це зручний спосіб комунікації, отримання інформації та навіть реєстрації у глобальній мережі. Такий інструмент дуже зручний, але з самого початку став ціллю багатьох злодіїв та шахраїв. За допомогою різноманітних методів інтернет-шахраї можуть проводити хакерські атаки, отримувати особисті дані та масові

розповсюджувати розсилки, як із звичайним текстовим змістом, так і з вірусами [4].

Проблема спаму – масового розсилання рекламних або інших текстів, існує і сьогодні. Від цього страждає багато людей, і спам-атаки наносять досить великі збитки як звичайним людям, так і організаціям. **Актуальність теми** дипломної роботи залишається високою, оскільки технічний захист конфіденційної інформації та особистого простору від злодіїв завжди мав пріоритет при розробці будь-яких систем.

Об’єктом дослідження є процес обробки природньої мови за допомогою машинного навчання.

Предметом дослідження є методи та алгоритми класифікації в додатках, які використовують обробку природньої мови.

Мета дипломної роботи – класифікація змісту електронного листа (аналіз семантики тексту) на наявність спаму з використанням методів та алгоритмів машинного навчання, за допомогою розробленої інтелектуальної системи на мові програмування Python, бібліотек Scikit-Learn та PyQt5.

Для досягнення зазначеної мети були визначені для виконання наступні задачі:

1) дослідження сфери застосування методів NLP;
2) аналіз наявних методів та алгоритмів машинного навчання для обробки природної мови;
3) освоєння бібліотек машинного навчання на Python;
4) проектування інтелектуальної системи з використанням алгоритмів машинного навчання;
5) тестування створеної системи.

1 ОБРОБКА ПРИРОДНОЇ МОВИ. ПОСТАНОВКА ЗАДАЧІ

1.1 Обробка природної мови та її задачі

Мова – складна система звукових і графічних знаків, яка має важливе соціальне призначення як засіб комунікації та пізнання для людей. Історія природніх мов дуже довга, і може складати сотні тисяч років; за такий час виникали одні мови, та зникали інші. У кожній мові існують свої символи, які найчастіше складають деякі слова – окремі одиниці мови, як найчастіше мають відношення до процесів, явищ, предметів у нашому світі, у свою чергу – набір слів складає речення.

Ще з минулих тисячоліть людина у деякому сенсі займалася обробкою мов – це можна помітити, наприклад, у давніх способах шифрування, які використовували математичні та статичні методи із особливостями мови, щоб маскувати сенс повідомлень. З розвитком математичних наук зростала складність обробки, а з появою перших обчислювальних пристроїв з’явилася можливість не просто автоматизувати цей процес, але й перейти на зовсім інший рівень.

Якщо коротко описати сутність NLP – це область у комп’ютерних науках, яка займається технічною обробкою природніх мов – представлення текстів у дані (зазвичай числові), інтерпретацію структурних частин, виділення зв’язків та сенсу між елементами тексту. За допомогою спеціальних методів та підходів – токенізації, лематизації, стемінгу, статистики, регулярних виразів та багато інших – комп’ютер має змогу розрізняти, інтерпретувати дані стосовно сенсу текстової інформації, виділяти необхідне та створювати власну базу знань [3, 5].

Якщо розглядати інтелектуальні системи на базі NLP, то можна сказати, що на їх вхід подаються деякі дані, необхідні для розв’язання задачі, в процесі обробки дані набувають зручного для машини виду: числові та символьні масиви; проходять необхідну кількість різноманітних маніпуляцій над ними,

частина яких може бути ітеративна і відбуватися до тих пір, поки не відбудеться деяка умова; кінцевий результат подається на вихід системи. Досить часто такі системи називають конвеєрами (див. рис. 1.1), через схожу схему роботи [3].

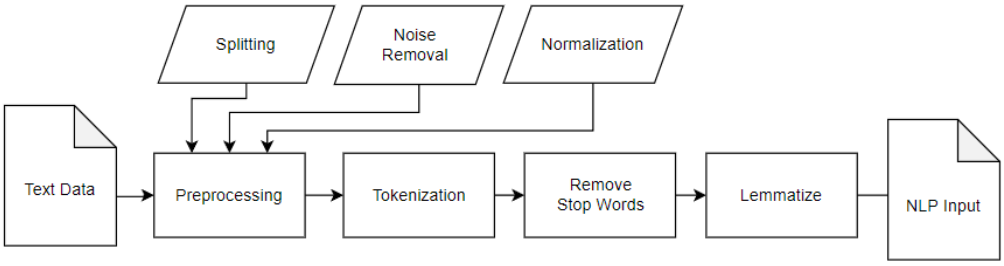


Рисунок 1.1 – Приклад роботи конвеєру

Такі системи досить часто використовуються у нашому світі. Наприклад, техніки NLP можуть використовуватися у сучасних додатках обробки текстів: для пошуку співпадінь слів, виділених частин, або автозаповнення тексту, у веб-браузерах: для пошуку в Інтернеті, пропозиції новин на базі пошукових даних, доповнення інформації (див. рис. 1.2), у сфері навчання – перевірка на плагіат, проведення експертиз, стильове оформлення тощо. Обробка природньої мови допомагає розв’язувати завдання бінарної та мультикласової класифікації, розпізнавання та синтезу мови, генерації нових даних та багато інших [3, 5].

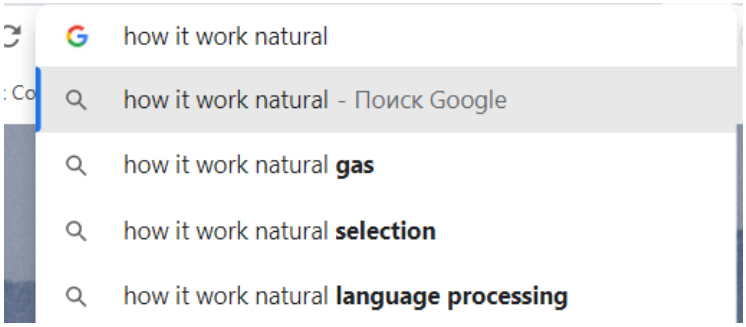


Рисунок 1.2 – Автоматичне пропонування доповнення тексту пошукового запиту у системі пошуку Google

Таким чином, методи та підходи NLP створюють досить складний та комплексний, але потужний механізм для технічної реалізації рішення багатьох задач, пов’язаних із природними мовами, текстовими даними.

1.2 Задача класифікації текстових даних

Задача класифікації – це розбиття деякої множини об’єктів, спостережень, даних на завчасно задані групи. Такі групи зазвичай називають класами, а спостереження мають однакові характерні властивості, ознаки, атрибути тощо, на основі яких і відбувається класифікація [6, 7]. Таким чином, деякі об’єкти, які мають схожі атрибути – будуть відноситися до одного класу, інші – до наступних класів.

Якщо кількість класів у задачі класифікації дорівнює двом, має місце бінарна класифікація, якщо більше – тоді багатокласова. В цілому класів може бути досить багато, наприклад, у задачі розпізнавання цифр в більшості випадків буде 10 класів, кожен з яких буде відповідати одній цифрі. У випадку класифікації видів тварин або рослин, таких класів може бути декілька десятків.

Класифікація тексту загалом не відрізняється від класифікації звичних даних по атрибутам, оскільки в деякому сенсі такими атрибутами являються частини мови – слова, або леми (базова форма слова) [7].

Класифікувати текст по сенсу допомагає семантика та статистичний аналіз. У контексті NLP терміни «семантика», «тема», «сенс» є досить схожими, і часто можуть заміняти один одного. Розпізнавати семантику тексту можна за допомогою латентно-семантичного аналізу, статистичних та ймовірнісних показників (TF-IDF, баєсів класифікатор) [3, 5]. Якщо максимально спростити, наприклад, класифікування тексту за допомогою статистичних показників, то можна навести в приклад токенізацію тексту, розрахунок входження усіх слів в текст, та виділення основних. Слова, які найчастіше зустрічаються у тексті можуть стати ключовим моментом до розкриття семантики, наприклад, якщо у тексті досить часто зустрічаються «games», «goal», «penalty» – навчена модель класифікатору може надати такому тексту клас «sport», оскільки ймовірно йде мова про футбол. Графічний приклад можна розглянути на рис. 1.3 на наступній сторінці.

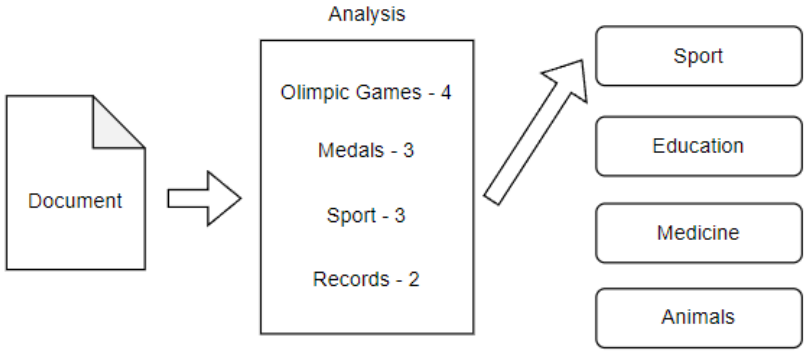


Рисунок 1.3 – Приклад класифікації на основі статистичного та семантичного аналізу

Звичайно, такі підходи є лише частиною усього процесу обробки природної мови, оскільки існують тексти різної складності, на різних мовах, які потребують своєї індивідуальної обробки. В такому випадку необхідне комплексне рішення, яке дозволить виявляти більше зв'язку та пов'язувати семантику між даними.

1.3 Підходи машинного навчання у NLP

Машинне навчання – підрозділ штучного інтелекту, у якому розглядаються методи та підходи побудови алгоритмів, які мають змогу до навчання, замість написання таких алгоритмів вручну [9, 10]. Підходи МН застосовуються у багатьох галузях, і досить часто безпосередньо у обробці природніх мов. У даному підрозділі будуть розглянуті підходи машинного навчання, які найчастіше використовуються при розв'язанні задач NLP з класифікації.

Баєсівський класифікатор. Наївний баєсів класифікатор – простий класифікатор, у якому використовується теорема Баєса, яка описує вірогідність події пов'язаних умов [11]:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \quad (1.1)$$

де A та B є подіями;

$P(A)$ та $P(\text{B})$ є ймовірностями A та B незалежні одна від одної;

$P(A|B)$ – умовна ймовірність A за умови істинності B;

$P(B|A)$ – умовна ймовірність B за умови істинності A.

Створити такий класифікатор можна з додавання міток класу екземплярам задачі, які представляються як вектори значень ознак. При цьому передбачається, що значення будь-якої заданої ознаки не залежить від інших, саме це припущення і складає «наївну» частину класифікатору [5].

Наприклад, існує деяка кількість даних для навчання, яка складається з різноманітних речень, із мітками «spam» та «not spam». Щоб застосувати даний алгоритм, необхідно виділити та розрахувати кількість входжень кожного слова у наборі даних, після розділити на відповідні класи та розрахувати ймовірність їх відношення до кожного.

Таким чином буде зіставлена матриця значень ймовірності (див. рис. 1.4) кожного слова для кожного класу, дані якої можна буде використовувати при класифікації вхідних даних, застосовуючи відповідну формулу 1.1.

Слово	Нормована ймовірність класу “Не спам”, 0.5	Нормована ймовірність класу “Спам”, 0.5
tomorrow	0.83	0.16
free	0.13	0.87
file	0.37	0.62
new	0.83	0.16
party	0.5	0.5

Рисунок 1.4 – Приклад створеної таблиці з нормованими ймовірностями класів

Даний класифікатор є досить простим та швидким у роботі, хоча має і свої недоліки. Наприклад, у ряді випадків сума ймовірностей може не дорівнювати одиниці – тоді необхідно нормувати результат, щоб привести до одиниці. Також, якщо не виконується припущення стосовно незалежності подій, модель може видавати досить неточні результати. Зазвичай дані недоліки перекриваються перевагами, серед яких простота, масштабованість, лінійна складність в залежності від розміру вибірки та помірні вимоги до пам’яті.

Існують також і інші алгоритми, які дозволяють вирішувати завдання класифікації таким чи іншим чином, але найбільшої популярності через свої можливості здобули нейронні мережі – підрозділ машинного навчання, який розглядає побудову обчислювальних систем, натхненних біологічними нейронними мережами людини [3, 5].

У нейронних мережах за основу взято ідею роботи клітини-нейрона: через дендрити у ядро поступають електричні сигнали (інформація), яка з часом накопичується, що призводить до збудження та відсилання електричного сигналу вже до аксону (див. рис. 1.5). Також можна відмітити, що клітини можуть бути більш, або менш чуттєвими до певних дендритів, що призводить до збудження із меншим сингалом [9].

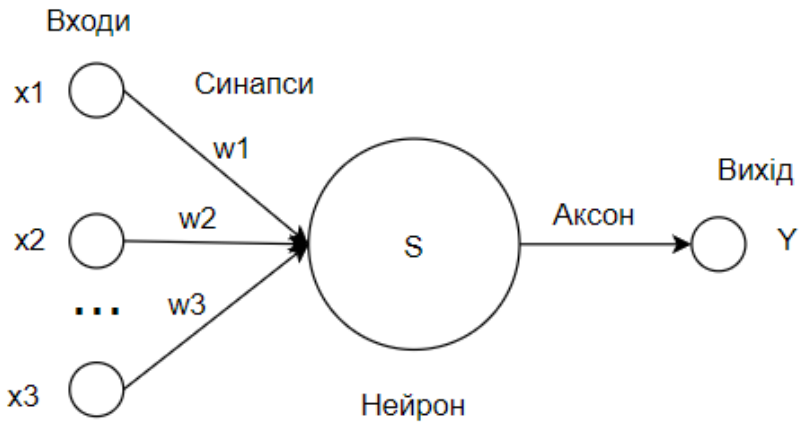


Рисунок 1.5 – Спрощений вигляд структури нейронної мережі

Звісно, реальні штучні нейронні мережі зазвичай мають набагато більше входів, кількість нейронів, а також їх шарів, та можуть мати не один вихід, а декілька. Архітектура нейронної мережі зазвичай залежить від складності задачі, а також безпосередньо її мети, оскільки кожна архітектура має кращі та гірші сторони.

Найпопулярнішими штучними нейронними мережами, які застосовуються у задачах NLP є згорткові нейронні мережі (англ. Convolutional Neural Network, CNN), рекурентні нейронні мережі (англ. Recurrent Neural Networks, RNN), та більш просунуті архітектури RNN: керований рекурентний

блок (англ. Gated Recurrent Units, GRU) та довга короткострокова пам’ять (англ. Long Short-Term Memory, LSTM) [9]. Для подальшого порівняння цих нейронних мереж буде наведена основна інформація, виділені недоліки та переваги.

Згорткові нейронні мережі. Дані мережі за останні роки стали досить популярними, в основному в області розпізнавання зображень – як кольорових, так і монохромних. Згорткові мережі отримали свою назву завдяки поняттю «згортки», даний процес можна порівняти із ковзанням рамки або вікна по заданому фрагменту даних, як це описує Коул Ховард у своїй книзі (див. рис. 1.6) [3].

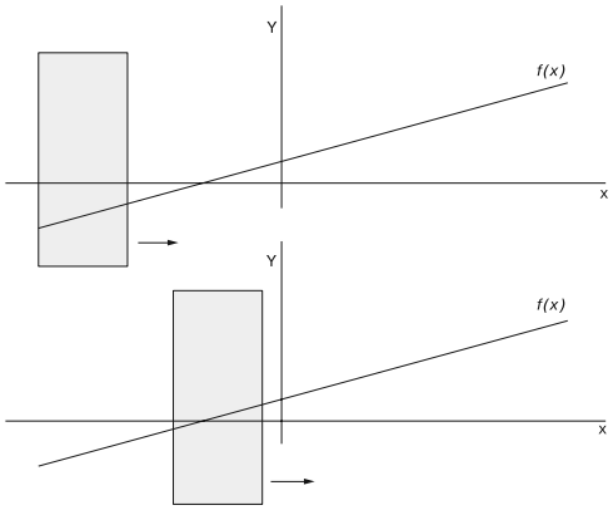


Рисунок 1.6 – Умовне представлення процесу згортки

Такі мережі мають ряд переваг із роботою зі специфічними даними, оскільки звичайні нейронні мережі ігнорують структуру вхідних даних, та перетворюють усі в одномірний масив. Для зображень, або схожих даних, таке представлення не є зручним, саме тому CNN стали досить зручним та популярним інструментом для роботи. Згорткові мережі як і звичайні складаються із нейронів, які мають вагу та зміщення. Основний нюанс роботи – у типі шарів та обробкою даних.

CNN враховує структуру даних, оскільки нейрони організовані у трьох вимірах: ширині, висоті та глибині, а кожен нейрон даного шару з’єднаний із фрагментом виходу попереднього. Таким чином дану структуру та роботу

можна порівняти із фільтром, який має розмір N на N. Враховуючи, що один фільтр не зможе обробити усі особливості зображення, даний процес необхідно повторити M разів (див. рис. 1.7). Використовуючи такий принцип, навіть прості згорткові мережі можуть уловлювати особливості як ребра та кути, якщо архітектура мережі ускладняється – це дозволяє виявляти ще більше ознак [3, 5].

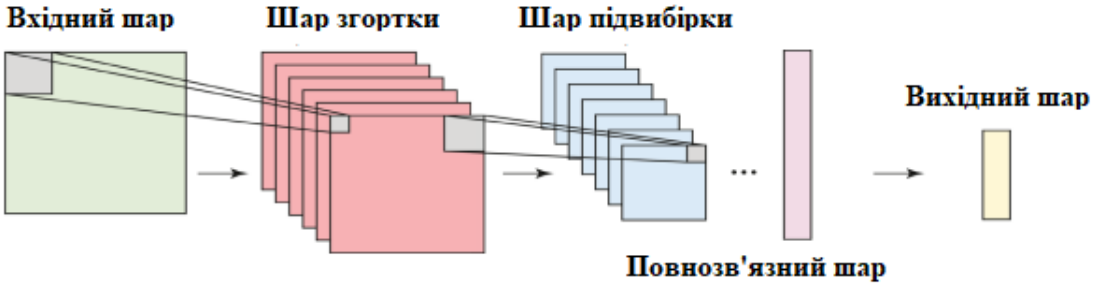


Рисунок 1.7 – Загальна структура згорткової мережі

Звісно, використовувати дані мережі можна і для інших задач, одна з яких – обробка текстів, у тому числі їх класифікація. Замість значень у комірках будуть вхідні дані із векторів слів [3]. В такому випадку сутність залишається не між вертикальними зв’язками, а горизонтальними (див. рис. 1.8).

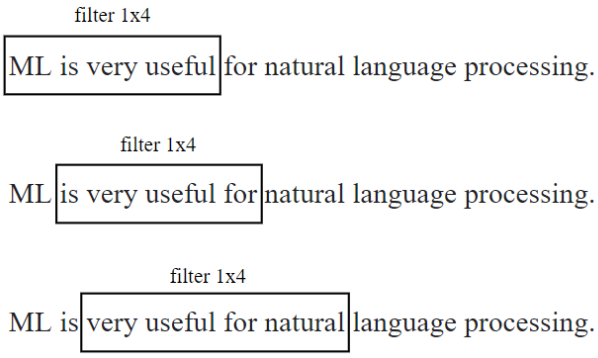


Рисунок 1.8 – Приклад застосування 1x4 фільтру в згортковій мережі

Таким чином, мережі CNN на базі згортки можуть стати корисним інструментом не тільки у класифікації зображень, але й текстових даних.

Рекурентні нейронні мережі. Якщо розглядати даний тип мережі, можна сказати, що рекурентні мережі – нейронні мережі, які можуть запам’ятовувати інформацію у ході їх обробки. Таким чином, стан нейронної мережі дозволяє

проявляти динамічну поведінку у часі, охоплюючи більше зв'язків між елементами даних, які поступають на вхід [10, 12].

Саме динамічній поведінці та уловлюванні зв'язку між деякою частиною даних у часі робить RNN унікальними. Можна привести у приклад речення, яке поступово подається на вхід, але не повністю. Без механізму запам'ятовування, система не буде мати змогу зрозуміти «сенс» даних, оскільки вона буде отримувати лише їх частину, без повної картини.

На рис. 1.9 зображено приблизну структуру рекурентної мережі. Можна побачити, що в цілому вона повторює вже розглянуті, за винятком наявних зв'язків у прихованому шарі.

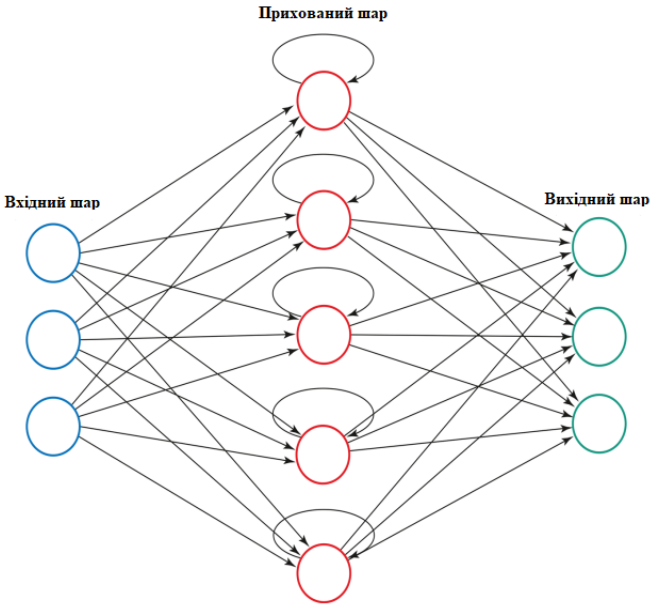


Рисунок 1.9 – Приклад внутрішньої структури рекурентних нейронних мереж

Якщо повернутися до прикладу на рис. 1.8, то можна пояснити роботу рекурентної мережі: при перегляді рядка з послідовністю літер «u», «s» та «e», вірогідно, що мережа запропонує для генерації наступні символи як «f» та «u», оскільки подібна послідовність вже зустрічалася у слові «useful».

Також непоганим прикладом може бути схема на рис. 1.10. Якщо розглядати таке представлення, то кожному уявному кроку t буде відповідати одна із згорток, яка представляє стан мережі у часі. Дане представлення можна порівняти із переглядом кіноплівки [3].

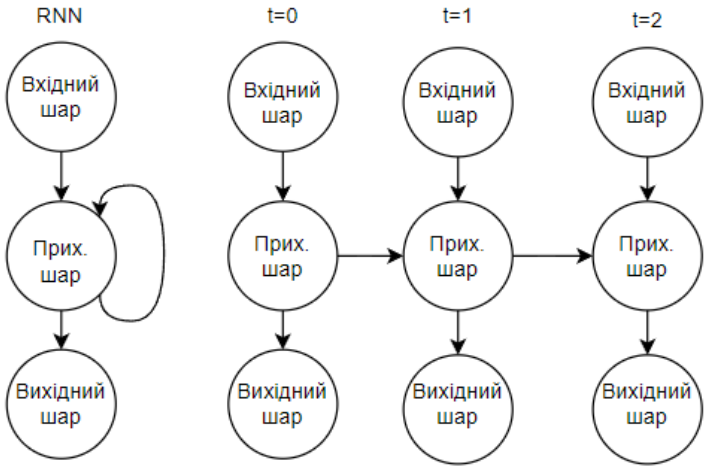


Рисунок 1.10 – Приклад розгорненої мережі

Рекурентні нейронні мережі являють собою дуже потужний інструмент для завдань розпізнавання: неперервного рукописного тексту, мовлення людини тощо, оскільки дані мережі можуть навчатися розпізнавати унікальні особливості хаотичних даних. Також варто відмітити застосування RNN при генерації осмислених послідовностей символів, зображень, машинний переклад текстів.

Система RNN досить складна, тому в залежності від потрібної довжини послідовностей токенів задача навчання та роботи мережі може значно ускладнюватися. Тим часом виникає і інша проблема – через наявність додаткової пам’яті значення градієнту (вектор похідних функції втрат) може як швидко затухати, так і експоненціально зростати, оскільки у деяких випадках глибина обробки нейронної мережі може зіставляти десятки і сотні шарів [12].

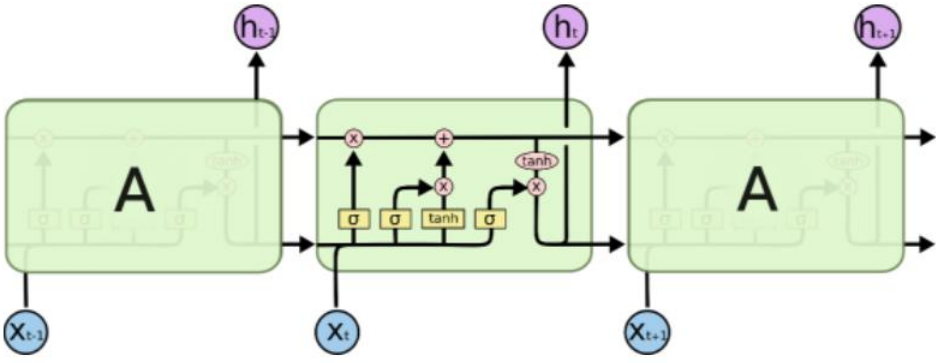
Мережі LSTM. Звичайні рекурентні мережі мають досить багато позитивних сторін, але в них все ще залишаються проблеми стосовно знаходження причинно-наслідкових зв’язків. Ефект токену може повністю втрачатися з часом, оскільки нові дані будуть затирати старі. Таким чином, зв’язок між семантично пов’язаними токенами (лемами) буде втрачатися, якщо вони знаходяться на достатній відстані один від одного [10, 12].

Можна розглянути все той самий приклад з рис. 1.8. Якщо між словами додати деякі нові, змінивши речення наступним чином: «ML, a sub-branch of

artificial intelligence, is very useful for natural language processing.», RNN вже не зможе так ефективно знайди зв’язок із токенами «ML» та «useful».

З цією задачею справляються спеціальні RNN-модифіковані мережі, такі як LSTM та GRU. За допомогою довгої короткострокової пам’яті вони допомагають зберігати деякі зв’язки між токенами, підтримуючи при цьому семантику між ними. Також дані мережі вирішують проблему затухаючих та зростаючих градієнтів, як зазначають винахідники даних мереж [13, 14].

Ключовим моментом у даній мережі є стан комірки (горизонтальна лінія зверху, див. рис. 1.11). Загалом стан клітин – внутрішньої пам’яті – нагадує конвеєрну стрічку, по якій переміщується інформація, над якою виконуються деякі операції. За допомогою таких операцій LSTM має змогу регулювати надходження даних; структурні елементи, які виконують такі операції називають воротами (англ. gates), які складаються з шару сигмовидної нейронної мережі та оператора перемноження. Сигмовидний шар пропускає усі значення при нулі, та нічого не пропускає, якщо значення є одиницею [3, 13].



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Рисунок 1.11 – Загальна структура роботи мереж LSTM

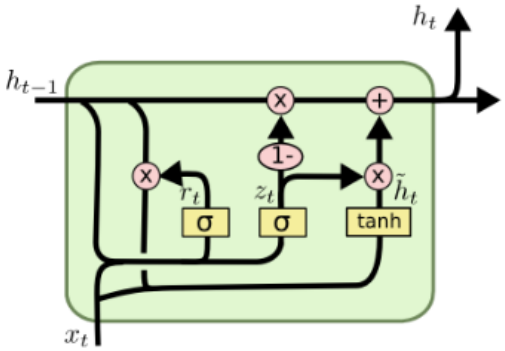
Складна структура нейронних мереж LSTM дозволяє працювати із довгостроковими залежностями, результат яких дуже відчутно у таких задачах як розпізнавання мовлення, музики, складних рукописних текстів, людських дій, генерації нових текстів на базі вже існуючих та багато інших [3, 5, 10, 13].

Варто звернути увагу, що мережі LSTM достатньо складні для застосування, потребують великих даних для навчання, і мають тенденцію перенавчатися у деяких випадках. Окрім цього, через складну архітектуру та розрахунки, мережі з короткострокової пам'яттю відповідно потребують більше часу для навчання та фізичної пам'яті комп'ютера.

Мережі GRU. Нейронні мережі GRU є такими самими RNN, які фактично є спрощеною версією LSTM. Замість трьох виходів дана мережа залишила усього два, також дещо спростивши внутрішню дію у клітині та розрахунки.

Мережі GRU мають менше параметрів, і як показали результати досліджень різних науковців та розробників, можуть бути більш ефективними у специфічних задачах, таких як моделювання та розпізнавання музики, обробки сигналів та природніх мов [13, 14].

Структура GRU може виглядати так, як вказано на рис. 1.12.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Рисунок 1.12 – Загальна структура мереж GRU

GRU запропонували науковці з університету Монреалю, такі як Кюнхьон Чо, Барт ван Меррієнбур та ряд інших, включаючи науковців з інших освітніх установ у 2014 році. В такому випадку можна стверджувати, що ця архітектура є достатньо новою, і не досить вивченою на сьогодні, хоча це не заважає створювати на основі неї і інші варіації, які виконують більш специфічні завдання, наприклад як адаптивний рекурентний блок – CARU (англ. Content Adaptive Recurrent Unit), який представили лише у 2020 році [14, 15].

У GRU такі самі переваги як у мереж LSTM, оскільки вони мають спільну структуру та спосіб роботи, але варто відмітити, що GRU є більш простою реалізацією RNN із короткостроковою пам’яттю, і може використовуватися у тих випадках, коли датасет являється меншим, ніж потрібно для навчання мереж LSTM. Дані нейронні мережі можна застосувати у випадках, коли переваги домінують над недоліками, такими як менша збіжність та точність.

1.4 Порівняльний аналіз підходів для класифікації у NLP

З попереднього підрозділу можна зробити висновки, що кожен підхід є ефективним у деякій задачі, та має власні переваги та недоліки. Щоб зробити порівняння простим та якісним, проаналізовані дані стосовно використання підходів машинного навчання в NLP для задачі класифікації будуть зібрані в одну таблицю (див. табл. 1.1).

Таблиця 1.1 – Порівняльний аналіз підходів машинного навчання

Назва	**Переваги**	**Недоліки**
Баєсів класифікатор	Проста реалізація, швидкодія, масштабованість, помірні вимоги до фізичної пам’яті. Може використовуватися при класифікації і видавати досить точні результати.	Необхідно враховувати можливість невиконання припущення про незалежність подій, а також перевіряти суму ймовірностей. Простий, і не може бути застосований до складних задач.
CNN	Відносно проста реалізація, уміння використовувати складну структуру текстових даних, виділяти основні ознаки, швидкість.	Складніша структура відносно простих алгоритмів, можлива втрата швидкості на великих об’ємах даних, неможливість знаходити, оброблювати документи змінної довжини.
Simple RNN	Вміння оброблювати документи змінної довжини, знаходження семантичних	Складність архітектури, більш затребувана у фізичній пам’яті, швидкість навчання та роботи залежить від глибини схованих

Кінець таблиці 1.1

Назва	Переваги	Недоліки
	зв’язків між токенами, динамічна поведінка у часі.	шарів та довжини вектору токенів.
LSTM	Більшість переваг звичайної RNN, вміння запам’ятовувати семантичні зв’язки між дальніми елементами, ефективна робота при	Повільна робота при великих даних відносно інших систем через складність структури, можливість перенавчання, затухання та зростання значень градієнту.
GRU	Переваги LSTM, зменшення впливу затухання та зростання градієнту, можливість навчання на менших вибірках	Основні недоліки LSTM через наявну складну структуру, використання у специфічних задачах

Виходячи з таблиці можна зробити висновки, що нейронні мережі типу RNN, LSTM, GRU є досить потужними інструментами, які підходять для складних задач та великих об’ємів інформації. Менш складні підходи, такі як баєсів класифікатор не можуть застосовуватися у ряді задач, але цілком ефективно та швидко можуть виконувати класифікацію об’єктів за деякими параметрами.

Згорткові мережі мають і достатньо складну архітектуру для обробки більш комплексних даних, і здатні виявляти деякі особливості структури.

Оскільки у даній роботі розглядається завдання розпізнавання семантики електронного листа на наявність спаму (його наявність або відсутність), має місце бінарна класифікація текстових даних. Використання складних рекурентних нейронних мереж має більше недоліків в такому випадку, ніж переваг, тому є сенс застосувати такі підходи машинного навчання як баєсів класифікатор, а також згорткові нейронні мережі, щоб оцінити можливу ефективність в обох випадках.

1.5 Специфікація розробки програмного рішення

Мета дипломної роботи – класифікація змісту електронного листа на наявність спаму за допомогою методів машинного навчання та NLP. Таким чином, для досягнення мети необхідно створити деяку інтелектуальну систему, яка буде використовувати технічні реалізації методів та алгоритмів як машинного навчання, так і обробки природньої мови.

Оскільки визначені роботи передбачають комплексний підхід для вирішення даної задачі, необхідно також визначити основні кроки, які повинні бути виконані при розробці програмного забезпечення:

1) вивчити основні техніки NLP та їх програмну реалізацію;
2) розглянути існуючі пакети та бібліотеки реалізації підходів штучного інтелекту, такі як TensorFlow, Keras, Scikit-learn тощо;
3) обрати та встановити інтегроване середовище розробки;
4) обрати мову програмування, на якій буде реалізований програмний додаток згідно технічного завдання (ТЗ);
5) розробити технічне рішення виконання поставленого завдання, при необхідності реалізувати користувальницький інтерфейс для взаємодії програми з оператором;
6) протестувати додаток на наявність похибок, помилок, правильності роботи тощо.

Таким чином, для реалізації інтелектуальної системи, окрім теоретичної частини, необхідно поставити технічне завдання. ТЗ встановлює основні показники якості, техніко-економічні та спеціальні вимоги, та призначення об’єкту розробки. Для даної системи можна визначити наступні ТЗ:

1) програмний додаток повинен працювати згідно встановленого алгоритму:
   – відкриття додатку;
   – аутентифікація користувача за допомогою поштового сервісу;

– ініціалізація пошти з аккаунту користувача;

– проведення аналізу з застосуванням навченої моделі для класифікації змісту пошти на наявність спаму;

– надання користувачеві результатів аналізу;

2) згідно обраної мови програмування реалізувати необхідні методи для роботи додатку, провести навчання та тестування моделі машинного навчання;

3) реалізувати графічний інтерфейс для спрощення взаємодії між системою та користувачем;

4) провести необхідні тестування додатку.

Основною мовою програмування було обрано Python. Python – високорівнева, інтерпретована мова програмування загального призначення. Python має динамічне типування (тобто, немає необхідності визначати тип кожної змінної), автоматичний збирач сміття, що допомагає в керуванні пам’яті, підтримує процедурне, об’єктно-орієнтоване та функціональне програмування. Таким чином, дана мова програмування є легким, але потужним засобом у розробці як простих, так і складних додатків [16, 17].

Ще однією необхідною та корисною особливістю є те, що Python являється відкритим програмним забезпеченням, що означає більше можливостей для розробників через доступність до сирцевого коду. Через це така мова програмування одразу стала популярною, і набула неймовірно великої бази різноманітних бібліотек, фреймворків та готових реалізацій, в тому числі методів та алгоритмів в області штучного інтелекту [17].

Для реалізації головної частини, як теоретичного, так і технічного завдання – розробки методів NLP у машинному навчання для задачі класифікації, було обрано бібліотеку Scikit-learn, яка допомагає у реалізаціях методів штучного інтелекту та машинного навчання, підтримує різноманітні конфігурації комп’ютерного обладнання, та має оптимізовані алгоритми роботи [19].

Для створення графічного інтерфейсу користувача (англ. Graphic User Interface, GUI) обрано пакет PyQt5, який доступний у Python. PyQt5 – дуже потужна бібліотека з великою кількістю інструментів, яка дозволяє створювати функціональні та практичні графічні додатки для систем Unix, MacOS, а також Windows, що дозволить запускати програмну на цих платформах без зайвих завантажень [19]. Дана бібліотека також має можливі додаткові фреймворки для реалізацій інтерфейсів за допомогою функцій «drag-and-drop», що значно пришвидшує створення графічних елементів інтелектуальної системи.

У якості поштового сервісу було обрано Google Mail (Gmail), який на даний момент є безсумнівно однією із найпопулярніших систем для зберігання, відправки та іншими операціями над власною поштою. Сервіси Google мають власну інтеграцію із іншими сервісами та системами цієї компанії, а також багатьма іншими, що робить дану платформу корисною для розробки. Для реалізації функціоналу доступу та маніпулювання поштовими даними використовується Gmail API – програмний інтерфейс додатку (англ. Application Programming Interface, API), який дозволяє застосовувати реалізації необхідних функцій та програмних класів даного сервісу [20].

Висновки до розділу 1

Із розвитком інформаційних та комп’ютерних технологій виникли ще більше сфер застосування методів обробки природної мови та машинного навчання. Методи NLP застосовуються у розпізнаванні, генерації, обробці, а також класифікації відповідних даних. Класифікування текстів можливо не лише стандартними засобами, але з використанням різноманітних алгоритмів машинного навчання та штучного інтелекту, які спрощують цей процес та дозволяють його автоматизувати. Велику роль в обробці природніх мов грають штучні нейронні мережі, які дозволяють вивести NLP на абсолютно новий рівень з досягненням результатів, які не були доступні у минулому.

Проблема спам-листів на сьогодні залишається, оскільки з розвитком інтернету та відповідних технологій знаходиться все більше способів надсилання даних листів користувачам, так і обходів різноманітних вбудованих спам-фільтрів, які існують у інформаційних системах.

Для вирішення даної задачі все більше застосовуються відповідні підходи NLP та машинного навчання. Варто відмітити, що для задач класифікації не обов’язково використовувати складні за архітектурою та роботою нейронні мережі, як наприклад, LSTM або GRU. Такі мережі можуть виявитися складними та повільними; для більш простих задач можуть застосовуватися і прості баєсів алгоритм, або згорткові мережі. Різні модифікації даних алгоритмів та способи їх застосування дозволяють швидко та з достатньою точністю класифікувати різноманітні текстові дані.

2 ТЕХНОЛОГІЇ ТА МЕТОДИ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОГО ЗАВДАННЯ

2.1 Підготовка та попередня обробка даних

Як було визначено у минулому розділі, класифікація передбачає ідентифікацію деякого класу, до якого відноситься об’єкт спостереження. Щоб розпочати процес, необхідно навчити модель – для цього необхідна деяка кількість даних, зазвичай, чим більше даних використовується – тим краще модель може навчитися, як результат – класифікація виявиться точнішою.

До первинних навчальних даних зазвичай застосовуються наступні етапи підготовки: вибірку, інтеграцію, очищення, перетворення даних, генерацію ознак, скорочення та форматування даних. Звісно, не завжди к даним застосовуються усі види перевірок. Для виконання задачі класифікації можуть застосовуватися наступні методи підготовки початкових даних [6].

Бінаризація. Цей процес досить простий і означає перетворення деяких кількісних показників у булеві значення, тобто 0 та 1. Такий метод може застосовуватися для більш простого представлення даних для навчання, або у якості заміни однієї категоріальної змінної на декілька числових.

Прикладом може стати заміна міток «спам» та «не спам», на одиницю та нуль. Або заміна декількох ознак «високий», «помірний», «низький» на відповідні масиви значень (див. табл. 2.1).

Таблиця 2.1 – Заміна категоріальних ознак на бінарні

Ознака	**Значення x1**	**Значення x2**	**Значення x3**
Високий	0	0	1
Помірний	0	1	0
Низький	0	1	1

Масштабування. Процес масштабування ще називають стандартизацію, або z-нормалізацією [5, 6]. Досить часто можуть виникнути ситуації, коли необхідно привести усі значення до деякого діапазону, наприклад від 0 до 1, або від -1 до 1. За допомогою масштабування (формула 2.1) кожна ознака не буде набувати надприродньо великого, або занизького значення, і матиме середнє значення 0, а стандартне відхилення – 1.

$$x_{i,st} = \frac{x_i - \mu}{\sigma}, \quad (2.1)$$

де $x_{i,st}$ – стандартизований елемент;

x_i – елемент стандартизації;

μ – середнє арифметичне;

σ – стандартне відхилення.

Як приклад можна взяти деякий вектор значень: [-5, 7, 1, 10, 3, -0.5, -8, 15], використавши стандартизацію до значень від 0 до 1.

Результат буде наступним: [0.13, 0.65, 0.39, 0.78, 0.48, 0.32, 0, 1].

Нормалізація. Процес нормалізації полягає у зміні значень вектору ознак деяким чином, щоб мати змогу застосувати одну шкалу для їх оцінки. У машинному навчанні зазвичай використовують L1-нормалізацію (формула 2.2), або L2-нормалізацію (формула 2.3) [5, 6].

$$|x|_1 = \sum_{j=1}^{k} x_j, \quad (2.2)$$

де $|x|_1$ – нормалізоване значення;

x_j – елемент нормалізації;

$\sum_{j=1}^{k} x_j$ – сума x_j елементів від j-елементу до k;

$$|x|_2 = \sqrt{\sum_{j=1}^{k} |x_j|^2}, \quad (2.3)$$

де $|x|_2$ – нормалізоване значення;

$|x_j|^2$ – значення елементу нормалізації у квадраті;

$\sqrt{\sum_{j=1}^{k} |x_j|^2}$ – корінь суми квадратів x_j елементів від j-елементу до k;

Як можна помітити, L1-нормалізація використовує метод найменших абсолютних відхилень (англ. Least Absolute Deviations, LAD), а L2-нормалізація – евклідову відстань.

Кодування міток. Сам сенс кодування простий – людина краще розуміє текстові дані, які мають деяке значення. Наприклад, значення 0, 1, 2 можна закодувати як «зелений», «жовтий», «червоний». Але на жаль, машина не розуміє слова, та в цілому використання міток на програмному рівні було б значно складніше та менш ефективніше, тому зазвичай кодують деякі символьні значення у цифрові.

Окрім методів попередньої обробки, до текстових даних зазвичай ставляться ще наступні умови:

– кількість об’єктів у вибірці повинна бути досить великою, оскільки модель може не навчитися на малій кількості даних;

– вибірка повинна вміщувати дані усіх можливих класах;

– відповідно, у початковому наборі даних повинно бути достатньо об’єктів кожного класу, щоб модель не перенавчалася на одному із класів.

Таким чином, знаючи одні із методів попередньої обробки, можна переходити до наступних кроків, серед яких – техніки обробки природніх мов.

2.2 Техніки обробки текстових даних у NLP

Щоб розпочати використовувати підходи NLP, необхідно знати деякі техніки обробки текстових даних. Оскільки текст не така проста структура, як може здаватися на перший погляд, існує досить багато методів, які допомагають у виділені семантичних зв’язків, або являються окремим інструментом для досягнення цілі виділити деякий сенс з тексту.

Прикладами таких технік є «мішок слів» (англ. Bag of Words, BOW), статистична оцінка відносності слова у документі (англ. Term Frequency-Inverse Document Frequency, TF-IDF), токенізація, стемінг, лематизація, видалення стоп-слів та багато інших.

Bag of Words. Даний метод являє собою мультимножину слів, яка міститься у всьому тексті (або документі), з розрахунком їх частоти. Це дозволяє досить зручно використовувати дані для машинного навчання [3, 5]. Наприклад, такий метод використовується у баєсовому класифікаторі.

Як приклад можна навести декілька простих речень:

Children play on the road.
The road to the airport is long.
Children played with a long lace.

Як можна побачити, у даних реченнях зустрічаються наступні слова, деякі з яких зустрічаються не один раз:

children, play, on, the, road, to, airport, is, long, played, with, a, lace.

У таблиці можна показати слова та кількість входжень кожного у тексті, тобто у всіх реченнях (див. табл. 2.2)

Таблиця 2.2 – Кількість входжень слів у тексті

children	play	on	the	road	to	airport	is	long	played	with	a	lace
2	2	1	3	2	1	1	1	2	1	1	1	1

Підрахувавши загальну кількість слів, можна сформувати «мішок слів», який набагато легше використовувати у подальших операціях. Наприклад, можна підрахувати входження слів у реченнях, і зробити висновок, що у першому та другому йдеться про відпочинок дітей, а у другому – про аеропорт. Звісно, такий метод занадто примітивний, але він використовується в інших концепціях.

TF-IDF. Являє собою статичну міру, яка зазвичай використовується для оцінки важливості деякого слова у документів, або колекції [3]. Дана міра складається із двох показників: TF (англ. Term Frequency) – визначає

важливість слова у тексті через відношення числа входжень, та IDF (англ. Inverse Document Frequency) – яка є оберненим значенням частоти, з якою слово може зустрічатися у колекції.

Таким чином визначення важливості слова є більш точним, оскільки сам визначним TF не розраховує це значення відносно усіх документів колекції, і окреме застосування якого могло б не відповідати дійсності через можливий великий розрив у кількості сторінок у кожному документі.

Обрахувати TF-IDF можна за формулою 2.4 [3]:

$$tf - idf(t, d, D) = \frac{n_t}{\sum_k n_k} \times log \frac{|D|}{|\{d_i \in D | t \in d_i\}|}, \quad (2.4)$$

де $tf - idf(t, d, D)$ – розраховане значення з параметрами;

t – слово у документі;

d_i – і-тий документ з колекції D;

D – колекція;

n_t – число входжень слова t у документ;

k – кількість слів у документі;

$|\{d_i \in D | t \in d_i\}|$ – число документів з колекції D, в яких зустрічається слово t;

Як приклад можна привести наявність 100000 документів, в 1000 з яких зустрічається слово «batman», у одному з документів, розміром 250 слів, слово зустрічається 5 разів. Таким чином, застосування методу буде наступним (див. формула 2.5).

$$tf - idf(t, d, D) = \frac{3}{250} \times log \frac{100000}{1000} = 0.012 \times 2 = 0.024 \quad (2.5)$$

Варто відмітити, що чим більше значення – тим важливішим є слово відносно документа до колекції.

Токенізація. Щоб запрограмувати минулі та наступні методи, слова (леми) у тексті вже повинні бути розбиті на деякі частини – терми, тобто фрагменти тексту. У цьому допомагає токенізація, яка і має у собі поняття розбиття тексту на токени. Варто відмітити, що токенами не обов’язково

повинні бути лише слова по одинці, можна враховувати також і групи слів. Якщо токен містить два слова, таку сполуку називають біграмою, якщо три – триграмою, чотири сполуки – 4-грама і так далі.

Приклад токенізації фактично вже було розглянуто у минулому підрозділі, де йшла мова про «мішок слів», а використання n-грам на рисунку 1.8.

Стемінг. Процес стемінгу – це групування початкових форм слова по деяким кластерам [3, 5]. Такий метод дуже корисний, наприклад, якщо потрібно прибрати закінчення у слові, щоб замість «played» було «play», тобто прибрати закінчення слова «ed». Варто відмітити, що просто прибрати закінчення – не панацея в даному випадку. Можна привести слова «flying» та «running» із закінченням «ing». Просто видаливши закінчення – не усі форми будуть вірними.

Методи стемінгу дозволяють визначати семантичні основи слів без використання регулярних виразів. Існує достатня кількість різних реалізацій, серед яких стемери Портера, Ланкестера та Сноубола. На рис. 2.1 буде наведено приклад із реалізації процесу стемінгу на Python [5].

INPUT WORD	PORTER	LANCASTER	SNOWBALL
writing	write	writ	write
calves	calv	calv	calv
be	be	be	be
branded	brand	brand	brand
horse	hors	hors	hors
randomize	random	random	random
possibly	possibl	poss	possibl
provision	provis	provid	provis
hospital	hospit	hospit	hospit
kept	kept	kept	kept
scratchy	scratchi	scratchy	scratchi
code	code	cod	code

Рисунок 2.1 – Приклад роботи різних методів стемінгу

Як можна побачити, різні методи можуть видавати різні результати, тому варто ретельно аналізувати результати і використовувати ті алгоритми, які доречні у ситуації. Також, звертаючи увагу на коментар автора Коула Ховарда, він просить максимально можливо не використовувати стемінг та лематизацію,

особливо у невеликих текстах, оскільки дані процеси можуть зменшувати чуттєвість процесу обробки та знижувати рівень точності [3].

Лематизація. Ще один метод, який дозволить отримати зі слова його кореневу частину – лематизація. Це більш точний спосіб, оскільки стемінг може обрізати частини слів таким чином, що перетворює їх у неіснуючі леми. На рис. 2.1 можна побачити такий приклад на слові «calv» та «hors» [3].

Алгоритми лематизації використовують більш складні та точні методи отримання кореневих слів, на рис. 2.2 можна побачити результат виконання процесу лематизації на тих самих словах, які були наведені у минулий раз.

INPUT WORD	PORTER	LANCASTER	SNOWBALL
writing	write	writ	write
calves	calv	calv	calv
be	be	be	be
branded	brand	brand	brand
horse	hors	hors	hors
randomize	random	random	random
possibly	possibl	poss	possibl
provision	provis	provid	provis
hospital	hospit	hospit	hospit
kept	kept	kept	kept
scratchy	scratchi	scratchy	scratchi
code	code	cod	code

Рисунок 2.2 – Приклад роботи різних алгоритмів лематизації

Як вже згадувалося – лематизацію краще використовувати у випадках з невеликими текстами, та при потребі, оскільки велика кількість нових слів та жаргонізмів може приводити до збою алгоритмів та втрати точності.

Видалення стоп-слів. Стоп-слова в реченнях мають роль зв’язки, і у деяких випадках не грають особливої ролі. Видаленням стоп-слів можна досягнути необхідного ефекту, зменшивши кількість лем без особливого лексичного та семантичного сенсу [3, 10].

Такими словами можуть бути «a», «an», «the», «of» та інші аналоги в природніх мовах. Але необхідно пам’ятати, що стоп-слова у зв’язці з лемами можуть створювати сенсові словосполуки, які втрачають своє семантичне навантаження з видаленням таких слів. Тобто, використовуючи токенізацію по одному слову, можливо, видалення стоп-слів буде мати потрібний ефект, але з використанням n-грам це стає більше проблемою, ніж вирішенням питання.

Використовуючи вже готові рішення в програмних реалізаціях також варто перевіряти списки стоп-слів, оскільки природні мови досить часто розвиваються, можуть з’являтися нові, або навіть видалятися старі, враховуючи різні мови та версії пакетних модулів.

2.3 Процес навчання у штучному інтелекті

Для того, щоб вирішити деяке завдання завдяки штучному інтелекту, найчастіше потрібно не тільки застосувати деякий алгоритм, але й створити вірну модель. Як зазначалося раніше, велику роль в цьому процесі грають безпосередньо дані – оскільки від них залежить подальший результат. Не коректні або неповні дані можуть призвести до неповноцінного результату, або взагалі неможливості вирішити завдання.

Але дані – не єдина умова ефективної роботи, модель потрібно «навчити» роботи те, для чого вона створювалася. Існує досить багато методів навчання різних алгоритмів та штучних нейронних мереж, але досить часто вони відносяться до двох великих груп: навчання з вчителем, та навчання без вчителя [3, 5].

Навчання з вчителем. Можливо, найпоширеніший метод навчання, це навчання з вчителем. Метою даного процесу являється попереднє маркування вірних та хибних відповідей у даних. Як приклад можна навести використання у задачах регресії, апроксимації, класифікації та розпізнавання образів, прогнозування.

При даному процесі модель навчається поступово, порівнюючи результат на виході системи з маркуванням, при досягненні деякого значення функціоналу якості – значенню, яке характеризує точність системи, навчання припиняється.

Частковим варіантом навчання з вчителем є **навчання з підкріпленням**. Сутність даного методу є відсутність точних маркувань правильної або

неправильної відповіді. Можна сказати, що в даному методі машина повинна «в сліпу» знаходити вірне рішення, аналізувавши отримувані результати.

В цьому випадку машина завжди знаходиться в деякому векторі станів, модель може виконувати визначені дії – та отримувати винагороду, перевестись у інший стан. Дія являється оптимально у випадку, коли винагорода максимізується.

Навчання без вчителя. Даний процес є протилежний минулому, при навчанні без вчителя у початкових даних повністю відсутні якісь мітки. Машина на основі цього методу повинна сама визначати закономірності у даних, та розподіляти по можливим ознакам згідно способу, який визначено оптимальним у такому випадку.

Даний метод навчання зазвичай використовується при рішенні задач класифікації (об’єктів, тексту, зображень тощо), пошуку асоціативних правил, скорочення розмірності тощо.

2.4 Наївний баєсів класифікатор

Як вже було сказано, баєсів класифікатор використовує теорему Баєса про ймовірність події, яка описується формулою 1.1. Щоб мати змогу в подальшому реалізувати даний алгоритм, необхідно розібратися на прикладі у деталях його роботи.

Нехай згідно задачі необхідно класифікувати деяку кількість речень на наявність спаму з відповідними мітками (ham – відсутність спаму):

[SPAM] Free clothes during a month
[SPAM] Incredible clothes discounts
[HAM] Clothes care tips
[HAM] Weather this month
[SPAM] Super discounts absolute free
[HAM] Absolute win this match

Для початку необхідно створити частотний словник, із словами, які зустрічаються в усіх реченнях, та кількості входжень кожного. В цьому нам

допоможе метод BOW, також за допомогою видалення стоп-слів можна прибрати деякі з тих, які для такого завдання не несуть якогось семантичного навантаження, тобто «a» та «this».

Згідно аналізу усі інші слова будуть додані до мішку та обраховано їх частоту, результати наведені у табл. 2.3.

Таблиця 2.3 – Словник слів та їх частота

free	**clothes**	**during**	**month**	**incredible**	**discounts**	**care**
2	3	1	2	1	2	1
tips	**weather**	**super**	**absolute**	**win**	**match**	
1	1	1	2	1	1	

Наступним кроком згідно алгоритму є розподілення згідно класів кількості входжень кожного слова, та розрахунок імовірностей (див табл. 2.4).

Таблиця 2.4 – Словник слів з розрахованими кількостями та ймовірностями

Слово	**Кількість у класі SPAM**	**Розрахована ймовірність (SPAM)**	**Кількість у класі HAM**	**Розрахована ймовірність (HAM)**
Free	2	1	0	0
Clothes	2	0.66	1	0.33
During	1	1	0	0
Month	1	0.5	1	0.5
Incredible	1	1	0	0
Discounts	2	1	0	0
Care	0	0	1	1
Tips	0	0	1	1
Weather	0	0	1	1
Super	1	1	0	0
Absolute	1	0.5	1	0.5
Win	0	0	1	1
Match	0	0	1	1

Як можна побачити, дана таблиця дійсно відображає сутність роботи баєсового класифікатору на статистичних основах, таким чином, якщо у

реченні зустрічаються слова «free», «clothes», «discounts» – вірогідно, що текст являється спамом. Звісно, це лише тестовий приклад, і наявних у ньому слів недостатньо для створення повноцінної моделі.

Як можна побачити, деякі слова мають нульову вірогідність відношення до визначеного класу, це може призвести до невірних розрахунків, оскільки у такому разі вірогідність теж може дорівнювати нулю. Щоб цього уникнути, можна застосувати нормування (див. формулу 2.6) [21].

$$p_{norm} = \frac{n \times p_{word} + p_{class}}{n+1}, \quad (2.6)$$

де p_{norm} – розраховане значення нормованої ймовірності;

n – повторень слова у вибірці;

p_{word} – ненормована ймовірність слова;

p_{class} – ймовірність класу;

Ймовірність класу у даному випадку дорівнює 0.5, оскільки у наявності лише два класи, які мають рівну кількість повідомлень у початковому наборі даних. Результати застосування формули наведені у табл. 2.5.

Таблиця 2.5 – Нормовані ймовірності класів

Слово	**Нормована ймовірність (SPAM)**	**Нормована ймовірність (HAM)**
Free	0.83	0.16
Clothes	0.62	0.38
During	0.75	0.25
Month	0.5	0.5
Incredible	0.75	0.25
Discounts	0.83	0.16
Care	0.16	0.83
Tips	0.16	0.83
Weather	0.16	0.83
Super	0.75	0.25
Absolute	0.5	0.5
Win	0.16	0.83
Match	0.16	0.83

Таким чином, за допомогою статистики та технік NLP була створена таблиця нормованих ймовірностей слів для кожного класу. За допомогою цієї таблиці розрахувати приблизне значення відповідності до класів буде значно простіше, ніж будь-яким іншим способом.

Щоб розрахувати загальну ймовірність для обох класів, необхідно використати формулу 2.7:

$$p_{class\_letter} = p_{class} \times \prod_{i=1}^{n} p_{i,norm}, \quad (2.7)$$

де p_{class_letter} – розраховане значення ймовірності належності до класу;

p_{class} – ймовірність класу;

$\prod_{i=1}^{n} p_{i,norm}$ – сума ймовірностей слів у реченні;

$p_{i,norm}$ – нормована ймовірність слова;

Таким чином, використавши формулу 2.6 для кожного класу, можна з’ясувати ймовірність входження тестового повідомлення до одного з них. Для прикладу можна навести два речення:

Super discounts this month
Win clothes this match

Застосувавши два рази формулу для кожного речення можна буде знайти показники вірогідності (див. формули 2.8-2.11).

$$p_{1,spam} = 0.5 \times (0.75 \times 0.83 \times 0.5) = 0.156 \quad (2.8)$$

$$p_{1,ham} = 0.5 \times (0.25 \times 0.16 \times 0.5) = 0.01 \quad (2.9)$$

$$p_{2,spam} = 0.5 \times (0.16 \times 0.62 \times 0.16) = 0.008 \quad (2.10)$$

$$p_{2,ham} = 0.5 \times (0.83 \times 0.38 \times 0.83) = 0.131 \quad (2.11)$$

Отже, застосувавши наївний баєсів алгоритм і отримавши результати роботи, можна сказати, що задача класифікації листів на тестовому прикладі вирішена. Алгоритм дав досить точні результати на наявних даних, хоча варто відмітити, що вони були обрані виключно для прикладу.

Даний алгоритм є досить популярним, і застосовується у різних сферах для задач класифікації через свою простоту та ефективність. Для баєсівського класифікатора існують різноманітні модифікації, які дозволяють урахувати різні фактори відносно сфери дослідження, що дозволяє використовувати даний підхід і у більш складних задачах.

2.5 Згорткові нейронні мережі

Як же було згадано, нейронні мережі були створені на уявлені про те, як працює людський мозок: наявність нейронів, зв’язків між ними та передачі сигналів. Один із недоліків «звичайних» нейронних мереж (наприклад, перцептрону) – ігнорування структури вхідних даних.

Згорткові нейронні мережі мають досить складну структуру, яка складається шарів декількох видів, операцій згорток тощо. Даний процес робить обробку даних, які мають специфічну структуру, ефективнішою, та дозволяє збирати більше ознак. Для подальшої реалізації даної мережі необхідно розібрати такі ключові моменти, як її топологію, види та роботу шарів, навчання мережі тощо.

Загальна архітектура мережі. Архітектура CNN має принципіальні відмінності від звичайних мереж, де зазвичай існує вхідний, прихований та вихідний шари, кожен або деякі нейрони яких пов’язані з нейронами інших шарів. Якщо розглядати архітектуру на прикладі класифікації зображень, можна навести топологію, яка зображена на рис. 2.3.

В цілому, робота згорткової мережі на зображенні та тексті досить схожа, оскільки структура даних, які подаються на вхідний шар залишається такою ж. На рис. 1.7 у першому розділі вже наводився більш простий приклад архітектури, але оскільки ці шари мають власні принципи роботи, необхідно розглянути кожен і дослідити їх вплив на загальний результат у мережі.

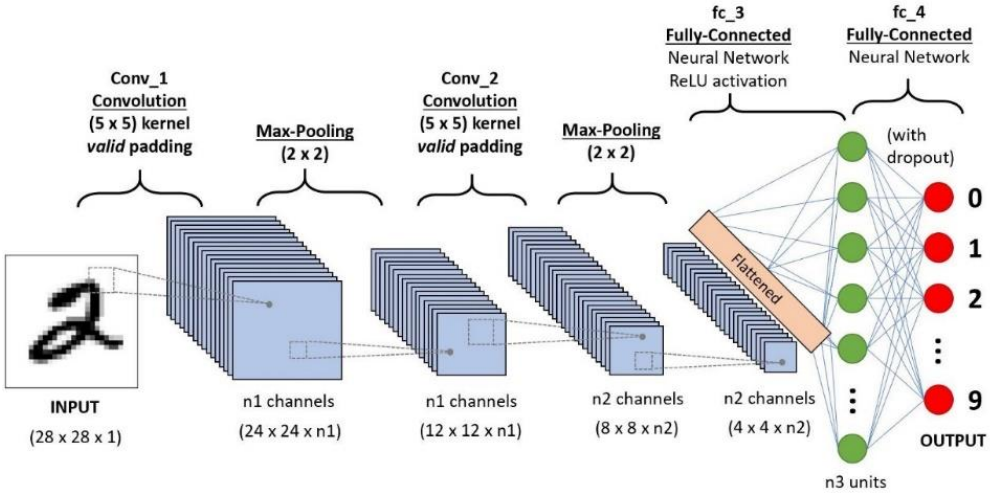


Рисунок 2.3 – Топологія CNN на прикладі розпізнавання зображень

Вхідний шар (Input Layer). Вхідний шар приймає відповідні значення з поданих даних, це можуть бути значення пікселів на фото, символи тексту або його фрагменти. Для складних даних, як зображення, «глибина» шару може залежати від кількості фільтрів (RGB) та інших показників, для текстових даних може цілком вистачати одного. В такому разі вхідний шар являє собою деяку матрицю N на M елементів.

Після вхідного шару дані «згортаються» у наступний – шар згортки.

Шар згортки (Convolutional Layer, Kernel). Даний шар обраховує так звані згортки між нейронами вхідного шару та деякими фрагментами (можна назвати їх фільтрами).

Фільтри складаються із деякого набору ваг та функції активації. Загалом, дані набори фільтруючих нейронів схожі на звичайні, які відносяться до прихованого шару, але ваги цих фіксовані на весь «прохід» по шару. Усі фільтри в згорткових мережах унікальні, але кожен має окремий елемент фільтру, який зафіксовується у час «знімку» даних [3]. Щоб краще зрозуміти логіку, можна розглянути приклад роботи фільтрів на рис. 2.4 [22].

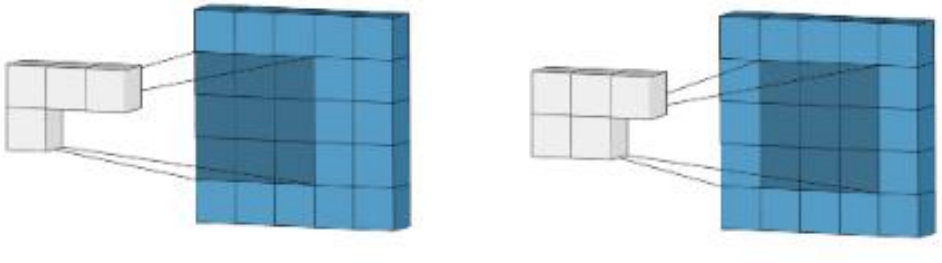


Рисунок 2.4 – Принцип роботи фільтрів у шару згортки

Таким чином, кожен із фільтрів ітеративно проходить по вхідним даним та заповнює власну матрицю, перемножуючи значення із вагами, які відповідають положенню фільтра (див. рис. 2.5).

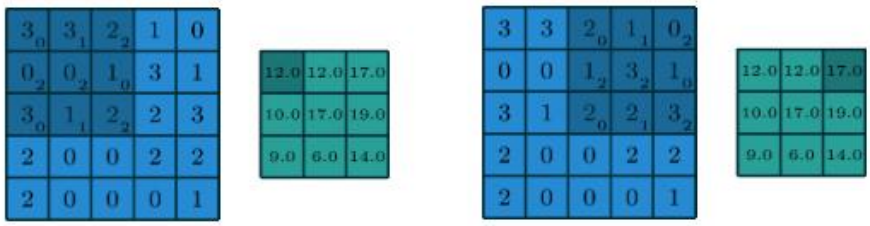


Рисунок 2.5 – Розрахунки фільтру

Даний процес має декілька технік [3]: padding та striding. Padding (можна перекласти як «набивки») дозволяє зменшувати (або збільшувати) обчислення для матриці ознак, «обрізаючи» краї зони, яку може охопити фільтр, але розмір фільтру при цьому не зменшується (див. рис. 2.6).

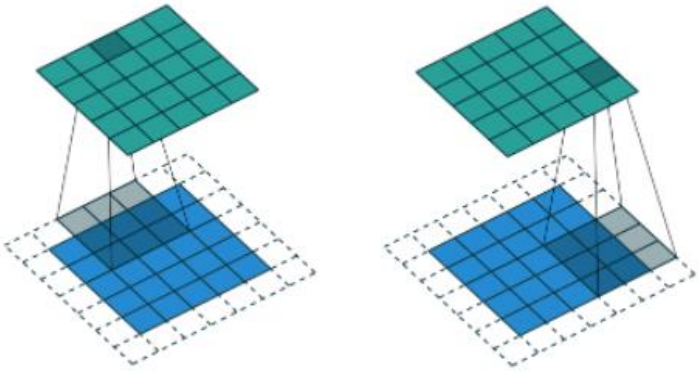


Рисунок 2.6 – Принцип роботи padding

У той час значення striding – кроку фільтру – дозволяє проходити шар даних швидше. На рис. 2.7 зображено проходження фільтру з кроком 2, таким чином зміщення відбувається не на одну умовну клітину матриці, а на дві.

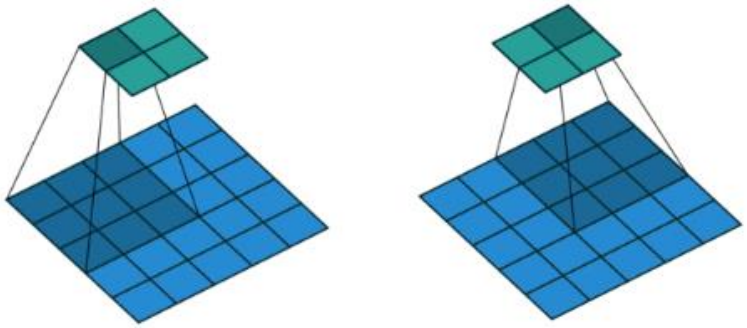


Рисунок 2.7 – Принцип роботи striding

Шар згортки досить важливий, оскільки дозволяє, по-перше, фільтру охоплювати різні високорівневі ознаки даних. На зображенні це можуть бути кути, а у тексті – синтаксичні зв’язки, якщо вони існують.

Також згортка зменшує обчислювальне навантаження про процес навчання мережі, оскільки дозволяє проводити меншу кількість обрахунків або дозволяти фільтру проходити шар вхідних даних швидше. Це контролює кількість наявних ваг у мережі, що відповідно зменшує кількість параметрів та прискорює роботу.

Після згортки дані, які отримав фільтр сумуються та передаються на наступний шар.

Шар скоригованих лінійних блоків. Даний шар (англ. Rectified Linear Unit - ReLU) застосовує деяку функцію активації до вихідного сигналу (з попереднього шару). Він існує для того, щоб була можливість внести нелінійність в мережу, яка буде допускати узагальнення на будь-який тип функції [5].

Шар об’єднання (Pooling). Пулінговий шар, як його іноді називають, дуже схожий на шар згортки. Він вибирає із вже існуючих ознак, які були обрані минулим шаром, найосновніші, зазвичай за допомогою деякої функції активації. Часто це max-функція, яка з набору значень обирає максимальне, саме тому шар об’єднання часто позначається Max-Polling.

За допомогою згортки та шару об’єднання у процесі навчання модель має змогу успішно вивчити усі ознаки, для закінчення процесу усі дані подаються у повнозв’язний шар, який у цілому є звичайною нейронною мережею.

Повнозв’язний шар (Fully-Connected). Даний шар дозволяє досить просто вивчити нелінійні комбінації високорівневих ознак, які представляються у результаті минулих операцій.

На вхід у нейрони подаються значення, ініціалізуються ваги, та за допомогою одних з методів навчання (зазвичай це метод оберненого розповсюдження помилки, англ. Backpropagation) налаштовуються параметри моделі, які передаються у вихідний шар. Для прикладу цього шару буде розглянуто навчання моделі звичайного перцептрону за допомогою методу Backpropagation.

Загальний алгоритм навчання наступний:

1) ваги мережі ініціалізуються деякими невеликими значеннями;

2) обирається вибірка для навчання, подається вектор значень на вхід системи;

3) обчислюється вихід мережі згідно розрахункових формул (див. рис. 2.8);

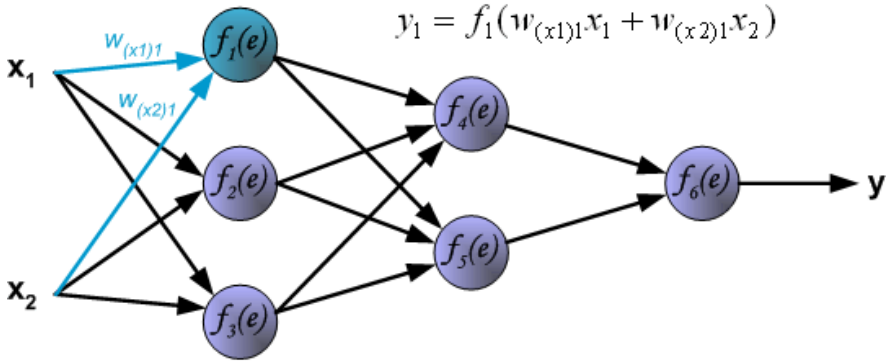


Рисунок 2.8 – Розрахунок вихідних значень нейронів

4) обчислюється різниця між цільовим значенням та результуючим – помилка (див. рис. 2.9);

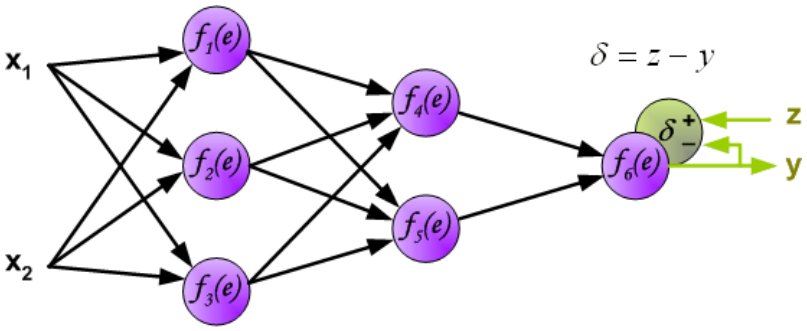


Рисунок 2.9 – Розрахунок помилки

5) використовуючи обернені дії помилка розраховується для кожного нейрону усіх шарів (див. рис. 2.10);

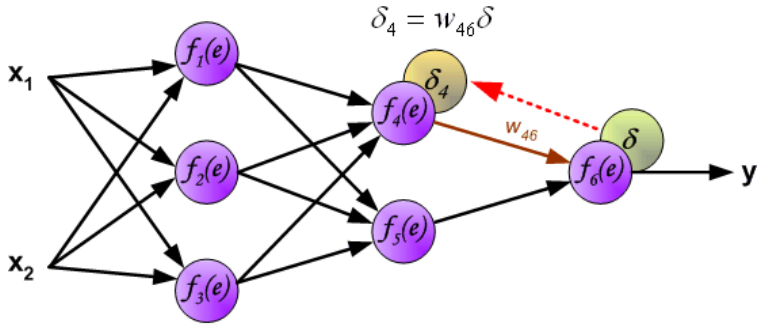


Рисунок 2.10 – Розрахунок помилки для кожного нейрону

6) кроки 2-5 повторюються, поки значення помилки не досягне еталонного значення, яке задається оператором.

Вихідний шар. На вихід отримується вектор значень, які дорівнюють кількості класів. Таким чином навчена нейронна мережа дозволяє класифікувати наступні дані, що будуть надходити на вхід системи.

Як можна зрозуміти, нейронні мережі мають набагато складніший механізм роботи, та потребують більше часу для вивчення та налаштування. Згорткові нейронні мережі можуть використовуватися для обробки текстових даних, але варто відзначити більшу складність обчислень для простих задач, як класифікація. Додаткові шари дозволяють як спрощувати обчислення та знаходження семантичних зв'язків, так і збільшувати їх кількість шляхом додавання нових шарів, поглиблюючи мережу.

В основному, згорткові мережі мають сенс, якщо потрібно знайти семантичні зв'язки окремих елементів у тексті.

Висновки до розділу 2

Процес NLP досить складний, але має зрозумілі техніки, які можна використовувати для різноманітних задач. Більш того – дані підходи є досить розповсюдженими і використовуються не тільки у нейронних мережах, але і більш простих алгоритмах. Основними є: токенізація, «мішок слів», видалення-стоп слів, стемінг, лематизація, а також використання статистичних та ймовірнісних показників (TF-IDF). Вони дозволяють проаналізувати різні ознаки, і визначити семантичні зв’язки між даними.

При реалізації нейронних мереж, важливо не тільки обрати правильну архітектуру, але й підготувати дані – оскільки правильно підібрані навчальні та тестові вибірки означають, наскільки точним буде вихідний результат. Важливо пам’ятати і про процес навчання мережі, оскільки навіть із відповідними вибірками, погано налаштована мережа не зможе правильно класифікувати дані.

Розв’язати задачу класифікації текстових даних можуть методи машинного навчання, наприклад – баєсів класифікатор та штучні мережі. Для відносно простих задач баєсів класифікатор може показувати гарні результати. Згорткові нейронні мережі можна використовувати не тільки для розпізнавання образів на картинках, але й символьних даних, вони суміщають як урахування складної структури тексту, так і швидку роботу.

Якщо розглядати бінарну класифікацію змісту електронних листів, то в даному випадку є сенс використання більш простого алгоритму – баєсового класифікатору. Разом із техніками обробки природної мови, він дозволяє знаходити семантичні зв’язки шляхом урахування частотності слів, окремі модифікації можуть покращувати його можливості у розпізнанні, дозволяючи збільшити точність класифікації.

3 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ З ВИКОРИСТАННЯМ PYTHON ТА БІБЛІОТЕКИ SCIKIT-LEARN

3.1 Мова програмування Python та її можливості

Мова програмування Python – інтерпретована об’єктно-орієнтовна, високого рівня із суворою динамічною типізацією. Через наявність динамічної типізації, семантики та зв’язування, а також структур даних високого рівня, роблять дану мову гарним кандидатом для швидкої розробки програм, або використання як потужного інструменту написання системних скриптів [17].

Python підтримує модулі та пакети, які дозволяють розмежувати необхідний код, підключати лише ті програмні компоненти, які необхідні у тому чи іншому застосунку. Такі модулі можуть бути завантажені одразу, або при необхідності – що дозволяє зменшити розмір майбутнього додатку. Також варто відмітити, що кожен користувач може створити власний модуль та поширювати його, оскільки Python має платформу із відкритим кодом [17].

Дана мова програмування стала популярною не лише через зручний синтаксис та швидке написання програм, але й через власну портованість, тобто можливість запуску на визначених платформах, система – дана мова існує майже на будь-якій, починаючи від мобільних пристроїв, та закінчуючи мейнфреймами та серверами. На даний момент наявна підтримка на Microsoft Windows, UNIX-системах, Mac OS, iPhone OS, Symbian, Android та багато інших [16,17].

Історично склалося, що Python у ході розробки розділився на дві гілки: 2.x версій та 3.x, які до 2020 року розроблювалися паралельно. Це давало можливість підтримувати більше пристроїв та проектів, які були створені багато років тому, не заставляючи компанії переходити на іншу версію, оскільки це могло б погіршити ситуацію із підтримкою старих версій, та перехід на більш нові – для великих проектів це завжди складний та багатоетапний процес, і може займати місяці, а можливо і роки.

Можливості та переваги. Вище наведені основні можливості мови програмування, але для повного розуміння ключових особливостей, необхідний короткий аналіз усіх моментів програмування. Python, як і більшість інших мов, має ряд необхідних технічних та програмних інструментів та використовує підходи, які напрацьовані вже багато років у інших мовах.

У таблиці 3.1 буде наведено короткі відомості як технічних, так і програмних переваг даної мови [16, 17].

Таблиця 3.1 – Перелік можливостей мови програмування Python

Назва	Коротке пояснення
Інтерактивний режим	Даний функціонал передбачає можливість введення з клавіатури деяких виразів, частин коду, які програмне забезпечення одразу виконує та виводить результати роботи. Ця особливість робить з Python потужний інструмент для створення та виконання скриптів, які корисні при роботі адміністраторів, науковців та рядових користувачів.
Об’єктно-орієнтоване програмування	Дана мова програмування підтримує реалізацію ООП принципів, а саме: класи та об’єкти, звичайне та множинне успадкування, поліморфізм, інкапсуляцію, конструктори, деструктори, розподілення пам’яті, перевантаження операторів, управління доступу до полів, метапрограмування тощо.
Функціональне програмування	Відповідно, існує підтримка функціонального програмування – функція як об’єкт, функції вищих порядків, наявність рекурсії, обробка списків, замикання та інші.
Модульність та пакети	Програми на Python оформлюються у вигляді модулів, які потім можуть бути сформовані у пакети. Модулі можуть розташовуватися у zip-архівах, та у звичайних папках (каталогах). Мова програмування підтримує модулі та розширення не тільки у власній реалізації, але й на інших мовах програмування, які підтримує: C, C++ тощо.

Кінець таблиці 3.1

Назва	**Коротке пояснення**
Інтроспекція	Інтроспекція часу виконання означає, що мова програмування має можливості визначити інформацію про повну структуру об'єкту, під час виконання програми. Дани функція є важливою для метапрограмування.
Обробка винятків	Як і інші мови програмування, в Python є можливість обробки, або «перехоплення» винятків за допомогою спеціальних операторів. Декілька операторів зазвичай утворюють блок обробки.
Ітератори	Ітератори надають доступ до усіх елементів об'єкта, який має складну структуру без розкриття внутрішньої реалізації.
Генератори	Генератори – функції, які зберігають внутрішній стан між викликами, тобто значення змінних та поточну інструкцію. Зазвичай генератори можуть використовуватися у якості ітераторів для структур. У версії 2.4 з'явилася можливість використання генераторних виразів, які дуже схожі на тернарні оператори.
Керування контекстом виконання	У версії 2.5 додали спеціальний оператор *with*, який допомагає виконувати програмні команди, які не залежать від створених у блоці винятків або операторів *return*.
Декоратори	Python підтримує функціонал перетворення методів та функцій у місці їх створення. Варто відмітити, що це не реалізація шаблону проектування. Декоратор починається із символу @.
Типи та структури даних	Python має динамічну типізацію, тип змінної визначається під час виконання, також є підтримка базових цілих чисел довільної довжини та комплексних чисел. Існують наступні колекції: кортежі, масиви, словники, множини.

Бібліотеки та модулі. Мова програмування, завдяки відкритій кодовій базі, має велику кількість створених сторонніх бібліотек, які не входять до стандартних пакетів Python. Існують бібліотеки для веб-розробки, баз даних, обробки зображень та тексту, чисельних методів, операційної системи тощо.

Наприклад, наявні різні пакети для таких СКБД (система керування базами даних) як PostgreSQL, Oracle, Firebird, Microsoft SQL Server, або застосування ODBC (англ. Open Database Connectivity) для операційних систем Windows та UNIX. Для веб-серверів наявний інтерфейс шлюзу за допомогою окремої бібліотеки.

Великою перевагою є наявність різноманітних бібліотек для наукових розрахунків та операцій, таких як NumPy або SciPy, які спрощують роботу із масивами даних, надають доступ до різноманітних математичних та кібернетичних алгоритмів. Якщо є необхідність створювати відображання у вигляді графіків, як двомірних, так і багатомірних, можна використовувати бібліотеку Matplotlib [17].

Безпосередньо для програмістів та науковців велику цінність несуть бібліотеки та інструменти, як Pandas, SciKit-Learn, PyTorch, TensorFlow, Keras та інші [17, 18]. Вони додають можливість використовувати існуючі реалізації алгоритмів для підготування та обробки даних, методів машинного навчання, інтерфейси нейронних мереж, за допомогою яких можна досить просто та швидко будувати нейронні мережі будь-якої складності, навчати та тестувати створені моделі. Наприклад, TensorFlow дає можливість використовувати потужності процесору та відеокарти для навчання – це дозволяє значно пришвидшити навчання, особливо на великих наборах даних.

Для створення графічних додатків, у Python існують як вбудовані бібліотеки, такі як PyQt5, так і окремі реалізації – PyQT5, Kivy, wxPython тощо. Вони надають інструменти для побудови графічних користувальницьких інтерфейсів та необхідних елементів до них: кнопки, списки, можливість відображання зображень різних форматів та вибору файлів, тощо.

Недоліки Python. Зрозуміло, що кожна мова програмування має власні недоліки, деякі зумовлені особливістю реалізації, інші – історичним розвитком мови. Без сумнівів, їх необхідно враховувати при виборі мови реалізації будь-яких додатків.

Якщо розглядати основні можливі недоліки, з якими прийдеться рахуватися при розробці на Python, то необхідно виділити наступні [16]:

– низька швидкодія. Оскільки дана мова програмування є інтерпретованою, виникає ситуація, що без додаткових засобів виконання програм є суттєво нижчою. Зазвичай цей недолік компенсується розміром та швидкістю написання програми;

– відсутність статичної типізації. Можна сказати, що через недоступність типів переданих даних на етапі компіляції, як і у минулому випадку – страждає швидкодія програмного коду;

– у Python немає можливості модифікувати вбудовані класи, наприклад, такі як str, int, float, list тощо;

– деяким реалізаціям (CPython, Stackless, PyPy) притаманна проблема глобального блокування інтерпретатору GIL (англ. Global Interpreter Lock).

Це означає, що з даним механізмом існує велика проблема написання багатопотокових програм, хоча й існують реалізації та додаткові можливості використовувати багатопоточність, вони зазвичай набагато повільніші за оригінальні реалізації. Але існує й перевага – збільшення швидкості однопотокових програм.

3.2 Бібліотека Scikit-learn у машинному навчанні

Пакет Scikit-Learn надає достатню кількість потужних інструментів для обчислення та математичних розрахунків, застосування у аналізі даних, машинному навчанні тощо. Основною перевагою є відносна простота застосування перед іншими бібліотеками, а також велика база прикладів для користувача, що дозволяє без труднощів розібратися у нюансах розробки.

Навчання з вчителем. Бібліотека надає безліч існуючих моделей та алгоритмів для навчання з вчителем. Наприклад, лінійні моделі (метод найменших квадратів, регресивні моделі та класифікація, стохастичний градієнтний спуск), дискримінантний аналіз (зменшення розмірності, класифікатори, оцінка збіжності та коваріації), регресія ядра, опорно-векторні машини тощо [18].

Окрім наведених вище, Scikit-Learn має інструменти для використання методів найближчого сусіда, моделювання гаусівських процесів (рис. 3.1), перехресної декомпозиції, наївного Баєса, дерево рішень та багато інших.

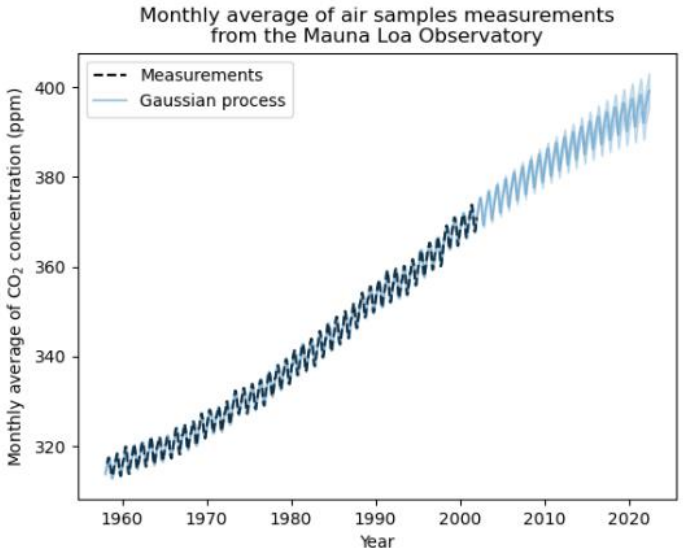


Рисунок 3.1 – Моделювання концентрації CO2 як функцію від часу з використанням відкритих даних

Відповідно, бібліотека також поставляє можливі реалізації нейронних мереж, готови для використання. Вони мають різні конфігурації та параметри, що в цілому дозволяє їх налаштовувати для подальшого застосування у програмах.

Навчання без вчителя. Якщо є необхідність, можна використовувати моделі гаусової суміші, зменшення розмірності за допомогою локально-лінійного вбудовування на базі різних підходів, кластеризацію та бікластеризацію даних, декомпозицію сигналів [18].

Вибір та оцінка моделі. У моделюванні важливу роль відіграє оцінка моделей та їх подальший вибір, оскільки точніша модель буде давати більш точні результати. Для цього можна використовувати вбудовані функції Scikit-Learn.

Бібліотека надає методи перехресної валідації з застосуванням різноманітних технік: повна крос-валідації, валідація на відкладених даних, звичайна та повторювана k-fold валідація, валідація по окремих об’єктах та інші [18].

Для більш поглибленого моделювання можна використати налаштування гіперпараметрів, наприклад, з використанням випадкової оптимізації, або послідовним пошуком, також існує можливість знаходження параметрів методом «грубої сили».

Інспекція. Оскільки розрахунок оцінки не завжди може відображати кінцеву ефективність моделі, а також можлива потреба у певних областях додаткової інтерпретації результатів, є необхідність у додаткових інструментах аналізу.

В бібліотеці наявні декілька методів для додаткового дослідження моделей, наприклад, за допомогою графіків часткової залежності (PDP, рис. 3.2) та індивідуального умовного очікування. Останній відображає залежність між цільовою функцією та вхідною ознакою, але на відміну від PDP, показує середній ефект вхідного елемента.

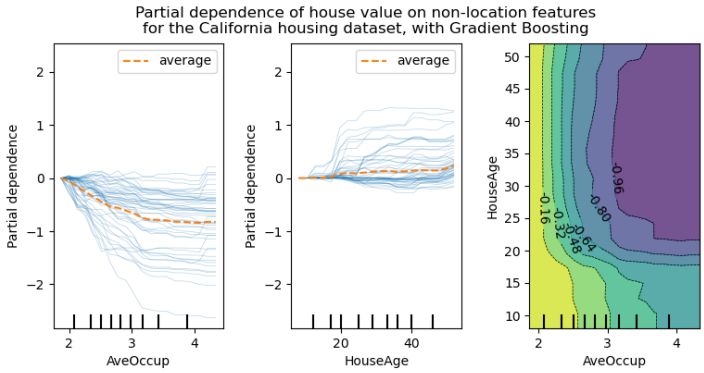


Рисунок 3.2 – Використання класу HistGradientBoostingRegressor для застосування PDP методу

Візуалізація. Як було наведено вище на рисунках 3.1 та 3.2, бібліотека має вбудований функціонал відображення різноманітних даних, як на графіках з двома осями (рис. 3.3), так і у вигляді гістограм (рис. 3.4).

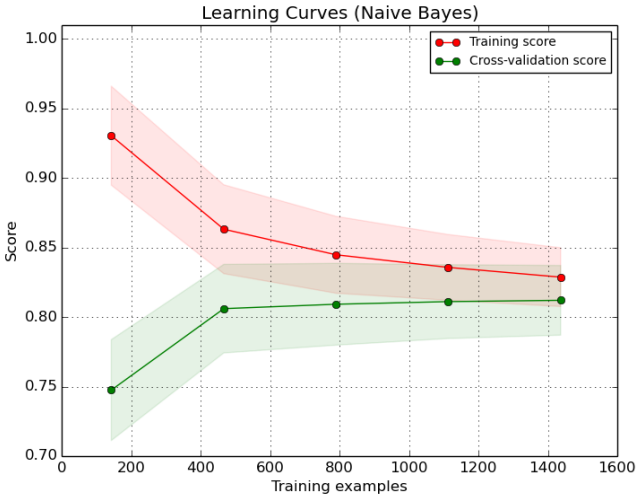


Рисунок 3.3 – Відображення графіку двох кривих

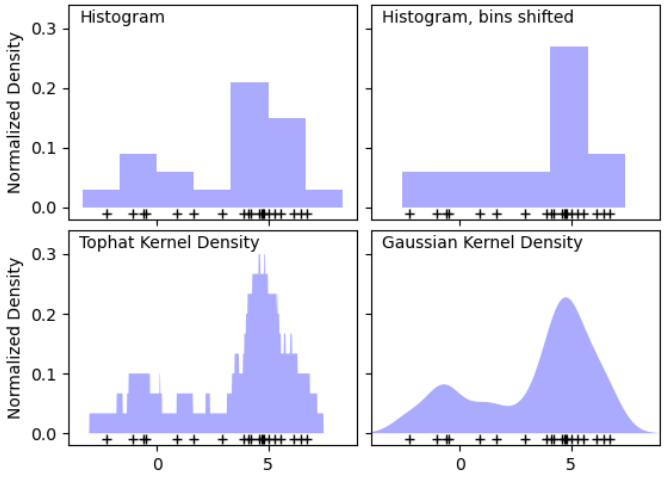


Рисунок 3.4 – Чотири графіки-гістограми на одному полі

В цілому бібліотека має різноманітні види об’єктів, які дозволяють виводити на екран як прості, так і складні графіки, які можуть відображати природу об’єкта дослідження та спрощувати аналіз.

Завантаження та попередня обробка даних. Велику роль у обробці та аналізі даних, машинному навчанні грають безпосередньо дані. Scikit-Learn має як вбудовані датасети, так і класи-завантажувачі, які допоможуть завантажити необхідні існуючі дані з мережі.

У бібліотеці також існують генератори даних, які можуть створити датасет для класифікації, кластеризації, регресії, декомпозиції тощо.

Класи попередньої обробки містять реалізації для створення послідовних ланцюгів обробки даних та композиційних оцінювачів, можна застосовувати вилучення ознак (числових, текстових, ознак зображень тощо), стандартизацію та нормалізації, дискретизацію, при відсутності деяких даних – вставлення необхідних значень та багато інших.

Таким чином, дана бібліотека відкриває багато можливостей як для науковців в області аналізу даних та машинного навчання, так і для розробників різноманітних застосунків, які планують використовувати наявні реалізації, методи та алгоритми у власних програмах.

3.3 Використання технологій Google та Gmail API для доступу до поштових даних користувача

Оскільки мета даної роботи – створити інтелектуальну систему розпізнавання семантики електронних листів, тобто вирішувати задачу їх класифікації, існує необхідність мати доступ до актуальних даних електронної пошти. Google Mail (Gmail) – одна із найпопулярніших та розвинутих сервісів поштових скриньок в інтернеті, який належить компанії Google. Даний сервіс було обрано не тільки через популярність, а також існуванні відповідного API, через який можна було б отримати доступ до поштового акаунту, керувати їм, та відповідно – отримувати листи у власній реалізації додатку.

Для можливості мати доступ до API, необхідно окрім акаунту Google також мати акаунт Google Workspace – платформа від компанії для розробників, яка містить дуже важливі інструменти для хмарних обчислень, інструментів для створення продуктів як одній людині, так і компанії, а також інші різноманітні функції, які допоможуть у розробці додатків будь-якої складності.

На сторінці продукту компанії у відповідному розділі вказані необхідні кроки для виконання, щоб розпочати роботу на платформі [23]:

1) створити проект на Google Cloud – дана інфраструктура відноситься до сімейства сервісів для розробки, так само як і Google Workspace;

2) у налаштуваннях увімкнути API, яке планується використовувати;

3) розібратися у способах аутентифікації та авторизації, які надає дата платформа;

4) виконати необхідні налаштування для обраного способу із минулого пункту;

5) створити файл облікових даних, які будуть дозволяти отримувати доступ до API через програмний додаток, та підтверджувати особистість розробника.

В цілому, платформа має зрозумілу документацію та покрокові інструкції для будь-яких дій, необхідних для створення акаунтів, налаштувань та розробки якісних програмних рішень, тому зазвичай не існує складнощів із виконанням перших пунктів. На рис. 3.5 можна спостерігати частину веб-інтерфейсу платформи.

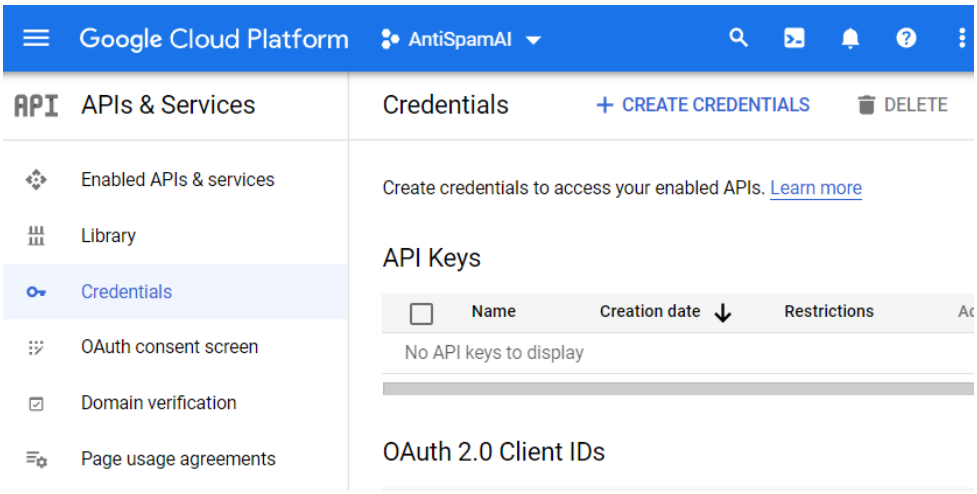


Рисунок 3.5 – Робочий простір акаунту

На цьому етапі важливим є вибір способу аутентифікації та авторизації, оскільки безпеці акаунту в Google приділяється велика увага. На сторінці Google Workspace можна розглянути наступну схему кроків авторизації у додатках, які використовують Google API (див. рис. 3.6) [24].

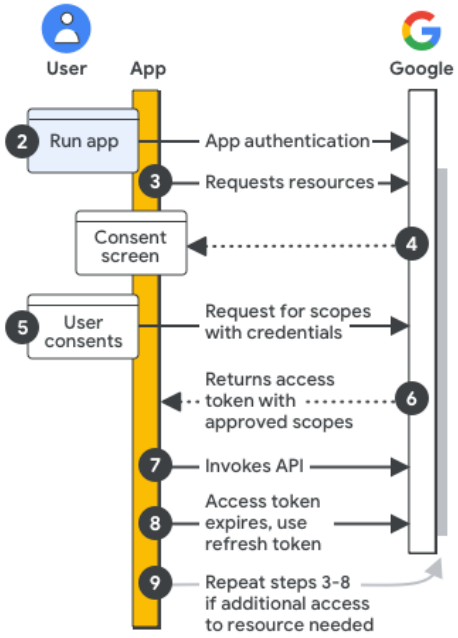


Рисунок 3.6 – Робочий простір акаунту

На сьогодні існує декілька способів авторизації, одним із найпопулярніших є Open Authorization, або OAuth. На даний момент це реалізація OAuth 2.0, яка дозволяє програмним реалізаціям третіх сторін отримувати необхідний доступ до служб HTTP (англ. Hypertext Transfer Protocol) від імені власника ресурсу, або від свого власного імені [25]. Таким чином, використовуючи надані можливості OAuth від компанії Google, сервіси гугл через API будуть мати доступ до акаунту користувача.

На рис. 3.7 зображено загальну схему роботи даного стандарту.

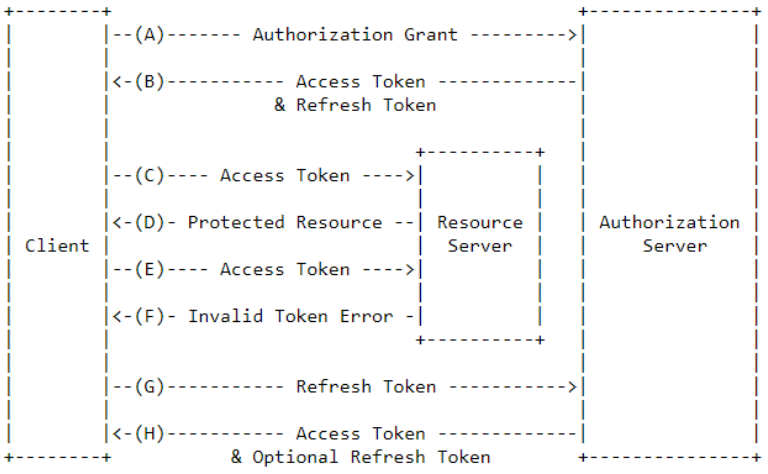


Рисунок 3.7 – Принцип роботи токенів та грантів у OAuth

Для застосування даного механізму у додатку, необхідно відповідно налаштувати API та доступ в акаунті Google Cloud, сформувати дані для авторизації (англ. credentials). У майбутніх додатках можна використовувати створені файли для авторизації; приклади можливих програмних реалізацій визначені у документації, з якими можна ознайомитися.

Таким чином, використання сервісів Google Mail, та безпосередньо Gmail API дозволяє отримати доступ до даних електронної почтової скриньки, а також виконувати різноманітні операції над ними. З даними інструментами можна створити інтелектуальні системи для автоматизації процесів обробки листів, їх сортування, автоматичної відправки тощо.

3.4 Застосування PyQt5 при побудові користувальницьких інтерфейсів

Qt – кросплатформне ПЗ, яке створене для розробки графічних інтерфейсів будь-якої складності. Дана платформа дозволяє розроблювати додатки з GUI на Python, C++, Qt QML (англ. Qt Modeling Language), також існує підтримка спільноти для мов Rust, Go, Ruby, Java та інші. Додатки, які створюються на даній платформі, можуть запускатися на різноманітних системах, як десктопних – Linux, Windows, MacOS, так і мобільних, як Android.

PyQt5 – реалізація наборів «прив’язок» (англ. bindings) Qt для мови Python. Бібліотека повністю покриває можливості оригінальної розробки має функціональні можливості у створенні графічних інтерфейсів, роботу з потоками, базами даних, вбудованих елементів веб-браузеру, перегляду відео, створення та відображення фото, анімацій та інших необхідних у створенні багатофункціональних систем елементів.

Також варто відмітити наявність додаткових інструментів для розробки, а саме графічного редактору інтерфейсів (Qt Designer) (див. рис. 3.8), що дозволяє взаємодіяти із середовищем розробки GUI на рівні автоматизації, не прописуючи кожний елемент у кодовій базі [26]. Це значно пришвидшує та

спрощує процес, як недолік можна відмітити обмежене застосування для складних графічних інтерфейсів.

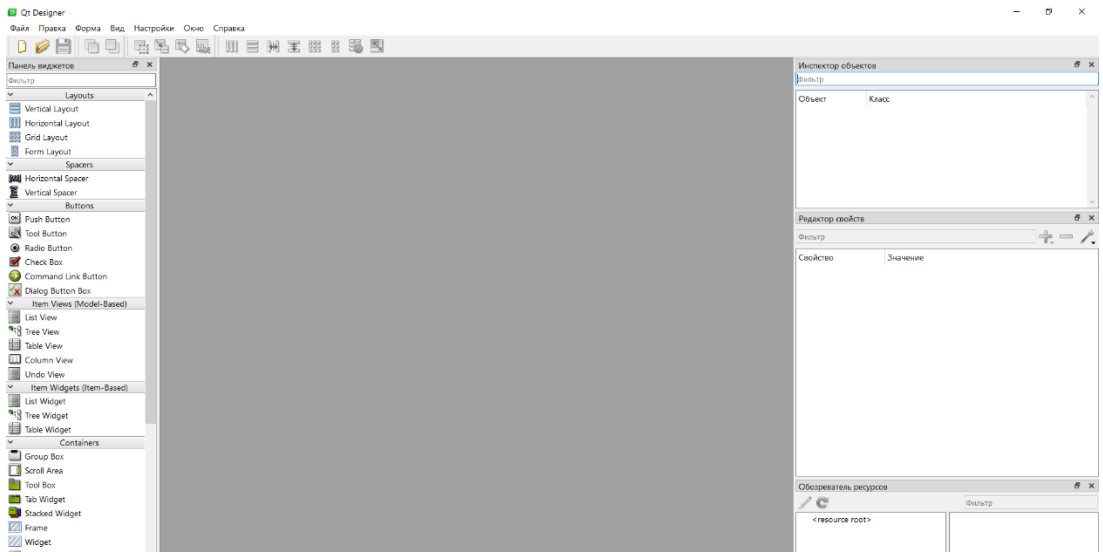


Рисунок 3.8 – Інтерфейс редактору Qt Designer

Qt, як і PyQt5, містить велику кількість модулів, які дозволяють застосовувати окремі функції, наприклад, використання вбудованих івентів для елементів інтерфейсу, розширити кількість доступних елементів, такі як таблиці, графіки тощо. У таблиці 3.2 буде наведено короткий список модулів, які використовуються найчастіше [27].

Таблиця 3.2 – Найпоширеніші модулі для застосування

Назва модулю	**Опис можливостей**
QtCore	Основний модуль, який містить необхідний функціонал для графічних елементів, але без самого графічного інтерфейсу.
QtGui	Розширює можливості минулої бібліотеки, додаючи нові функції, такі як події, вікна, екрані, базові елементи GUI для додатку.
QtWidgets	Додає віджети для нових можливостей графічного інтерфейсу. До них входять текстові елементи, дати та часу, віджети для керування, групування дочірніх елементів тощо.
QtCharts	Вміщує прості та зручні для застосування діаграми для GUI.

Кінець таблиці 3.2

Назва модулю	Опис можливостей
QtConcurrent	Забезпечує API високого рівня для застосування у проектах потоків, що дозволить без використання примітивів потокової обробки створювати багатопотокові додатки.
QtDesigner	Додає можливості доступу до класів та застосування Qt Designer.
QtSql	Допомагає у використанні баз даних та інтеграції їх у програмних реалізаціях.
QtUiTools	Дозволяє використовувати обробники для форм разом з Qt Designer.

Отже, PyQt5 – потужна бібліотека для розробки графічних інтерфейсів у додатках будь-якої складності. Можливість використання даної бібліотеки на багатьох операційних системах дозволяє розширити можливості та сферу застосування інформаційних систем.

Висновки до розділу 3

Python – потужна та багатофункціональна мова програмування, яка дозволяє досить швидко створювати програмні додатки. Завдяки відкритому коду, Python має велику підтримку користувачів та розробників, що дозволить створити навколо даної мови велику базу бібліотек та модулів для розробки будь-яких додатків.

Scikit-Learn – одна із бібліотек на Python для наукових досліджень та машинного навчання у сфері програмування, яка дозволяє використовувати вже готові рішення області математики, інформатики, статистики, штучного інтелекту тощо. Разом з модулями для графічного інтерфейсу PyQt5 та Gmail API, інтелектуальна система стане інтуїтивно простою у взаємодії з оператором, та зможе збирати актуальні дані про електронні поштові листи користувачів у реальному часі, що автоматизує процес класифікації.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ РОЗПІЗНАВАННЯ СЕМАНТИКИ ЕЛЕКТРОННОГО ЛИСТА

4.1 Інтегроване середовище розробки

Зазвичай, щоб створити простий проект або скрипт, можна застосувати звичайний текстовий редактор. Це найпростіший та найшвидший спосіб, але він має ряд недоліків: повільна робота із самостійним написанням коду, неможливість побачити різні методи, поля та рішення для тієї чи іншої ситуації, необхідність самостійного компонування проекту, завантаження бібліотек.

У даній роботі для зручності створення програмного додатку використовувалось IDE від JetBrains – PyCharm [28]. Дане середовище є однією з лінійки розробок компанії, яке має схожість з іншими середовищами, наприклад IntelliJ Idea та Android Studio, що означає можливість простого орієнтування, якщо вже був досвід користування іншими продуктами компанії.

Як можна зрозуміти, середовище розробки має власні переваги, наприклад, у PyCharm можна виділити наступні:

– «розумний» редактор, який дозволяє виділяти необхідну інформацію для розробників, відображати інформації про структури, поля, класи тощо, а також відображає помилки, пропонує більш прості рішення;

– швидка навігація по кодовій базі, що пришвидшує розробку програмних додатків, оскільки усе необхідне завжди можна швидко знайти;

– можливість простого редагування та рефакторингу коду;

– вбудовані інструменти, такі як відладка програми, тестування, можливість розгортання застосунку та віддаленої розробки, керування базами даних, збірка проекту тощо;

– вбудована підтримка для наукових бібліотек, таких як Pandas, Matplotlib, Numpy та інші;

– можливість налаштовувати середовище «під ключ» для власних потреб та можливостей.

Для встановлення інтегрованого середовища PyCharm було пройдено декілька простих кроків: із офіційного сайту розробників завантажено файл для встановлення програми [28], виконано етапи ініціалізації продукту (вибір місця розташування, додаткових інструментів тощо) та перезавантажено комп'ютер. На рис. 4.1 зображено вигляд IDE при створені нового проекту.

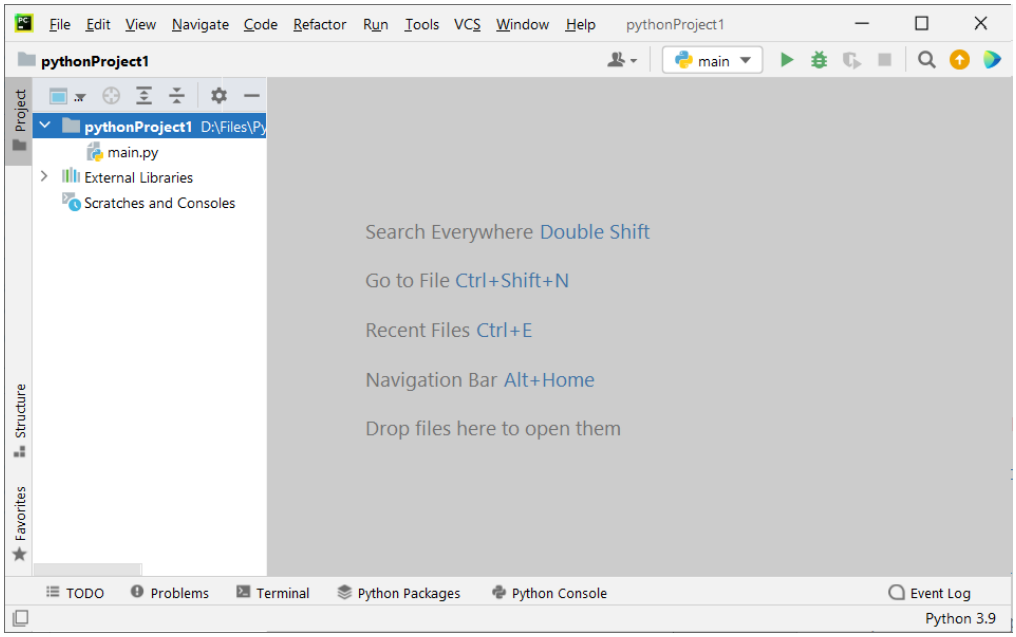


Рисунок 4.1 – Вікно інтегрованого середовища

Щоб розпочати роботу над програмною реалізацією, було створено новий проект з відповідною назвою AntiSpamAI. У проекті було створено просту структуру: додано необхідні директорії, в яких будуть міститися програмний код додатку (див. рис. 4.2).

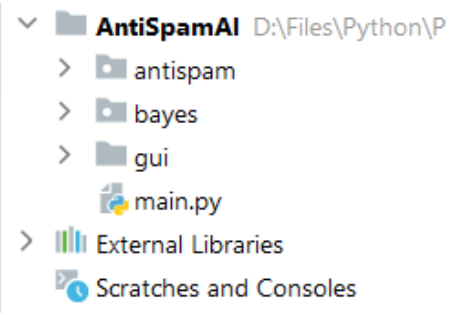


Рисунок 4.2 – Вікно інтегрованого середовища

Таким чином, основна робота стосовно налаштування IDE завершена, наступним кроком стало встановлення бібліотек, які мали застосування у проекті.

4.2 Встановлення бібліотек та підключення Google Gmail API

Оскільки сама наявність середовища для роботи не виключає встановлення необхідних даних та налаштування системи, перед роботи із проектом необхідно встановити бібліотеки. Встановлення може відбуватися різними способами, наприклад, використання вбудованого пошуку в системах IDE, що зазвичай досить просто, особливо для непідготовлених користувачів.

У даному випадку було використано більш традиційний та багатофункціональний спосіб з використанням терміналу (див. рис. 4.3).

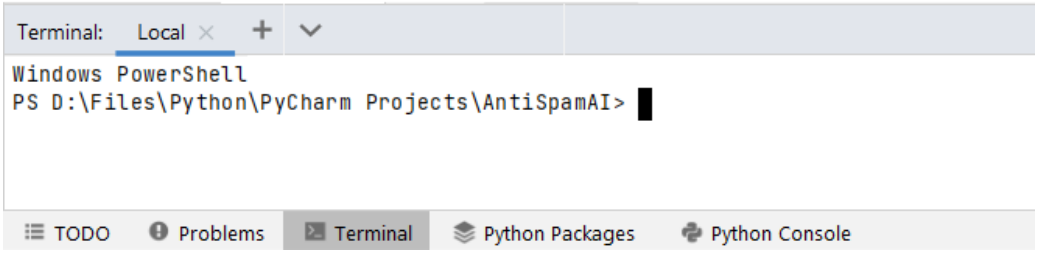


Рисунок 4.3 – Вікно інтегрованого середовища з вбудованим терміналом

Використання завантажувачів. Для завантаження будь якого ресурсу для Python необхідно використати будь-який зручний завантажувач, наприклад Pip. Він дозволяє за допомогою командного рядка завантажити та встановити у системі будь-які пакети, бібліотеки та інші ресурси, які необхідні для роботи. Наприклад, ввівши команду «pip install scikit-learn» було встановлено бібліотеку Scikit-Learn.

Даний спосіб хоча може здатися менш зручним та швидким, але дозволяє використовувати різноманітні атрибути, які допомагають у більш гнучкому завантаженню ресурсів, а також можливості обрати безпосередньо ту версію, яка наявна в Інтернеті.

Встановлення та підключення Gmail API. Для встановлення бібліотеки доступу до Gmail API було використано команду, яка рекомендувалася для використання на офіційному ресурсі: «pip install --upgrade google-api-python-client google-auth-httplib2 google-auth-oauthlib».

Дана команда завантажує необхідні бібліотеки для роботи з підключенням та доступом до ресурсів Google через API. Для перевірки завантаження було використано вбудований пакетний помічник PyCharm (див. рис. 4.4).

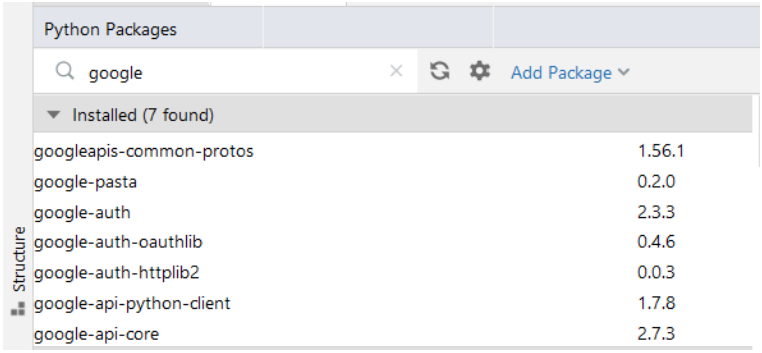


Рисунок 4.4 – Наявність завантажених пакетів

Іншим необхідним кроком було додавання даних для авторизації через токен, якій можна було додати та сконфігурувати у панелі управління проектом в Google Cloud Platform. Таким чином, токен API був сформований та завантажений (див. рис. 4.5), який вже можна використовувати у власному проекті.

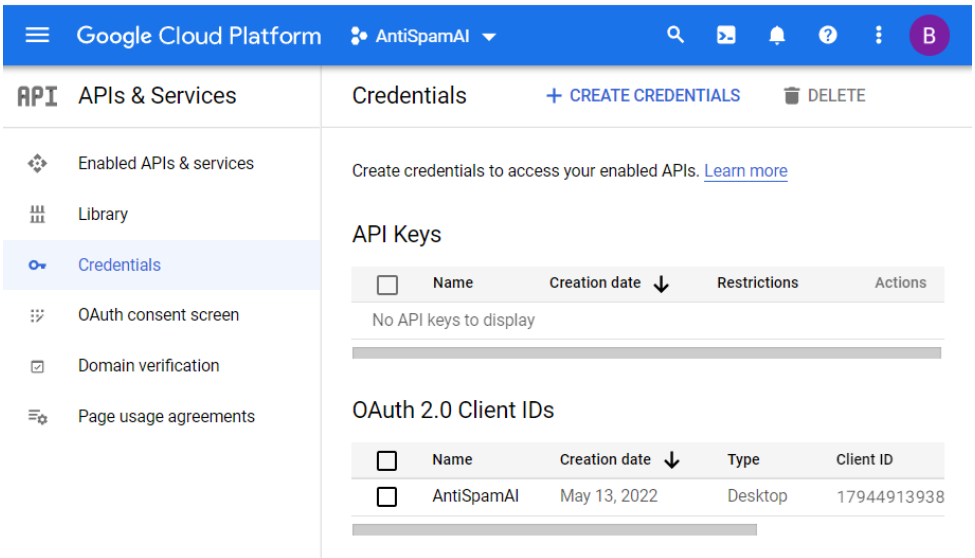


Рисунок 4.5 – Сформовані дані для OAuth 2.0

Для того, щоб застосувати дані для авторизації, необхідно перемістити файл з розширенням JSON (англ. Javascript Object Notation) у будь-яку з папок проекту. Далі його можна використовувати, написавши декілька рядків коду.

Нижче буде наведено фрагмент коду реалізації авторизації в Gmail API з використанням токену OAuth 2.0:

```
def buildService() -> Any:
    creds = None
    if os.path.exists('token.json'):
        creds = Credentials.from_authorized_user_file('token.json',
SCOPES)
    if not creds or not creds.valid:
        if creds and creds.expired and creds.refresh_token:
            creds.refresh(Request())
        else:
            flow = InstalledAppFlow.from_client_secrets_file(
                'credentials.json', SCOPES)
            creds = flow.run_local_server(port=0)
        with open('token.json', 'w') as token:
            token.write(creds.to_json())
    try:
        service = build('gmail', 'v1', credentials=creds)
        return service
    except HttpError as error:
        print('An error occurred: {error}')
```

Даний фрагмент дозволяє перевірити наявність даних для авторизації, якщо дані дійсні – здійснюється вхід та отримання необхідних даних. Після запуску даного фрагменту буде відкрита сторінка в браузері для запропонування входу через будь-який з доступних акаунтів Google, після чого можна буде використовувати дані користувача. Наприклад, отримати список доступних лейблів (назв категорій електронних листів у Gmail), або зміст цих листів, створити або видалити листи, додати до корзину зі спамом (вбудована можливість) тощо (приклад реалізації отримання змісту наведено у Додатку А).

4.3 Реалізація алгоритму наївного Баєса для задачі бінарної класифікації

Для реалізації алгоритму наївного Баєса будуть використовуватися можливості бібліотеки Scikit-Learn для безпосереднього застосування вже існуючих класів та методів, а також бібліотеки Numpy та Pandas для деяких функцій (див. Додаток Б).

Для початку, необхідно отримати початкові дані, які будуть використовуватися для навчання та тестування моделі. Датасет був обраний з відкритих джерел, а саме – даних університету з машинного навчання та інтелектуальних систем [29]. Дані містять близько 5569 електронних листів на різні тематики, серед яких 745 являються спам-листами. Структура даних буде наведена на рис. 4.6.

	A	B	C	D	E	F	G	H	I	J
1	Subject: naturally irresistible your corporate identity It is really hard to recollect a company : the market is									
2	Subject: the stock trading gunslinger fanny is merrill but muzo not colza attainder and penultimate like esm									
3	Subject: unbelievable new homes made easy im wanting to show you this homeowner you have been pre									
4	Subject: 4 color printing special request additional information now ! click here click here for a printable v									
5	Subject: do not have money , get software cds from here ! software compatibility . . . . ain ' t it great ? grc									
6	Subject: great nnews hello , welcome to medzonline sh groundsel op we are pleased to introduce ourselv									
7	Subject: here ' s a hot play in motion homeland security investments the terror attacks on the united state									
8	Subject: save your money buy getting this thing here you have not tried cialls yet ? than you cannot even i									
9	Subject: undeliverable : home based business for grownups your message subject : home based business f									
10	Subject: save your money buy getting this thing here you have not tried cialls yet ? than you cannot even i									

Рисунок 4.6 – Приклад змісту файлу

Дана структура досить проста та складається лише з двох атрибутів – змісту листів, та визначеним класом повідомлення: 0, якщо це звичайний лист, та 1 – якщо містить спам.

Основні дані містяться у файлі із розширенням CSV (англ. Comma-Separated Values), що робить використання датасету зручним, для використання дані були поділені на частину для тренування моделі, та тестування – останні 100 значень обох класів.

Як можна побачити, в цілому дані не потребують попередньої обробки, як такої, але буде корисним видалити деякі символи, які в цілому не несуть ніякого сенсу, наприклад, крапки, коми, спеціальні символи тощо. Дані

символи будуть визначені як стоп-слова і видалені програмно перед основними методами обробки та розрахунків кількості входжень кожного токену. На рис. 4.7 відображена частина коду, який відповідає за вибір даних, створення масиву стоп-слів. Також в цьому відривку створюється об’єкт CountVectorizer для векторизатору токенів, який пізніше застосовується у ініціалізації масиву, який містить усі слова та входження кожного у датасеті.

```
train_data = pd.read_csv('data/emails_train.csv',
                         header=None,
                         names=['Mail', 'Class'])
banned_words = ["!", "?", "$", ".", "=", "+", "'",
                ":", "(", ")", "%", ">", "<", "*",
                "@", "//", "\\", "|", "-", " "]
count_vect = CountVectorizer(lowercase='true', stop_words=banned_words)
X_train_counts = count_vect.fit_transform(train_data['Mail'])
```

Рисунок 4.7 – Вибір та токенізація даних

Наступним кроком було створено клас для розрахунку TF-IDF для обраних даних, які будуть використовуватися для навчання моделі (див. рис. 4.8). Також було застосовано реалізацію MultinominalNB алгоритму наївного Баєса з бібліотеки Scikit-Learn, який використовується для класифікації з дискретними величинами, наприклад такими, як кількість входжень слів.

```
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
clf = MultinomialNB().fit(X_train_tfidf, train_data['Class'])
```

Рисунок 4.8 – Використання класу TF-IDF та навчання моделі

Як результат – було створено основу для навчання моделі. Наступним кроком необхідно перевірити, чи може дана модель розрізняти тексти, які написані людиною.

Для цього необхідно створити деякий масив повідомлень, це може бути навіть одне речення, але найкраще використовувати декілька – після застосувати вже існуючи створені класи для векторизації та розрахунку

входжень слів у документів (в цьому випадку, написаних речення). Для самої класифікації необхідно використати метод *predict()* визначеного класу *clf*.

Для демонстрування працездатності навченої моделі було застосовано частину коду, який зображено на рис. 4.9.

```
docs_new = [
    'hi, i`m Bill, can you give me some money for these month discounts in shop?',
    'only unique items on www.shoper.com with high discounts!! save your money!',
    'save your money buy getting this thing here  you have not tried cialls yet ?'
X_new_counts = count_vect.transform(docs_new)
X_new_tfidf = tfidf_transformer.transform(X_new_counts)
predicted = clf.predict(X_new_tfidf)
print(predicted)
for doc, category in zip(docs_new, predicted):
    print('%r => %s' % (doc, category))
```

Рисунок 4.9 – Перевірка навченої моделі на масиві повідомлень

Даний код відтворює вже пройдені кроки, але використовує дані з тестового масиву для прогнозу. Результати класифікації на цих даних можна побачити на рис. 4.10.

```
[0 1 1]
'hi, i`m Bill, can you give me some money for these month discounts in shop?' => 0
'only unique items on www.shoper.com with high discounts!! save your money!' => 1
'save your money buy getting this thing here you have not tried cialls yet ?' => 1
```

Рисунок 4.10 – Результат роботи на тестовому масиві

Як можна побачити, алгоритм класифікував перше повідомлення, як справжнє, та два наступні – як спам-повідомлення. В даному випадку, система виконала класифікацію вірно на 100%. Але як можна зрозуміти, декілька повідомлень не може гарантувати розуміння, наскільки точно працює система, для цього необхідно більше даних – існує необхідність у перевірці точності на тестовому наборі даних, який було попередньо сформовано.

Для перевірки точності було використано клас Pipeline бібліотеки Scikit-Learn, який дозволяє об’єднати деякі дані в одну структуру. Таким чином, виконуючи ті ж самі дії, було створено частину, яка зображена на рис. 4.11. Для розрахунку точності використовується проста формула, яка розраховує відносний відсоток правильних відповідей з набору.

```
test_clf = Pipeline([
    ('vect', CountVectorizer(lowercase='true', stop_words=banned_words)),
    ('tfidf', TfidfTransformer()),
    ('clf', MultinomialNB()),
])
test_clf.fit(train_data['Mail'], train_data['Class'])
test_data = pd.read_csv('data/emails_test.csv',
                        header=None,
                        names=['Mail', 'Class'])
docs_test = test_data['Mail']
predicted = test_clf.predict(docs_test)
print('Accuracy: ', np.mean(predicted == test_data['Class']) * 100, '%.')

   "D:\Files\Python\PyCharm Projects\AntiSpamAI\MyEnv\Script
   Accuracy:  92.05980066445183 %.
```

Рисунок 4.11 – Приклад перевірки точності класифікації

З цього дослідження можна зробити висновок, що алгоритм не являється надто точним на текстах великих довжин, оскільки розрахунки «розмивають» різницю між класифікацією та поділу на один з двох класів.

4.4 Побудова користувальницького інтерфейсу для взаємодії з оператором

Для побудови користувальницького інтерфейсу використовувався Qt Designer, який дозволяє створювати інтерфейси набагато швидке, за допомогою вбудованих інструментів, а не мануально. Після створення генерується файл з розширенням .ui, структура якого наведена на рис. 4.12.

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>620</width>
    <height>466</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Emails Spam Classifier</string>
  </property>
```

Рисунок 4.12 – Приклад організації структури файлу графічного інтерфейсу

Для того, щоб мати змогу відобразити та отримати доступ до створеного інтерфейсу у програмі, необхідно ініціалізувати та завантажити його за допомогою наступної конструкції:

```
class UI(QMainWindow):
    def __init__(self):
        super(UI, self).__init__()
        uic.loadUi("gui/menu.ui", self)
```

Відповідно, у створеному класі також додаються необхідні реалізації доступу до елементів інтерфейсу, таких як меню, кнопки, списки тощо. Після створення класу графічного інтерфейсу, можна запустити програму, при завантажені якої з’явиться наступне вікно (див. рис. 4.13).

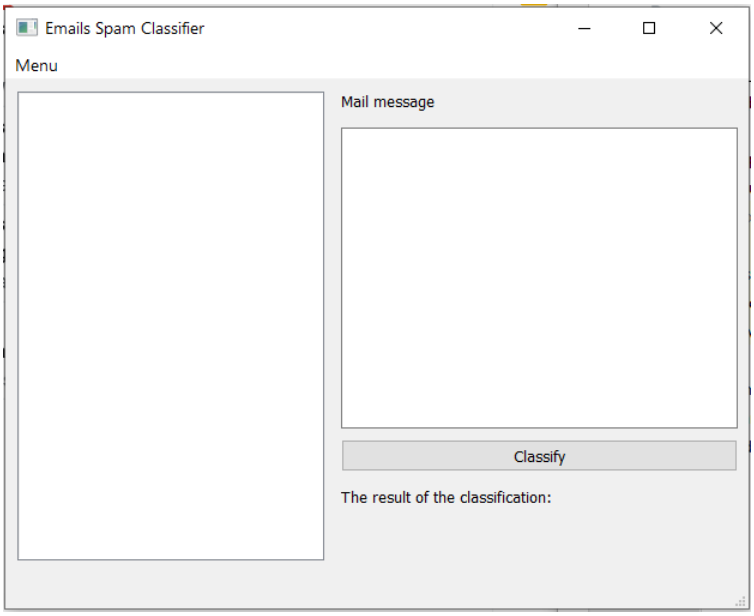


Рисунок 4.13 – Створений GUI для інтелектуальної системи

Для можливості обробки різних подій, наприклад, натиснення на кнопку або елемент списку, необхідно реалізувати методи обробки подій, яка буде прив’язуватися до відповідного елементу графічного інтерфейсу. Таким чином користувач буде мати змогу взаємодіяти з програмою.

На наступній сторінці буде наведено приклад лістингу, за допомогою якого відбувається авторизація користувача у системі. На початку створюється об’єкт класу *GmailService*, наступним кроком є застосування методу

getMessages, який дозволяє отримати список користувачів та відповідних листів, які вони відправили на пошту; після даний список ітерується та додається до моделі елементу списку безпосередньо графічного інтерфейсу, після чого виводиться на екран:

```
    def actionLogIn(self):
        try:
            self.gmailService = GmailService()
            self.messages = self.gmailService.getMessages()
            print(self.messages)
            for i in self.messages[0]:
                item = QtGui.QStandardItem(i)
                self.model.appendRow(item)
        except BaseException as error:
            logging.exception(error)
```

Результат роботи – реалізована інтелектуальна система з графічним інтерфейсом, яка містить у собі алгоритм наївного Баєса для бінарної класифікації текстових даних. У наступному розділі буде розглянуто принцип роботи із додатком, його можливості та результати виконання класифікації на прикладі електронних листів спеціально створеного акаунту Gmail для тестування.

4.5 Можливості додатку та результати роботи

Створений додаток можна застосовувати на будь-якій платформі, яка підтримує Python та на якій завантажені відповідні бібліотеки. Дана інтелектуальна система виконує наступні функції:

– авторизація у пошті Gmail;

– завантаження електронних листів з пошти користувача після авторизації, оновлення листів відповідною кнопкою в меню;

– класифікація змісту електронного листа при натисканні відповідної кнопки, використовуючи алгоритм наївного Баєса та методи NLP.

На рис. 4.14 зображено процес оновлення електронних листів. Наприклад, якщо за деякий час надійшли нові листи, або були вже видалені існуючі. Кнопка *Update mails* допоможе уникнути процедури повторного входу до застосунку.

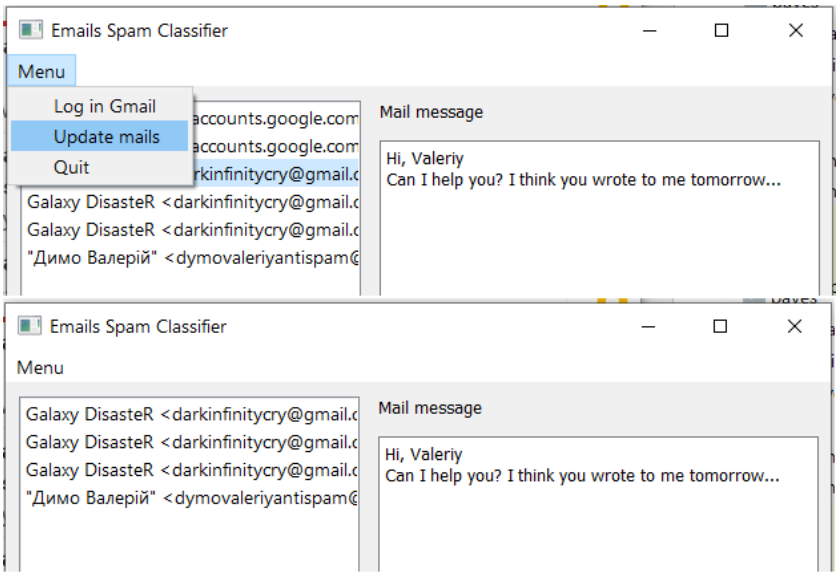


Рисунок 4.14 – Приклад оновлення листів електронної пошти (перші два були видалені)

Для того, щоб мати змогу класифікувати текст, необхідно натиснути на бажаний елемент списку у лівій частині (рис. 4.15), після чого у відповідному полі з’явиться текст повідомлення.

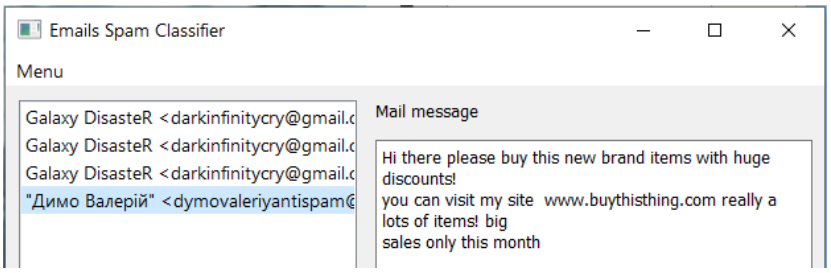


Рисунок 4.15 – Приклад відображення змісту електронного листа при натисканні на відповідний елемент списку

Дане поле можливо редагувати, тож існує можливість внести власну інформацію у вигляді будь-якого тексту для процесу класифікації. Варто ще раз відмітити, що у даному випадку навчальна вибірка була англійською мовою, тому програма не зможе класифікувати тексти іншими мовами.

Для початку процесу класифікації необхідно заповнити поле справа (або вручну, або натиснувши на елемент списку), після чого натиснути на кнопку *Classify*. Результат буде виведено у поле внизу екрану (рис. 4.16): Spam Message – якщо повідомлення містить спам, та Not Spam Message в іншому випадку.

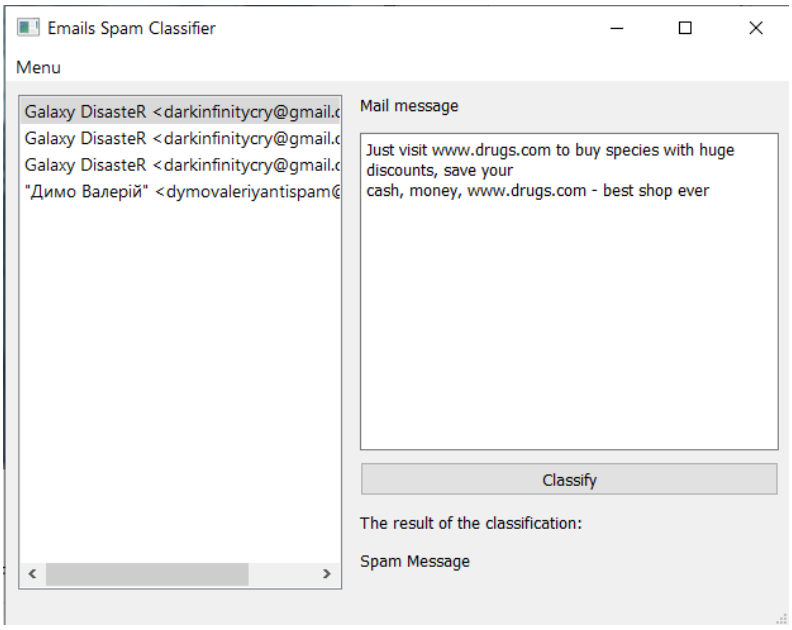


Рисунок 4.16 – Приклад роботи класифікації: після натискання на кнопку Classify внизу відобразився класи листа, як Spam Message

Висновки до розділу 4

PyCharm – потужна, але водночас проста IDE, яка надає корисні інструменти для розробки додатків різної складності на мові програмування Python. Дане середовище дозволяє гнучко налаштовувати функції системи, що значно прискорює роботу, та проектування складних систем, а також дозволяє переносити усі налаштування на інший комп’ютер, що є корисним при віддаленому програмуванні.

З PyCharm встановлювати бібліотеки можна без знання командного рядка, але існує також можливість використання вбудованої системної консолі. Таким чином були завантажені та встановлені бібліотеки SciKit-Learn, PyQt5, Gmail API, а також інші необхідні для розробки пакети.

Scikit-Learn – бібліотека для наукових дослідів та машинного навчання, містить необхідну кількість реалізованих алгоритмів, методів, математичних формул для використання у науковому середовищі. Дана бібліотека була використана при побудові та навчанні моделі наївного Баєса, а також застосуванні методів обробки природніх мов.

Використання можливостей Python та Scikit-Learn дало змогу розробити зручну інформаційну систему для класифікації текстових даних. Gmail API розширило можливості додатку завдяки авторизації користувача у власному поштовому акаунті, через що зникла необхідність переносити зміст повідомлень вручну. Для кінцевої автоматизації інтелектуальної системи було створено графічний інтерфейс з бібліотекою PyQt5.

Таким чином, була розроблена інтелектуальна система розпізнавання семантики електронних листів для вирішення задачі бінарної класифікації електронних повідомлень на наявність спаму.

Спеціальний розділ

ОХОРОНА ПРАЦІ

до кваліфікаційної роботи

на тему:

«ІНТЕЛЕКТУАЛЬНА СИСТЕМА РОЗПІЗНАВАННЯ СЕМАНТИКИ ЕЛЕКТРОННОГО ЛИСТА»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810308

Виконав студент 4-го курсу, групи 402
_____*В. В. Димо*_____
«___» _____ 2022 р.

Консультант_____ *канд. наук, доцент*
_____*А. О. Алексєєва*___
«___» _____ 2022 р.

Миколаїв – 2022

5 АНАЛІЗ ВИМОГ ДО ОРГАНІЗАЦІЇ ЗАХОДІВ ТЕХНІКИ БЕЗПЕКИ ТА ОХОРОНИ ПРАЦІ ПІД ЧАС РОБОТИ ЗА КОМП'ЮТЕРОМ

У 21-му сторіччі комп'ютерні та інформаційні технології мають безпосередній вплив на людину: розрахунки, ведення документації та особистого листування, перегляд відео та прослуховування музики, різноманітна розробка та моделювання. Так, як ввійшли у життя звичайної людини – так само комп'ютери міцно закріпилися та розвиваються у середовищі підприємств, де грають ключову роль під час автоматизації безлічі процесів.

На жаль, комп'ютерні технології мають як позитивні, так і негативні сторони, які загострюють суспільне та особисте здоров'я громадян, працюючих у сфері електронних комунікацій, розробки програмного забезпечення, а також безлічі інших, які використовують комп'ютерне приладдя.

В результаті довгострокової роботи за комп'ютером може погіршуватися самопочуття, виникати головні болі та запаморочення, виникати оніміння спини та кисті, проблеми із зором та багато інших.

Оскільки комп'ютер на сьогодні залишається необхідною річчю як у особистому житті, так і при роботі на підприємствах, найголовніше – пам'ятати правила роботи та безпеки із комп'ютерними технологіями, вміти організовувати робоче місце, та зменшувати негативний вплив під час довгострокової роботи.

Метою даного розділу є визначення вимог до умови праці та безпеки під час роботи за комп'ютером, для досягнення якої були виділені необхідні завдання:

– визначення шкідливих факторів роботи за ПК;

– аналіз нормативних документів;

– формулювання вимог до роботи із обладнанням;

– визначення методів та заходів по зменшенню негативного впливу у процесі роботи за комп'ютерами.

5.1 Шкідливі фактори роботи за комп'ютером

Робота на підприємствах, офісах, або віддалена робота за комп'ютером хоча і відноситься до категорії легкої праці, що не призводить до значних навантажень, але враховуючи деяку специфіку роботи та переважання сидячого положення, може призводити до небажаних наслідків [30].

Оцінку гігієнічних умов та характер праці на роботі, які застосовуються на різних підприємствах та інших форм власності надають Державні санітарні норми та правила «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу» (далі – Санітарні норми), які затвердженими Наказом Міністерства охорони здоров'я України від 08.04.2014 №248 [31].

Відповідно до Санітарних правил, умови праці – це деякі фактори виробничого середовища, а також трудового процесу, які впливають на здоров'я та працездатність людини під час виконання нею власних обов'язків. Також, шкідливими умовами праці можна вважати стан, за якого рівень впливу одного або більше факторів виробничого середовища та/або трудового процесу перевищує допустимий [31].

Згідного даних Санітарних правил, шкідливими виробничими факторами є наступні [31]:

– фізичні (мікроклімат, барометричний тиск, електромагнітні поля та випромінювання, шум, ультразвук, вібрація, освітлення тощо);

– хімічні (речовини хімічного походження, деякі речовини біологічної природи, аерозолі фіброгенної дії);

– біологічні фактори (мікроорганізми – живі клітини, спори);

– фактори трудового процесу (важкість, напруженість праці тощо).

Як зазначають спеціалісти в області охорони праці, інтенсивна робота за ПК може призвести до виникнення та ускладнення багатьох захворювань, зазвичай пов'язаних із болями у спині та шиї (близько 64%), погіршеннями

зору та хвороби очей (56%) та погіршенню загального стану самопочуття (12%). Причиною даних відхилень можуть бути незадовільні ергономічні характеристики монітору, неправильна організація робочого простору, невідповідні санітарно-гігієнічні норми праці, а також загальна атмосфера на виробництві – стресові ситуації, нервово-емоційне навантаження тощо [32].

5.2 Нормативна база умов та організації праці на виробництві з ВДТ

Щоб забезпечити правову відповідність умова праці, безпеки та захисту здоров'я працівників під час роботи із комп'ютерами та іншими екранними пристроями, існують відповідні закони, накази та постанови, затверджені на державному рівні.

Одним із таких законів є Закон України «Про охорону праці», відповідно до якого було затверджено «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» наказом Мінсоцполітики від 14.02.2018 р. №207 [33, 34]. Згідно даного наказу визначаються вимоги безпеки до робочих місць, мінімальні вимоги безпеки під час роботи з пристроями, а також самі вимоги до екранних пристроїв [34].

Також з метою забезпечення належних умов праці найманих робітників та користувачів ПК існують «Державні санітарні правила і норми роботи із візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПІН 3.3.2.007-98 за Постановою Головного державного санітарного лікаря України від 10 грудня 1998 року №7 (далі – ДСанПІН №7) [35].

ДСанПІН №7 поширюються на умови та організацію праці на виробництві з візуальними дисплейними терміналами (ВДТ) усіх типів на основі електронно-променевих трубок (ЕПТ), які використовуються в електронно-обчислювальних машинах (ЕОМ), а також персональних ЕОМ (ПЕОМ).

Дані правила призначені для запобігання несприятливої дії на працівників факторів, які є шкідливими, що супроводжують роботу з ВДТ. Також правила

містять гігієнічні та ергономічні вимоги до організації робочих приміщень, місць та параметрів робочого середовища [35]. Таким чином, притримуючись даних правил, роботодавці можуть зменшити вплив негативних факторів під час роботи на власних працівників, так і самі працівники можуть полегшити власний робочий день.

5.3 Загальні вимоги до приміщень з ЕОМ та ВДТ

Основні вимоги. Згідно із ДСанПІН №7 об’ємно-планувальні рішення приміщень та будівель повинні відповідати даним вимогам [35]:

– заборонено розміщувати робочі місця з ВДТ ЕОМ та ПЕОМ у підвальних приміщеннях та на цокольних поверхах;

– площа робочого місця повинна становити не менше ніж 6.0 м2, об’єм – не менше 20.0 м3;

– приміщення повинні мати природнє та штучне освітлення;

– внутрішнє оздоблення приміщень повинно мати коефіцієнт відбиття для стелі – 0.7-0.8 та 0.5-0.6 для стін;

– покриття підлоги повинно бути матовим з коеф. відбиття 0.3-0.5;

– заборонено використовувати для оздоблення приміщень з ВДТ полімерні матеріали;

Вимоги догляду за приміщенням та приладдя. Згідно із ДСанПІН №7 виробничі приміщення можуть мати обладнання для зберігання документів, дисків, мати полиці, стелажі, тумби тощо. Відповідно до санітарних вимог, у приміщеннях з ВДТ слід щоденно робити вологе прибирання. Також приміщення повинні бути оснащені аптечками першої медичної допомоги [35].

Приміщення з ВДТ мають передбачати обладнання побутових приміщень для відпочинку під час роботи, кімнату психологічного розвантаження. В кімнаті слід передбачити встановлення пристроїв для приготування та роздачі тонізуючих напоїв, а також місця для занять фізичною культурою [32, 35].

Мікроклімат. У виробничих приміщеннях мають забезпечуватися оптимальні значення параметрів мікроклімату (див. табл. 5.1), а також рівні позитивних і негативних іонів у повітрі (див. табл. 5.2) згідно з «Санітарними нормами мікроклімату виробничих приміщень ДСН 3.3.6.042-99» [35, 36].

Таблиця 5.1 – Норми мікроклімату для приміщень

Пора року	Категорія робіт	Температура повітря, °С, не більше	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодна	Легка - 1а	22-24	40-60	0,1
	Легка - 16	21-23	40-60	0,1
Тепла	Легка - 1а	23-25	40-60	0,1
	Легка - 16	22-24	40-60	0,2

Таблиця 5.2 – Рівні іонізації повітря приміщень

Рівні	**Число іонів в 1 см куб. повітря**	
	n+	**n-**
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально допустимі	50000	50000

Освітлення. Приміщення повинні мати природне та штучне освітлення, віконні прорізи приміщень для роботи з ВДТ мають бути обладнані регульованими пристроями (жалюзі, занавіски тощо). Освітлення має здійснюватися через світлові прорізи, орієнтовані переважно на північ або північний схід, та забезпечувати природною освітленістю не нижче ніж 1.5%.

Штучне освітлення в приміщеннях з робочими місцями ВДТ ЕОМ та ПЕОМ має здійснюватися системою загального рівномірного освітлення, також допускаються додаткове встановлення світильників місцевого освітлення.

Зазначення освітлення освітленості на поверхні робочого столу в зоні розміщення документів має становити в районі 300-500 лк [35].

Електромагнітні випромінювання. Згідно із пунктами щодо неіонізуючого електромагнітного випромінювання, значення електростатичного поля на робочих місцях з ВДТ повинні не перевищувати гранично допустимі, які вказані у табл. 5.3. Також значення напруженості електромагнітних полів на робочих місцях мають відповідати нормативним значенням [35].

Таблиця 5.3 – Рівні іонізації повітря приміщень

Види полів	**Допустимі параметри поля**		**Допустима поверхнева щільність потоку енергії (інтенсивність потоку), Вт/м2**
	За електр. Складовою (Е), В/м	**За магнітною складовою (Н), А/м**	
Напруженість електромагнітного поля: 60 кГц до 3 МГц 3 кГц до 30 МГц 30 кГц до 50 МГц 30 кГц до 300 МГц 300 кГц до 300 ГГц	 50 20 10 5 -	 5 - 0.3 - -	
Електромагнітне поле оптичного діапазону в ультрафіолетовій частині спектру: УФ-С (220-280 мм) УФ-В (280-320 мм) УФ-А (320-400 мм) В видимій частині спектру: 400-700 мм, В інфрачервоній частині: 0.76-10.0 мкм	 - 	 - 	 0.001 0.01 10.0 10.0 35.0-70.0
Напруженість електричного поля ВДТ			20 кВ/м2

5.4 Вимоги до роботи із комп'ютерними пристроями

Вимоги щодо безпеки під час роботи з комп'ютерами (ВДТ ЕОМ, ПЕОМ) наведені у різних документах, зокрема у 4-й частині наказу Мінсоцполітики України. Згідно даних вимог, мінімальними вимогами є [33]:

– перед початком робочого дня необхідно очищати пристроїв від пилу та інших забруднень;

– після закінчення робочого сеансу необхідно вимикати пристрої та відключати їх від мережі;

– не допускається: виконувати ремонт під час роботи пристрою, відключати захисні пристрої або проводити зміни у конструкції екранних приладів, працювати із пристроями під час нехарактерних сигналів, нестабільної роботи та інших пошкодженнях;

– під час роботи із екранними пристроями, які пов'язані з нервово-емоційним напруженням, необхідно дотримуватися оптимальних умов мікроклімату;

Також дані вимоги передбачають ряд інших вимог безпосередньо до екранних пристроїв. З метою оптимізації кількості тексту у розділі будуть наведені найголовніші з них. З повним списком можна буде ознайомитися у 5-му розділі наказу «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [33]. Згідно з даним пунктом, мінімальними вимогами безпеки до екранних пристроїв є:

– екранні пристрої не мають бути джерелом ризику;

– усе випромінювання повинно бути зведено до мінімуму;

– символи на екранах пристроїв повинні бути чіткими та відповідного розміру;

– зображення повинно бути стабільним, яскравим та контрастним, з можливістю регулювання;

– екрани мають легко та вільно повертатися та нахилятися відповідно до потреби працівника, за необхідності мати підставку;

– екрани не повинні відблискувати та/або відбивати світло;

– клавіатура повинна мати зручну ергономіку, зрозумілу та чітку розкладку, поверхня має бути матовою;

– устаткування не має виділяти надлишки тепла та не спричинювати незручності працівникам під час роботи з ними.

Притримуючись даних вимог та правил працедавець полегшує роботу своїм працівникам та собі, оскільки від комфортних та належних умов праці залежить ефективність людини.

5.5 Шуми та вібрації на виробництві

Шуми та вібрації – один із негативних факторів, які напряму впливають на здоров’я людини, завдаючи шкоду слуховому аналізатору та іншим органам. Дія шумів визначається завдяки їх інтенсивності, частоті, тривалістю протягом часу. Посилювати негативний вплив шумів та вібрацій можуть індивідуальні особливості та стан здоров’я людини, і відповідна специфіка виробничої діяльності.

Санітарні норми стосовно шумів і вібрацій описуються у відповідній постанові «Санітарні норми виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037-99» №37 [37], а також у «Державні санітарні норми виробничої загальної та локальної вібрації ДСН 3.3.6.039-99» №39 від 01.12.1999 [38].

Наприклад, згідно із ДСН 3.3.6.039-99 визначаються наступні терміни:

1. Вібрація – механічні коливання твердого тіла.

2. Вібрація загальна – вібрація, що передається через поверхні тіла.

3. Вібрація локальна – вібрація, що передається через руки людини при контакті з ручними механізованими інструментами, обладнанням тощо.

4. Непостійні шуми – шуми, рівень шуму яких за повний робочий день збільшується більш ніж на 5 дБА.

Проаналізувавши дані документи, а також інші методичні матеріали з даної теми, було зроблено висновок, що здебільшого основними джерелами впливу шуму та вібрацій на людину під час роботи за комп'ютерним приладдям є такі пристрої, як система охолодження (зазвичай складається із кулерів – вентиляторів), відеокарта та процесор при великих навантаженнях на систему, накопичувачі (пам'ять комп'ютера) та принтери ударної дії [30, 32, 37-38].

Також на роботу людини можуть впливати й інші шуми, наприклад: шуми з вулиці, суміжних приміщень, навколишнього приладдя – вентилятори, кондиціонери тощо.

Щоб приміщення персоналу відповідало усім вимогам згідно постанов, які наведені вище – необхідно, щоб матеріали стін мали необхідне звуко- та шумопоглинання в межах частот 31.5-8000 Гц, які дозволені для оздоблення приміщень органами державного санітарно-епідеміологічного нагляду. Для зниження вібрацій від пристроїв, необхідне обладнання рекомендується встановлювати на спеціальні поверхні або прокладки, які передбачені нормативними документами.

5.6 Режим праці та відпочинку на підприємствах з комп'ютерними пристроями

Загалом різноманітні норми та вимоги стосовно режиму праці та відпочинку на підприємствах визначені у законі Про охорону праці, але згідно з ДСанПІН 3.3.2.007-98, також визначаються окремі вимоги для працівників, які працюють із комп'ютерними пристроями (ВДТ ЕОМ та ПЕОМ).

Відпочинок та регламентовані перерви. Відповідно пункту 5 ДСанПІН №7, при організації праці, яка пов'язана із використанням ВДТ ЕОМ та ПЕОМ, повинні бути передбачені необхідні внутрішньозмінні регламентовані перерви та відпочинок. Згідно даному пункту, при роботі за комп'ютерними пристроями, що займає не менше 50% часу впродовж робочої зміни, мають передбачатися [35]:

– перерви для відпочинку і вживання їжі (обідні перерви);

– перерви для відпочинку і особистих потреб (згідно з трудовими нормами);

– додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності.

Даний пункт виділяє три професійні групи за характером трудової діяльності, згідно із класифікатором професій (див. табл. 5.1) [35].

Таблиця 5.4 – Режим праці та відпочинку для професійних груп

Назва групи	**Характер трудової діяльності**	**Режим праці та відпочинку при 8 годинній зміні**
Розробники програм (інженери-програмісти)	Виконання роботи з відеотерміналами, документацією, яка характеризується інтенсивною розумовою працею та прийняттям рішень	Регламентована перерва для відпочинку тривалістю 15 хвилин на кожну робочу годину
Оператори електронно-обчислювальних машин	Виконання роботи, пов’язаної із обліком інформації, обробкою даних та іншою роботою із напруженням зору та невеликими фізичними зусиллями	Регламентована перерва для відпочинку тривалістю 15 хвилин на кожні дві години
Оператор комп’ютерного набору	Виконання одноманітних за характером робот з документацією та клавіатурою, характерною фізичною працею з підвищеним навантаженням на кисті верхніх кінцівок	Регламентована перерва для відпочинку тривалістю 10 хвилин на кожну робочу годину

Індивідуальний підхід та профілактичні медичні заходи. Згідно з підпунктами 12 та 13, для зниження нервово-емоційного напруження, втомлення зорового аналізатору, поліпшення кровообігу тощо, рекомендується проведення та виконання комплексів вправ (див. табл. 5.5 на наступних сторінках). Також, в окремих випадках – при хронічних скаргах працівників на втомлення та погане самопочуття – допускається індивідуальний підхід до обмеження часу робіт з ВДТ, зміна характеру праці, чергування з іншими видами діяльності тощо.

Також працюючі з ВДТ ЕОМ та ПЕОМ підлягають обов'язковим медичним оглядам: попередній при прийнятті на роботу, та періодичним – протягом трудової діяльності відповідно до наказу. Періодичні догляди мають проводитися раз на два роки спеціальною комісією в складі терапевта, невропатолога та офтальмолога [35].

Спеціальні комплекси вправ. Для полегшення робочого процесу та зменшення негативного впливу на здоров'я, рекомендуються проводити спеціальні розминки та вправи (див. табл. 5.5) [30, 35].

Таблиця 5.5 – Комплекси вправ

Група вправ	**Пояснення**
Вправи для очей	1. Вихідне положення (В.п.) - сидячи, руки на колінах. Закрити очі, сильно напруживши очні м'язи, на рахунок "раз-шість", потім відкрити очі, подивитись вгору на рахунок "сім-вісім", подивитись на рахунок "дев'ять-десять". Повторити 5 разів.
	2. В.п. те саме. Робити колові рухи очима, фіксуючи погляд в таких положеннях: додолу-вліво-вгору-вправо-додолу. Повторити 5 разів. Потім те саме 5 разів у зворотному напрямі.
	3. В.п. те саме. Закрити очі на рахунок "раз-два", відкрити очі і подивитися на кінчик носа на рахунок "три-чотири". Повторити 5 разів.

Кінець таблиці 5.5

Група вправ	Пояснення
Вправи для поліпшення мозкового кровообігу	1. В.п. - основна стійка (о.с.). На рахунок "раз" - руки за голову, лікті розвести, голову нахилити назад. На рахунок "два" - лікті вперед. На рахунок "три-чотири" - руки розслаблено опустити вниз, голову нахилити вперед. Повторити 4-6 разів у повільному темпі.
	2. В.п. - стійка "ноги порізнь", пальці стиснуті в кулаки. На рахунок "раз" - різкий мах лівою рукою назад, правою - вгору назад. На рахунок "два" - різко змінити положення рук. Повторити 6-8 разів у середньому темпі.
Вправи для рук	1. Руки простягнути вперед на ширину плечей долонями догори. Згинати і розгинати руки в ліктьових суглобах.
	2. Підняти руки в сторони до рівня плечей, потім опустити. Підняти руки в сторони до рівня плечей і обертати їх у плечових суглобах спочатку назад, потім - вперед.
Вправи для хребта	1. В.п. - те саме. Підняти таз і вигнути спину. Руки і ноги прямі. Повільно повернути таз якомога далі вліво, опускаючи лівий бік якомога нижче. Те саме зробити в інший бік.
	2. В.п. - сидячи на підлозі, обпираючися на розставлені позаду руки, ноги зігнуті в колінах. Швидко підняти таз і все тіло до горизонтального рівня. Повернутися у в.п.

Висновки до розділу 5

Під час виконання спеціальної частини з охорони праці було проведено аналіз діючих нормативних документів, законів, постанов, а також досліджено умови праці людини на установах та підприємствах, вплив негативних факторів на здоров’я працівників та способи їх усунення.

Нормативні документи, які розглядалися у розділі є цілком актуальними, оскільки на сьогодні процес автоматизації та модернізації підприємств лише

прискорюється, розширюється, і комп’ютерні системи найчастіше замінюють старе обладнання, через що виникає необхідність регулювати вимоги роботи з новітніми пристроями, створювати необхідні умови для комфортної та ефективної, а головне – безпечної праці людини.

Згідно з проведеним дослідженням можна виділити наступні рекомендації покращення охорони праці з використанням комп’ютерних приладів:

– дотримання працівником та працедавцем усіх вимог та законів згідно регулювання охорони праці, відпочинку та медичних оглядів;

– дотримання заходів безпеки та організації власного робочого місця;

– забезпечення санітарно-гігієнічних вимог стосовно робочих кімнат на підприємстві;

– створення комфортних умов праці із дотриманням мікроклімату в приміщенні;

– використання лише справної техніки, які погоджені законодавством та регламентом;

– догляд за власним станом здоров’я та створення індивідуальних умов праці за необхідності.

ВИСНОВКИ

Глобальний розвиток комп'ютерних систем, інформаційних технологій та безпосередньо Інтернету створили безліч нових сфер, наукових розробок, та навіть змусили людей переосмислити деякі повсякденні речі. Якщо людині раніше було необхідно знайти інформацію – вона йшла до бібліотеки, щоб прочитати новини – купити газету, написати листа – купити листа, марку, та відправити на пошту, дочекатися відповіді. Тепер усе можна зробити майже миттєво через Інтернет.

На жаль, нові можливості приносять не лише позитивні результати, але й негативні – у зловмисників з'явилося більше способів шахрайства, зловживанням довірою людей або наявністю «дірок» у безпеці інформаційних систем. Одна із наявних проблем у 21-му сторіччі – це спам. Небажані листи розважального або комерційного характеру «забруднюють» інформаційний простір як звичайних користувачів, так і великих компаній, а можливість додавання файлів або посилань загрожують не тільки зіпсувати настрій, але й викрасти інформацію, фінанси тощо.

Саме тому розробка програмного рішення для розпізнавання семантичних особливостей тексту – тобто його змісту – є актуальною у наш час. Використання автоматизованих рішень, математичних та алгоритмів машинного навчання із підходами обробки природніх мов дозволяють реалізовувати сучасні системи протидії масовій розсилці спам-повідомлень та їх розпізнання.

Метою дипломного проекту була класифікація змісту електронних листів на наявність спаму з використанням методів та алгоритмів машинного навчання за допомогою розробленої інтелектуальної системи на мові програмування Python та бібліотек Scikit-Learn, PyQt5. У процесі виконання БКР були розглянуті алгоритми машинного навчання для рішення задач класифікації текстових даних, досліджено підходи NLP та їх застосування, а також наявні готові програмні рішення та реалізації у бібліотеці Scikit-Learn.

Для вирішення поставленої задачі були виконані наступні завдання:

– досліджено та проаналізовано алгоритми машинного навчання та штучного інтелекту;

– ознайомлено із підходами та застосуванням обробки природніх мов на прикладі класифікації текстових даних;

– проаналізовано найпопулярніші бібліотеки Python для рішення поставлених завдань;

– реалізовано алгоритм наївного Баєса для задачі бінарної класифікації із бібліотекою Scikit-Learn;

– створено програмне рішення доступу до електронних листів поштового сервісу Gmail користувачів через Gmail API;

– спроектовано інтелектуальну систему із реалізованих раніше програмних елементів з передбаченим графічним інтерфейсом для взаємодії з користувачем;

– протестовано додаток на наявність та необхідність виправлення помилок, розраховано точність алгоритму класифікації.

Результатом роботи є пояснювальна записка та розроблений програмний додаток із графічним інтерфейсом, який дозволяє авторизуватися у сервісі електронних листів Gmail, отримувати список листів користувача, та класифікувати їх зміст на наявність спаму. Комбінація можливостей Python та бібліотек Scikit-Learn, PyQt5, Gmail API дозволили створити автоматизовану систему для розпізнавання семантики електронного листа, яка використовує машинне навчання для класифікації змісту тексту.

Інтелектуальна система розроблена згідно технічного завдання та виконує основну задачу БКР, але може бути модифікованою в подальшому для покращення результатів та розширення можливостей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Обробка природної мови. *Вікіпедія*: вебсайт. URL: https://uk.wikipedia.org/wiki/Обробка_природної_мови (дата звернення: 05.05.2022).

2. Natural Language Processing (NLP). *IBM*: вебсайт. URL: https://www.ibm.com/cloud/learn/natural-language-processing (дата звернення: 05.05.2022).

3. Hobson Lane, Cole Howard, Hannes Hapke. Natural Language Processing in Action: Understanding, analyzing, and generating text with Python. Manning, 2019. 544 p.

4. Zolt´an Gyongyi, Hector Garcia-Molina. Web Spam Taxonomy: стаття. California : Stanford University, 2004. 9 с. URL: http://ilpubs.stanford.edu:8090/771/1/2005-9.pdf

5. Prateek Joshi. Artificial Intelligence with Python. Packt Publishing, 2017. 445 p.

6. Бахрушин В. Є. Методи аналізу даних : навч. посіб. Запоріжжя: КПУ, 2011. 268 с.

7. Біла Н. І. Інформаційні системи та технології в управлінні : метод. вказівки. Запоріжжя: ЗНТУ, 2014. 50с.

8. О. Є. Литвиненко, Д. А. Бурко. Моделі семантичного аналізу текстів. *Наукоємні технології*. 2009. № 4. URL: https://jrnl.nau.edu.ua/index.php/SBT/article/view/5246 (дата звернення: 07.05.2022).

9. Nils J. Nilsson. Introduction to machine learning. Stanford University, 1998. 188 p.

10. Tommaso Teofili. Deep Learning for Search 1st Edition. Manning, 2019. 328 p.

11. Теорема Баєса. *Вікіпедія*: вебсайт. URL: https://uk.wikipedia.org/wiki/Теорема_Баєса (дата звернення: 10.05.2022).

12. Рекурентна нейронна мережа. *Вікіпедія*: вебсайт. URL: https://uk.wikipedia.org/wiki/Рекурентна_нейронна_мережа (дата звернення: 10.05.2022).

13. Understanding LSTM Networks. *Colah`s Blog*: вебсайт. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs (дата звернення: 11.05.2022).

14. Rahul Dey, Fathi M. Salem. Gate-Variants of Gated Recurrent Unit Neural Networks : стаття. East Lansing : Michigan State University, 2017. 5 с.

15. Gated recurrent unit. *Wikipedia*: вебсайт. URL: https://en.wikipedia.org/wiki/Gated_recurrent_unit (дата звернення: 13.05.2022).

16. Python (programming language). *Wikipedia*: вебсайт. URL: https://en.wikipedia.org/wiki/Python_(programming_language) (дата звернення: 14.05.2022).

17. Getting Started. *Python™*: вебсайт. URL: https://www.python.org/about (дата звернення: 14.05.2022).

18. Machine Learning in Python. *Scikit-learn*: вебсайт. URL: https://scikit-learn.org/stable (дата звернення: 20.05.2022).

19. Python bindings for the Qt cross platform application toolkit: вебсайт. URL: https://pypi.org/project/PyQt5 (дата звернення: 20.05.2022).

20. Read and send messages. Manage drafts and attachments. Set up push notifications and manage settings. *Gmail for Developers*: вебсайт. URL: https://developers.google.com/gmail/api (дата звернення: 25.05.2022).

21. Сергеєв-Горчинський О. О., Іщенко Г. В. Інтелектуальний аналіз даних. Комп'ютерний практикум : посібник. Київ: КПІ, 2018. 75 с.

22. Convolution arithmetic. *Github*: вебсайт. URL: https://github.com/vdumoulin/conv_arithmetic (дата звернення: 25.05.2022).

23. Develop on Google Workspace. *Google Developers*: вебсайт. URL: https://developers.google.com/workspace/guides/get-started (дата звернення: 25.05.2022).

24. Learn about authentication & authorization. *Google Developers*: вебсайт. URL: https://developers.google.com/workspace/guides/auth-overview (дата звернення: 01.05.2022).

25. The OAuth 2.0 Authorization Framework. *Datatracker*: вебсайт. URL: https://datatracker.ietf.org/doc/html/rfc6749 (дата звернення: 01.05.2022).

26. Pyqt5-tools 5.15.4.3.2. *PyPi*: вебсайт. URL: https://pypi.org/project/pyqt5-tools/ (дата звернення: 02.05.2022).

27. Qt Modules. *Qt Documentation*: вебсайт. URL: https://doc.qt.io/qtforpython/modules.html (дата звернення: 03.05.2022).

28. PyCharm. The Python IDE for Professional Developers. *Jet Brains*: вебсайт. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 04.05.2022).

29. Spambase Data Set. *UC Irvine Machine Learning Repository*: вебсайт. URL: https://archive.ics.uci.edu/ml/datasets/spambase (дата звернення: 04.05.2022).

30. Зеркалов Д. В. Охорона праці в галузі: Загальні вимоги: навч. посіб. Київ: «Основа», 2011. 551 с.

31. Про затвердження Державних санітарних норм та правил «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу» : Закон України від 6 трав. 2014 р. за № 472/25249. URL: https://zakon.rada.gov.ua/laws/show/z0472-14 (дата звернення: 20.05.2022).

32. Катренко Л.А., Катренко А. В. Охорона праці в галузі комп'ютинґу: підручник. Львів: "Магнолія 2006", 2012. 544 с.

33. Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями : Наказ від 14 лют. 2018 р. № 207. URL: https://zakon.rada.gov.ua/laws/show/z0508-18 (дата звернення: 21.05.2022).

34. Про охорону праці : Закон України від 14 жов. 1992 р. за № 49. URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 22.05.2022).

35. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98 : Постанова від 10 грудня 1998 р. №7. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98 (дата звернення: 22.05.2022).

36. Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99 : Постанова від 1 грудня 1999 р. №42. URL: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення: 23.05.2022).

37. Санітарні норми виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037-99 : Постанова від 1 грудня 1999 р. №37. URL: https://zakon.rada.gov.ua/rada/show/va037282-99 (дата звернення: 23.05.2022).

38. Державні санітарні норми виробничої загальної та локальної вібрації ДСН 3.3.6.039-99 : Постанова від 1 грудня 1999 р. №39. URL: https://zakon.rada.gov.ua/rada/show/va039282-99 (дата звернення: 23.05.2022).

ДОДАТОК А
Реалізація доступу до Gmail API

```
class GmailService:
    def __init__(self):
        self.service = buildService()
        self.mails = None

    def getMessages(self) -> Any:
        result = self.service.users().messages().list(maxResults=10,
userId='me').execute()
        self.mails = result.get('messages')
        messages = self.getContext()
        return messages

    def getContext(self) -> Any:
        sendersList = []
        messagesList = []
        for msg in self.mails:
            txt = self.service.users().messages().get(userId='me',
id=msg['id']).execute()
            try:
                payload = txt['payload']
                headers = payload['headers']
                for d in headers:
                    if d['name'] == 'From':
                        sender = d['value']
                parts = payload.get('parts')[0]
                data = parts['body']['data']
                data = data.replace("-", "+").replace("_", "/")
                decodedData = base64.b64decode(data)
                sendersList.append(sender)
                messagesList.append(decodedData.decode('ascii'))
            except:
                pass

        return [sendersList, messagesList]
```

ДОДАТОК Б
Реалізація класу моделі наївного Баєса

```
class NaiveBayesModel:
    def __init__(self):
        self.bannedWords = ["!", "?", "$", ".", "=", "+", "'", ":", "(",
")", "%", ">", "<", "*", "@", "//", "\\", "|", "-", " "]
        self.countVect = CountVectorizer(lowercase='true',
stop_words=self.bannedWords)
        self.tfidfTransformer = TfidfTransformer()
        self.trainData = pd.read_csv('bayes/data/emails_train.csv',
             header=None, names=['Mail', 'Class'])
        self.trainModel = self.trainModel()

    def trainModel(self) -> Any:
        XTrainCounts =
self.countVect.fit_transform(self.trainData['Mail'])
        XTrainTfidf = self.tfidfTransformer.fit_transform(XTrainCounts)
        clf = MultinomialNB().fit(XTrainTfidf, self.trainData['Class'])
        return clf

    def testModel(self):
        testClf = Pipeline([
            ('vect', CountVectorizer(lowercase='true',
stop_words=self.bannedWords)),
            ('tfidf', TfidfTransformer()),
            ('clf', MultinomialNB()),])
        testClf.fit(self.trainData['Mail'], self.trainData['Class'])
        testData = pd.read_csv('bayes/data/emails_test.csv', header=None,
names=['Mail', 'Class'])
        docsTest = testData['Mail']
        predicted = testClf.predict(docsTest)
        print('Accuracy: ', np.mean(predicted == testData['Class']) *
100, '%.')
    def predictModel(self, message) -> Any:
        XNewCounts = self.countVect.transform([message])
        XNewTfidf = self.tfidfTransformer.transform(XNewCounts)
        predicted = self.trainModel.predict(XNewTfidf)
        return predicted
```