МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Кафедра програмування
(повна назва кафедри)

ДИПЛОМНА РОБОТА

Аналіз емоційної забарвленості тексту

Виконала: студентка групи ПМІ-41
спеціальності 122 – комп'ютерні науки
(шифр і назва спеціальності)

Костецька У.З.
(підпис) (прізвище та ініціали)

Керівник доц. Рикалюк Р.Є.
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

2023

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра програмування

Спеціальність 122 Комп'ютерні науки

«ЗАТВЕРДЖУЮ»
Завідувач кафедри Ярошко С.А.

" __ " _____ 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Костецькій Уляні Зеновіївній

(прізвище, ім'я, по батькові)

1. Тема роботи «Аналіз емоційної забарвленості тексту»

керівник роботи Рикалюк Роман Євстахович, доцент, к. ф.-м. н. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені Вченою радою факультету від "13"_вересня 2022 року №_15_.

2. Строк подання студентом роботи 13 червня 2023 року

3. Вихідні дані до роботи літературні джерела, інтернет-ресурси, постановка задачі, наукові статті

4. Зміст дипломної роботи (перелік питань, які потрібно розробити)

1. Дослідити галузь аналізу емоційної забарвленості тексту

2. Скласти специфікацію вимог необхідних для програмної реалізації та обрати технології розробки;

3. Програмно реалізувати обрані для дослідження алгоритми;

4. Порівняти ефективність застосування обраних алгоритмів для аналізу тексту

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної області	15.09.2022 – 10.11.2022	
2	Аналіз сайтів-аналогів	11.11.2022 – 25.11.2022	
3	Написання специфікації вимог	26.11.2022 – 03.012.2022	
4	Вибір технологій	04.012.2022 – 09.12.2022	
5	Пошук та обробка набору даних для аналізу	10.12.2022 – 24.12.2022	
6	Проектування архітектури ПЗ	25.12.2022 – 05.01.2023	
7	Розробка алгоритму Наївний Баєсів Класифікатор	06.01.2023 – 20.01.2023	
8	Розробка алгоритму Логістична Регресія	21.01.2023 – 10.02.2023	
9	Програмна візуалізація алгоритмів та даних	11.02.2023 – 02.03.2023	
10	Тестування програми	03.03.2023 – 10.03.2023	
11	Оформлення дипломної роботи	11.03.2023 – 15.05.2023	
12	Подання дипломної роботи	13.06.2023	

Студент _____ Костецька У.З.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Рикалюк Р.Є.
(підпис) (прізвище та ініціали)

ЗМІСТ

АНОТАЦІЯ

Аналіз емоційної забарвленості тексту велика та цікава галузь для дослідження. Для неї створено безліч алгоритмів, кожен з яких має свої переваги та недоліки.

Для цієї роботи було обрано два алгоритми - Наївний Баєсів Класифікатор і Логістична Регресія, та порівняно їх ефективність роботи.

Актуальність аналізу емоційної забарвленості тексту для пошуку російської пропаганди полягає в важливості виявлення та протидії дезінформації, яка негативно впливає на настої людей та перебіг подій в стані російсько-української війни.

У першому розділі досліджено такі теми як:

- Обробка природної мови
- Інтелектуальний аналіз даних
- Наївний Баєсів Класифікатор
- Логістична регресія

У другому розділі описано програмну реалізацію двох вищевказаних алгоритмів, порівняння їх ефективності в контексті пошуку пропаганди, а також результати їх роботи.

Для аналізу використано набори даних, взяті з платформи Kaggle, детальніше про них розказано в розділі 1.4.

Загальний обсяг роботи – 43 сторінки.

ВСТУП

У сучасному цифровому віці, коли інформація поширюється швидко та безперервно, люди активно користуються великою кількістю ресурсів призначених для вираження думок щодо різних подій, товарів та послуг.

Кожного дня величезна кількість новин та відгуків з’являється на просторах інтернету, що робить "ручний" збір та аналіз інформації малоефективним або ж не ефективним взагалі. Саме з цієї причини галузь комп'ютерної лінгвістики, яка зосереджена на автоматичній обробці текстів у природній мові зараз користується популярністю.

Завдання обробки текстів можна умовно розділити на дві категорії. Перша категорія включає завдання, з якими щодня стикаються всі користувачі, такі як перевірка орфографії та фільтрація спаму. З точки зору дослідників у галузі автоматичної обробки текстів (АОТ), всі ці завдання майже повністю вирішені і на сьогоднішній день більш актуальними стають завдання другої категорії, які потребують обробки великих текстових обсягів. Це включає аналіз думок і відгуків, пошук релевантних відповідей на запитання (так зване завдання "питання - відповідь") та розробку рекомендаційних систем, які працюють з великими обсягами неструктурованих даних. Особливістю таких завдань є їх складність та відсутність формалізації, що призводить до нестачі повного набору рішень та використання додаткових методів класифікації текстів, виділення ключових слів та словосполучень [15].

Аналіз емоційного забарвлення тексту або тональність тексту(*англ.Sentiment Analysis*) - це одна з найактивніших дослідницьких галузей в обробці природної мови, аналізі даних, пошуку інформації та веб-інтелекту.

Він використовується у багатьох сферах, таких як:

- Соціальні медіа - для виявлення емоційних реакцій користувачів на новини, події або продукти, що дозволяє компаніям та брендам налагоджувати зв'язок зі своїми клієнтами.
- Маркетинг та реклама – для розуміння, які емоції викликаються у споживачів продукту чи рекламного повідомлення, що дозволяє

маркетологам і рекламним агентствам працювати над покращенням ефективності своїх кампаній.

- Психологія та психіатрія - для визначення емоційного стану пацієнтів, що допомагає в розробці ефективної терапії.
- Організаційна культура - для вимірювання емоційної атмосфери в організації та виявлення проблем, які можуть впливати на ефективність працівників.
- Кримінальна юстиція - для виявлення злочинів та визначення емоційного стану свідків, потерпілих чи підозрюваних.

Станом на 2023 рік в Україні триває війна і інформація є одним з її інструментів. Російська пропаганда проти України має неабиякий вплив на формування думок не тільки українців але і людей у всьому світі. Метою такої пропаганди є: дискредитація влади в Україні, її політики, економіки та культури; поширення хаосу та невпевненості серед українського населення; створення та посилення політичних, соціальних та етнічних розколів; виправдання російської агресії проти України та створення у світовій громадськості спотвореного уявлення про події.

Враховуючи вищезазначені фактори, розробка та застосування алгоритмів класифікації новин та текстів для пошуку російської пропаганди стають важливим завданнями для ефективної протидії дезінформації та забезпечення інформаційної безпеки України в умовах тривалої війни.

Для аналізу тексту я обрала два алгоритми, а саме Наївний Баєсів Класифікатор та Логістичну Регресію та порівняла їх ефективність. Для реалізації дослідження в області аналізу тональності тексту було використано можливості мови Python.

Дані було взяти з соціальної мережі Twitter.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Обробка природної мови

Обробка природної мови(NLP) – загальний напрямок штучного інтелекту і математичної лінгвістики. Домен NLP охоплює всі взаємодії між комп'ютером і людиною, шляхом використання писемної чи розмовної природної мови, вивчає проблеми комп'ютерного аналізу і синтезу природних мов.

В останні роки спостерігається лавиноподібне зростання обсягу доступних неструктурованих даних представлених в таких формах як інформаційні пабліки, блоги та соціальні мережі. Технологія обробки природної мови є найбільш підходящим інструментом для аналізу такого типу інформації [6].

NLP використовується для вирішення різноманітних типів завдань, таких як: класифікація текстів, машинний переклад, пошук інформації в інтернеті, автоматична обробка документів, фінансова аналітика.

Проте для якомога якіснішої обробки природної мови виникає низка проблем, які необхідно вирішити. Природна мова недарма називається живою – вона постійно змінюється та підлаштовується під її носія. В залежності від культури, країни, регіону та навіть манери спілкування людини якість сприйняття мови комп’ютером може дуже відрізнятися.

Машинне навчання(*англ. Machine Learning*) широко використовується в обробці природної мови для вирішення різних типів проблем. Під час перебігу цього процесу система опрацьовує велику кількість прикладів, виявляє закономірності і використовує їх, щоб прогнозувати вихідні характеристики для нових вхідних даних [4].

Одним з багатьох етапів обробки природної мови є розбиття загального тексту на окремі елементи – ***лексеми***.

Лексема може мати декілька варіантів написання, наприклад, слово "книга" може бути написано як "книга", "книжка" або "книжечка". Використання лексем допомагає знизити кількість унікальних слів в тексті, що спрощує подальшу обробку і аналіз даних.

Токенізація - це процес розбиття тексту на окремі одиниці, які називаються токенами. Токени можуть бути словами, символами, фразами або іншими текстовими одиницями, залежно від контексту і використовуваного підходу. Вони стають основними елементами, з якими можна виконувати різні операції, в числі яких і обробка природної мови. Головною метою токенізації є створення даних, якими легше маніпулювати та які краще піддаються обробці.

Токенізація може бути простою, наприклад, розбиття тексту на окремі слова, або складною, тобто включати розпізнавання розділових знаків, чисел, абревіатур, емоджі та інших спеціальних символів. Для прикладу, простий токенізатор розділяє фразу “кожному місту звичай й права” на наступні п’ять токенів: “кожному” “місту” “звичай” “й” “права”.

Не менш важливим процесом є видалення стоп-слів. Користь такого процесу полягає в зменшенні обсягу даних, що позитивно впливає на швидкість та точність їх обробки.

Стоп-слова - це слова, які вважаються загальними та незначущими в контексті обробки природної мови і зазвичай відкидаються під час обробки. Такими словами є службові слова, які не несуть суттєвої інформації про зміст тексту, наприклад: прийменники, сполучники, займенники, артиклі, деякі частки мови і так далі. В українській мові такими словами можуть бути “і”, “а”, “або”, “але”, “як”, “що”, в англійській - “the”, “a” або “an.

Зазвичай в обробці природної мови використовуються готові програмні пакети стоп-слів, проте можна створити і нові, більш підходящі для виконання поставленого завдання.

Стемінг – процес зведення слів до їх основної форми, яка називається стемом шляхом відкидання закінчення чи суфікса. Наприклад, застосування стемінгу для слово "книжечка" перетворить його на стем "книг".

Варто відзначити, що стемінг може мати свої обмеження, так як він оперує правилами відкидання закінчення, не завжди досягаючи повної точності. До прикладу, можуть виникати випадкові співпадіння або слова з різними значеннями можуть мати однаковий стем.

Лематизація – це процес перетворення слова в словниковий вид або лему. Дієслова перетворюються на їх інфінітивну форму, іменники реконструюється до їх однини, а прислівники або прикметники передбачають їх позитивну форму. В порівнянні зі стемінгом, лематизація враховує морфологічні, граматичні та семантичні аспекти мови, що дозволяє отримати більш точну та зрозумілу базову форму слова. Для прикладу, для слова "пишуть" лемою є "писати".

1.2. Інтелектуальний аналіз даних

Data Mining (*укр. Інтелектуальний аналіз даних*) – процес виявлення в сирих даних раніше невідомих та нетривіальних, практично корисних і доступних для інтерпретації знань, необхідних для ухвалення рішень в різних сферах людської діяльності [1]**.**

Традиційна математична статистика, яка довгий час претендувала на роль основного інструменту аналізу даних, не відповідала новим проблемам. Головна причина — концепція усереднювання по вибірці, що тягне за собою операції над фіктивними величинами. Методи математичної статистики виявилися корисними, головним чином, для перевірки наперед сформульованих гіпотез і для «грубого розвідувального аналізу», який є основою оперативної аналітичної обробки даних OLAP [5].

Основа сучасної технології Data Mining — концепція шаблонів (pattern), що відображають фрагменти багатоаспектних взаємостосунків даних. Цими шаблонами є закономірності, властиві підвибіркам даних, які можуть бути компактно виражені у формі, зрозумілій людині. Пошук шаблонів проводиться методами, не обмеженими рамками апріорних припущень про структуру вибірки і видом розподілів значень аналізованих показників [5].

Причини популярності Data Mining:

- Збільшення обсягів даних: Сьогодні існує велика кількість даних, які можуть бути використані для виявлення тенденцій та корисної інформації. Data Mining допомагає в аналізі цих даних та виявленні цінної інформації.

- Зростання потреби у прийнятті рішень на основі даних: У сучасному світі все більше бізнес-процесів залежать від даних, тому Data Mining стає важливим інструментом для прийняття.

- Покращення якості продукту та обслуговування: Data Mining допомагає виявляти потреби та проблеми клієнтів, що дозволяє покращувати якість продукту та обслуговування.

- Розвиток технологій: З розвитком технологій обробки даних та комп'ютерних можливостей, Data Mining стає все більш доступним та ефективним.

- Ефективність та економія часу: Data Mining дозволяє автоматизувати аналіз даних, що зменшує час та зусилля, необхідні для отримання корисної інформації.

Перелік основних задач Data Mining:

❖ Класифікація (*англ. Classification*).

Найпростіша і найпоширеніша задача Data Mining. Класифікація є процесом визначення приналежності об'єкта до певного класу або категорії на основі його атрибутів чи властивостей. Це один з основних методів навчання з вчителем (*англ. supervised learning*), де модель використовує навчальний набір даних, що містить завчасно відомі відповіді або мітки для тренування.

У процесі класифікації, алгоритми Data Mining навчаються розпізнавати закономірності і вибудовувати модель, яка може прогнозувати клас або категорію нових невідомих об'єктів. Це досягається шляхом використання різних алгоритмів

класифікації, які можуть бути засновані на правилах, статистиці, штучних нейронних мережах, деревах прийняття рішень та інших методах.

Основна мета класифікації в Data Mining - знаходження відповідності між вхідними атрибутами об'єкта і його класифікацією.

Класифікація допомагає здійснювати автоматичне призначення класів на основі аналізу вхідних даних та виробляти прийняття рішень на основі цих класифікаційних моделей.

- Кластеризація (*англ. Clustering*).

Логічне продовження ідеї класифікації. Кластеризація в Data Mining є методом групування схожих об'єктів разом у визначені кластери на основі їхньої схожості чи подібності. Це один з основних методів без вчителя(*англ. unsupervised learning*), де модель аналізує дані без наявності завчасно відомих відповідей або міток.

У процесі кластеризації, алгоритми Data Mining намагаються знайти природні групи або кластери в наборі даних. Кластери формуються таким чином, що об'єкти в межах кожного кластера схожі між собою, а об'єкти з різних кластерів відрізняються.

Основні принципи кластеризації включають в себе визначення міри схожості між об'єктами, вибір алгоритму кластеризації та визначення кількості кластерів або їхніх центрів.

Кластеризація допомагає відкривати структуру та взаємозв'язки в наборах даних, робити прогнози та приймати рішення на основі групування схожих об'єктів.

- Асоціація (*англ. Associations*).

Асоціація в Data Mining відноситься до процесу виявлення зв’язаних подій в наборі даних. Цей процес розглядається як навчання без вчителя.

Аналіз асоціацій спрямований на виявлення комбінацій подій, які часто зустрічаються. Ці зв'язки називаються асоціативними правилами. Зазвичай, асоціативні правила виражаються у вигляді "якщо/тоді" появлень, де поява певних подій спричиняє появлення з деякою ймовірністю інших подій.

Наприклад, в аналізі продажів можна знайти асоціативні правила, що показують, що якщо покупець придбав хліб, то з ймовірністю 80% він також придбає молоко. Ці правила можуть бути корисні для рекомендаційних систем або планування стелажів в магазинах. Один з основних підходів до виявлення асоціативних правил - це використання алгоритму Apriori, який базується на понятті підтримки (*англ. support*) і достовірності (*англ. confidence*) правил. Підтримка відображає частоту з'явлення асоціативного правила у наборі даних, а достовірність вимірює ймовірність, з якою правило справджується.

- Послідовність (*англ. Sequence*).

Послідовність в Data Mining відноситься до типу даних і аналітичних методів, які використовуються для аналізу та виявлення залежностей між подіями або об'єктами, які відбуваються у певному порядку часу або послідовності. Послідовність може бути представлена як послідовність подій, послідовність транзакцій, послідовність поведінки або інших впорядкованих даних.

В аналізі послідовностей головна мета полягає у виявленні цікавих шаблонів, залежностей або правил, що їх характеризують. Це можуть бути шаблони послідовностей, які часто зустрічаються, правила асоціацій між подіями, прогнозування майбутніх подій або виявлення аномалій.

- Прогнозування (*англ. Forecasting*).

Прогнозування в Data Mining - це процес використання аналітичних методів та алгоритмів для передбачення майбутніх подій, станів або значень на основі існуючих даних. Це один з ключових аспектів Data Mining, який дозволяє робити інформовані припущення та прогнози на основі виявлених залежностей та шаблонів у даних.

Прогнозування в Data Mining може мати широкий спектр застосувань. Наприклад, в бізнесі воно може бути використане для прогнозування продажів, попиту на товари, цін на ринку, клієнтських поведінкових тенденцій та іншого. У фінансовій сфері може використовуватися для прогнозування фінансових показників, ризиків, коливань ринкових цін та інвестиційних можливостей.

Для прогнозування використовуються різні алгоритми та методи, такі як лінійна регресія, навчання з учителем, нейронні мережі, дерева рішень, алгоритми класифікації та регресії, часові ряди та інші. Прогнозування може базуватися на статистичних методах, машинному навчанні, штучних нейронних мережах та інших техніках.

Прогнозування в Data Mining є потужним інструментом для прийняття рішень, планування та управління в різних галузях, де важливо мати передбачення щодо майбутнього на основі наявних даних.

- Візуалізація (*англ. Visualization, Graph Mining*).

Візуалізація в Data Mining - це процес використання графічних методів та інструментів для представлення та візуалізації даних з метою виявлення взаємозв'язків, трендів, закономірностей та інших цікавих аспектів в наборах даних. Вона дозволяє перетворити складні структури даних на зрозумілі візуальні форми, такі як графіки, діаграми, картографічні зображення, інфографіки та інші графічні елементи. Візуалізація допомагає аналітикам та дослідникам краще розуміти дані, знаходити нові інсайти, комунікувати результати та приймати обґрунтовані рішення. Прикладом методів візуалізації є представлення даних в 2D і 3D вимірюваннях.

- Підведення підсумків (*англ. Summarization*).

Підведення підсумків відноситься до процесу створення компактного, короткого та інформативного опису або зведення набору даних, виявлених під час

аналізу. Це процес витягування ключових аспектів, шаблонів, тенденцій або особливостей з великої за обсягом або складної інформації.

Підведення підсумків може використовуватись для зведення даних до більш зрозумілого та компактного представлення, що полегшує аналіз та виявлення значущих патернів. Це може включати створення статистичних звітів, графічних діаграм, ключових висновків або навіть автоматичне генерування текстових описів[9].

Воно допомагає зрозуміти суттєвість та важливість виявлених даних, спрощує комунікацію результатів аналізу та дозволяє зробити обґрунтовані висновки та прийняти рішення на основі отриманих висновків.

1.3. Python технології

1.3.1. Python Web Scraping (парсер HTML сторінок)

Інструменти web scraping (парсинг) розроблені для вилучення та збору відкритої інформації з веб-сайті, призначених для перегляду людьми. Ці ресурси призначені для швидкого отримання та зберігання в певному структурованому вигляді будь-яких даних з інтернету. Парсинг сайтів - це новий метод отримання даних, для якого не потрібні люди чи копіпаст. Таке програмне забезпечення здійснює пошук інформації під контролем користувача або в автоматичному режимі, вибираючи нові або оновлені дані. Дані зберігаються у потрібному вигляді, щоб користувач мав до них швидкий доступ.

Парсери зазвичай розробляються для кожного сайту окремо, враховуючи його структурні та технічні особливості. Існують і готові рішення, які дозволяють отримувати потрібну інформацію з сайту після попередньої конфігурації, не написавши жодного рядка коду [2].

Вилучення даних парсером з інтернету ділиться на два етапи, які виконуються у відповідному порядку [17]:

1. Знаходження веб-ресурсів, які нам потрібні

2. Вилучення потрібної інформації з отриманих даних веб-сайту.

Будь-яка програма-парсер розпочинає свою роботу зі створення HTTP запиту на отримання даних від цільового веб-сайту. Якщо запит успішно отримано та оброблено цільовим веб-сайтом, то в результаті отримуємо дані з потрібного веб-сайту. Відповідь веб-сайту може бути у різному форматі в залежності від того, як веб-сайт організований. Серед популярних форматів існують HTML власне з чого побудовані майже всі веб-сайти. Потоки даних також можуть бути у вигляді JSON або рідше у вигляді XML. Після завантаження веб-даних, програма аналізує, переформатовує та організовує дані у певний структурований вигляд [2].

1.3.2. Обробка текстових даних за допомогою бібліотеки NLTK

NLTK(англ. Natural Language Toolkit) - це пакет Python, створений для обробки природної мови.

Його можна використовувати для класифікації, токенізації, стемінгу, категоризації, синтаксичного та семантичного аналізу. Бібліотека NLTK створена для роботи у середовищі Python.

Обробка даних починається з проведення всіх символів тексту до нижнього регістру. Після цього з тексту видаляться знаки пунктуації, небуквенні символи та цифри. В бібліотеці NLTK є вбудований список стоп-слів, який можна використати для видалення таких слів з масиву даних. Проте варто зауважити, що цей список не є універсальним, тому в залежності від завдання його можна редагувати [16].

1.3.3. Короткий огляд бібліотек для візуалізації даних

- Plotly

Бібліотека Python з відкритим вихідним кодом, особливістю якої є створення необмеженої кількості графіків в автономному режимі, а також можливість створювати до 25 діаграм онлайн [11].

Приклад роботи бібліотеки відображено на рис. 1.

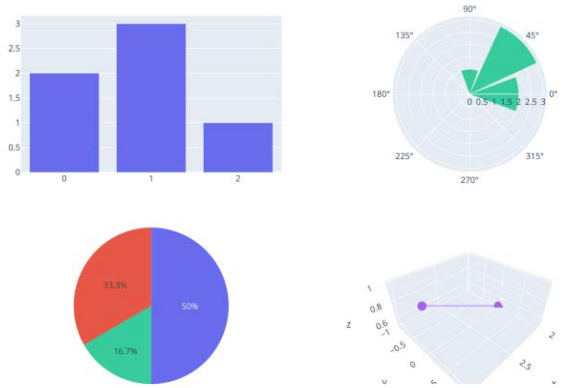

Рисунок 1 – Приклад роботи бібліотеки Plotly

❖ Pandas

Бібліотека надає функції та структури даних для покращення роботи із структурованими даними. Пакет надає можливість будувати зведені таблиці, виконувати угрупування, надає доступ до табличних даних, а при наявності matplotlib дає можливість будувати графіки на отриманих наборах даних. Pandas представляє дві основні структури даних: DataFrame та Series [3].

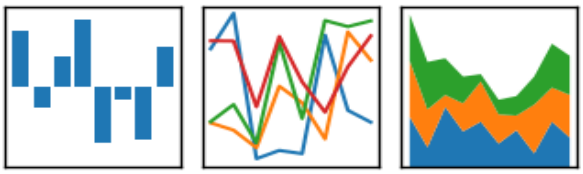

Рисунок 2 – Приклад роботи бібліотеки Pandas

❖ Seaborn

Бібліотека візуалізації даних Python, заснована на основі пакету Matplotlib, яка забезпечує високорівневий інтерфейс для зображення інформативної статистичної графіки.

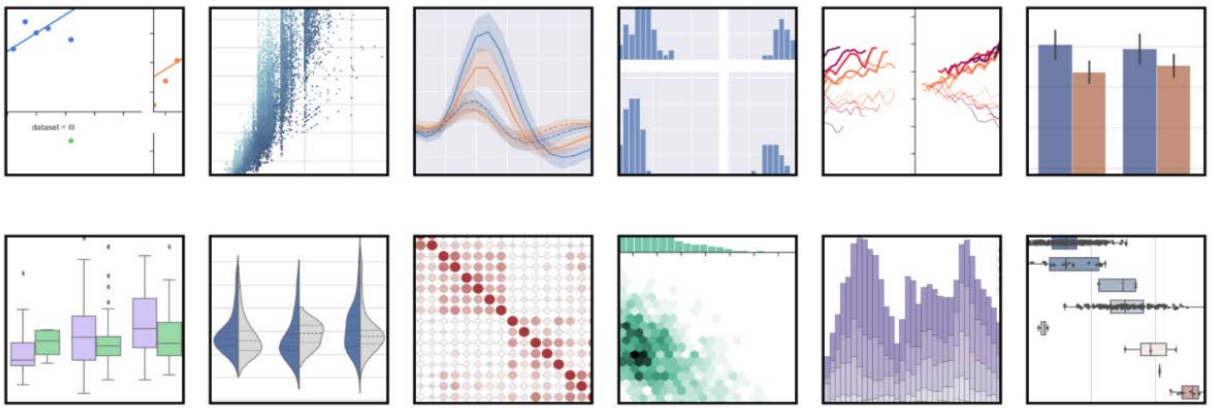

Рисунок 3 – Приклад роботи бібліотеки Seaborn

❖ Matplotlib

Бібліотека, яка є потужним інструментом для візуалізації даних в мові програмування Python. Вона дозволяє створювати різноманітні типи графіків, діаграм, графіків розсіювання та інших візуальних представлень даних.

Рисунок 4 – Приклад роботи бібліотеки Matplotlib

❖ Wordcloud

Бібліотека яка є інструментом для створення хмар слів (word clouds) - візуальних представлень текстових даних, де частотність кожного слова представлена його розміром. Wordcloud дозволяє швидко отримати загальну картину про найчастіші слова або терміни у текстових документах.

Рисунок 5 – Приклад роботи бібліотеки Wordcloud

1.4. Огляд алгоритмів для аналізу даних

1.4.1. Наївний баєсів класифікатор

Наївний Баєсів Класифікатор (*англ. Naive Bayes classifier*) є одним з простих та ефективних алгоритмів класифікації, який базується на теоремі Баєса. Він використовує статистичні методи для припущення незалежності між ознаками (параметрами) та знаходження ймовірностей належності до певного класу для нових екземплярів даних.

Основна ідея Наївного Баєсового Класифікатора полягає в припущенні, що всі ознаки незалежні між собою, тобто взаємна залежність між ознаками відсутня. Ця припущена незалежність спрощує обчислення ймовірностей та спрощує модель.

Основні кроки в роботі Наївного Баєсового Класифікатора:

- Зібрати набір тренувальних даних, який складається з прикладів з відомими класами.
- Обчислити апріорні ймовірності для кожного класу, що визначають, як часто кожен клас зустрічається у тренувальних даних.
- Обчислити умовні ймовірності для кожної ознаки, враховуючи класи. Це означає обчислення ймовірностей наявності певної ознаки в кожному класі.

- Використовуючи отримані ймовірності, обчислити апостеріорні ймовірності для нових екземплярів даних та призначити їм клас з найвищою ймовірністю.

1.4.1.1. Теорема Баєса

Формула Баєса в контексті класифікації тексту використовується для обчислення ймовірності належності тексту до певного класу, враховуючи ймовірності класів та ймовірності входження ознак (слів) в ці класи. Формально формула Баєса має наступний вигляд:

$$P(y|x_1, x_2,..., x_n) = (P(y)\, P(x_1, x_2,..., x_n|y)) \div (P(x_1,\ x_2, \ldots,\ x_n)) \quad (1)$$

де:

y - змінна класу, x - залежний вектор ознак;

$P(y|x_1, x_2,..., x_n)$ - ймовірність того, що текст належить до певного класу, враховуючи текст;

$P(y)$ - апріорна ймовірність класу, тобто ймовірність належності тексту до класу без урахування ознак тексту;

$P(x_1, x_2,..., x_n|y)$ - ймовірність тексту при умові належності до певного класу, тобто ймовірність входження ознак тексту, враховуючи клас;

$P(x_1, x_2,..., x_n)$ - ймовірність тексту без урахування класу;

На етапі навчання моделі розраховуються апріорні ймовірності класів $P(x_1, x_2,..., x_n)$ та ймовірності входження окремих ознак (слів) в кожен клас $P(x_1, x_2,..., x_n|y)$. Це робиться з використанням навчального набору даних, де для кожного тексту відомий його клас.

На етапі класифікації нового тексту розраховуються ймовірності для кожного класу $P(y|x_1, x_2,..., x_n)$, використовуючи формулу Баєса і відповідні ймовірності,

обчислені на попередньому етапі. Обчислення включають врахування ймовірності класу $P(y)$ та множення ймовірностей входження окремих ознак в текст при умові належності до класу $P(x_1, x_2,..., x_n|y)$.

Класифікація тексту здійснюється шляхом вибору класу з найвищою ймовірністю належності $(y|x_1, x_2,..., x_n)$.

1.4.1.2. Многочленний класифікатор Баєса

Многочленний класифікатор Баєса(*англ. Multinomial Naive Bayes classifier*) є варіантом Наївного Баєсового Класифікатора, який використовується для класифікації текстових даних.

Основна відмінність полягає у використанні розподілів ймовірностей мультиноміального типу. Кожен термін або слово в текстовому документі розглядається як окрема категорія або ознака, а його використання вимірюється кількістю входжень.

У многочленній моделі кожен вектор ознак x_i представляє частоту входжень певної події. Ймовірність спостереження гістограми x задається формулою:

$$p(x|C_k) = \frac{(\sum_{i=1}^{n} x_i)!}{\prod_{i=1}^{n} x_i!} \prod_{i=1}^{n} p_{k_i}^{x_i} \quad (2)$$

Для уникнення проблеми, коли ймовірності на основі частоти стають нульовими або дуже малими використовуються такі способи регуляризації наївного Баєса як згладжування Лапласа(*англ. Laplace smoothing*) і згладжування Лідстона(*англ. Lidstone smoothing)*. Ці методи додають певну корекцію до оцінок ймовірностей, що дозволяє зберегти інформацію з усіх ознак, навіть якщо деякі з них не зустрічаються у навчальних даних.

Згладжування Лапласа, також відоме як апріорне згладжування, полягає в додаванні невеликої константи (зазвичай 1) до кількості випадків входження ознаки у вибірці. Це означає, що кожна ймовірність входження ознаки

обчислюється з урахуванням її входжень і доданої константи. Таким чином, навіть якщо ознака не зустрічається у вибірці, ймовірність не стане нульовою.

Згладжування Лідстона є модифікацією згладжування Лапласа, де використовується необхідна змінна константа для згладжування. Замість постійного додавання однієї константи до всіх оцінок ймовірностей, використовується така, що залежить від деякого параметра (зазвичай позначеного як "α"). Цей параметр дозволяє контролювати рівень згладжування, де менші значення α призводять до більшого згладжування, а більші значення α збільшують вагу оригінальних частот.

1.4.2. Логістична регресія

Логістична регресія(*англ. Logistic Regression*) є ще одним із методів машинного навчання, що дозволяє класифікувати текстові документи. Вона може бути застосована для аналізу соціальних медіа, відгуків користувачів, коментарів та інших текстових джерел.

Логістична регресія є статистичним методом для прогнозування бінарних або категоріальних виходів на основі вхідних змінних [10].

В цьому методі класифікації вхідні змінні мають вплив на вихідний результат, який може бути в бінарному вигляді (так/ні, 0/1) або категоріальному (наприклад, A, B, C). Метод моделює логістичну функцію за допомогою параметрів, які відображають вплив кожної змінної на ймовірність належності до певного класу. Ці параметри оцінюються шляхом максимізації ймовірності спостережень у навчальних даних [12].

Однією з ключових переваг логістичної регресії є можливість отримувати ймовірності належності до певного класу, що дозволяє робити більш інформовані прогнози та приймати рішення на основі цих ймовірностей. Також через те, що логістична регресія добре працює з числовими та категоріальними змінними вона може бути розширена для моделювання багатокласових виходів.

Алгоритм роботи логістичної регресії можна описати наступним чином:

1. Збір даних: Збираються дані, враховуючи вхідні змінні (фактори) і вихідний результат (змінна, яку треба прогнозувати).
2. Підготовка даних: Вхідні дані можуть потребувати попередньої обробки, такої як видалення відсутніх значень, нормалізації або кодування категоріальних змінних.
3. Визначення моделі: Визначається логістична модель, яка використовується для прогнозування ймовірності належності до певного класу. Логістична модель використовує логістичну функцію для оцінки ймовірності.
4. Оцінка параметрів: Використовуючи навчальні дані, параметри моделі оцінюються шляхом максимізації ймовірності спостережень. Це може бути здійснено за допомогою методу максимальної ймовірності або ітеративних алгоритмів, таких як градієнтний спуск.
5. Побудова моделі: Після фіксації параметрів моделі, оцінених на попередньому кроці, модель може бути побудована.
6. Прогнозування: За допомогою побудованої моделі можна здійснити прогноз для нових невідомих даних. Модель повертає ймовірність належності до певного класу, і можна встановити порогове значення для класифікації.
7. Оцінка моделі: Оцінка моделі включає аналіз результатів, таких як матриця помилок, точність, чутливість, специфічність та інші метрики, що визначають ефективність моделі.

1.4.2.1. Логістична функція

Логістична функція використовується в логістичній регресії для моделювання ймовірності належності тексту до певного класу, наприклад, до класу пропаганда або не пропаганда. Вона перетворює ваговану суму вхідних ознак на значення ймовірності за допомогою нелінійної функції.

Одна з найпоширеніших логістичних функцій, використовуваних в логістичній регресії, називається "сигмоїда" або "логістична функція". Вона має таку формулу:

$$f(x) = \frac{1}{1+\exp(-x)} \quad , \tag{3}$$

де:

- $f(x)$ - значення функції сигмоїди для вхідного значення x.
- $\exp(-x)$ - експонента, яка підносить число експоненти до степені -x.

Сигмоїдна функція приймає будь-яке дійсне число x як вхід та повертає значення в діапазоні від 0 до 1. Це значення інтерпретується як ймовірність належності тексту до певного класу (наприклад, до класу пропаганда). Чим більше значення сигмоїди, тим вища ймовірність належності до класу [14].

Також логістична функцію може бути виражена наступним чином:

$$P(y = 1|x) = \frac{1}{1+\exp\left(-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_n x_n)\right)} \tag{4}$$

де $P(y = 1|x)$ є ймовірністю належності до класу 1 при заданих вхідних змінних x_1, x_2, ..., x_n. β_0, β_1, β_2, ..., β_n є параметрами моделі, які потрібно оцінити[14].

Оцінка параметрів логістичної регресії може бути здійснена за допомогою методу максимальної ймовірності або ітеративних алгоритмів, таких як градієнтний спуск.

1.4.2.2. Метод максимальної ймовірності

Метод максимальної ймовірності є статистичним підходом до оцінювання параметрів моделі в логістичній регресії. В основі цього методу лежить ідея максимізації ймовірності спостережень на основі навчальних даних.

У логістичній регресії, ми хочемо оцінити параметри моделі, щоб знайти найбільш імовірні значення, що пояснюють спостереження. Ймовірність спостережень може бути визначена за допомогою логістичної функції

Цей метод використовує навчальні дані для оцінки параметрів моделі шляхом максимізації ймовірності спостережень. Це означає, що ми шукаємо такі значення параметрів, які максимізують ймовірність спостережень, що ми фактично спостерігаємо у навчальних даних [7].

Максимізація ймовірності зазвичай досягається шляхом використання методів оптимізації, таких як градієнтний спуск чи ітеративні числові методи.

1.4.2.3. Градієнтний спуск

Градієнтний спуск є оптимізаційним алгоритмом, що використовується для оцінки параметрів моделі в логістичній регресії. Його основна мета полягає у знаходженні наближених оптимальних значень параметрів шляхом мінімізації функції втрат.

У логістичній регресії, функція втрат, яку ми намагаємося мінімізувати, називається функцією логарифмічної втрати (*англ. log loss*) або бінарною хрес-ентропією. Градієнтний спуск використовує похідні цієї функції втрат щодо кожного параметра моделі, щоб оновити їх значення на кожній ітерації.

Процес градієнтного спуску можна уявити таким чином: на початку, параметри моделі ініціалізуються деякими початковими значеннями. Потім на кожній ітерації алгоритм обчислює градієнт функції втрат, який представляє схил функції втрат у напрямку найшвидшого зростання. Параметри моделі оновлюються, рухаючись в протилежному напрямку градієнта з певним кроком, відомим як швидкість навчання (*англ. learning rate*). Цей процес продовжується до досягнення заданої кількості ітерацій або до досягнення зупинки згідно з певною умовою.

Градієнтний спуск є ефективним методом оптимізації в логістичній регресії, оскільки він дозволяє знайти локальні мінімуми функції втрат та покращити точність моделі [13].

1.4.4. Порівняння алгоритмів

Наївний Баєсів Класифікатор і логістична регресія є двома різними алгоритмами машинного навчання, які можна використовувати для аналізу емоційної забарвленості тексту. Незважаючи на те, що вони мають різні підходи, обидва методи є популярними і часто застосовуються.

Ключові відмінності між алгоритмами:

❖ Підхід до моделювання:

Наївний Баєсів Класифікатор ґрунтується на теоремі Баєса та припущенні про наївність (незалежність) між ознаками. Він моделює умовну ймовірність належності до певного класу, використовуючи ознаки тексту. З іншого боку, логістична регресія моделює логістичну функцію для прогнозування ймовірності належності до класу, використовуючи лінійну комбінацію ознак і їх ваг.

❖ Обробка залежностей між ознаками:

Наївний Баєсів Класифікатор передбачає незалежність між ознаками, тобто вважає, що кожна ознака впливає на клас незалежно від інших ознак. Це припущення, хоча й наївне, спрощує модель і полегшує обчислення. З іншого боку, логістична регресія не має такого припущення і може моделювати складніші залежності між ознаками.

❖ Інтерпретованість результатів:

Логістична регресія зазвичай надає більшу інтерпретованість результатів, оскільки можна аналізувати вплив кожної ознаки на вихідний результат. Вона надає значення коефіцієнтів, що вказують вплив ознак на класифікацію. У Наївного Баєсового Класифікатора немає такої прямої інтерпретації результатів.

❖ Обробка відсутніх даних:

Наївний Баєсів Класифікатор може працювати добре навіть у випадках, коли дані мають відсутні ознаки. Він використовує умовні ймовірності для класифікації,

що робить його менш чутливим до відсутності даних. З іншого боку, логістична регресія може вимагати обробки відсутніх значень, наприклад, заповнення їх середніми значеннями або виключення рядків з відсутніми даними.

❖ Розмір даних:

Логістична регресія може виявитися більш ефективною для великих обсягів даних, оскільки може швидше навчатися за допомогою градієнтного спуску та інших оптимізаційних методів. Наївний Баєсів Класифікатор, хоча і має простіший підхід, може бути менш ефективним при обробці великих наборів даних.

1.4. Огляд даних для аналізу

Для своєї роботи я використала два набори даних взятих з платформи Kaggle. Kaggle — це платформа для змагань з аналітики та передбачувального моделювання, в рамках якого статистики та добувачі даних конкурують у створенні найкращих моделі для прогнозування та опису даних, запропонованих компаніями або користувачами[18].

Перший набір даних називається russian_propaganda_tweets[19], а другий western_analysts_tweets[20]. Для роботи мені потрібні були лише стовпці з твітами, тож всі інші з обох наборів було видалено. Після видалення всього зайвого я додала до даних в першому наборі мітку з назвою label та класифікувала їх як propaganda, в другому наборі дані отримали класифікацію not propaganda. Наступним кроком було об’єднання наборів даних в один.

Загальна кількість твітів до обробки - близько 40 тис., проте багато з них вирвані з контексту тож назвати їх пропагандою чи не пропагандою складно. Щоб усунути цю проблему, я припустила що твіти, які містять в собі слова з коренями russia, ukrain, war, Mariupol і т.д. більш ймовірно матимуть сенс у контексті мого аналізу. Отже всі інші твіти було видалено. Після всіх маніпуляцій загальна кількість твітів у наборі даних становить 14 829, з яких 6624 пропаганда, а 8205 не пропаганда.

1. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

2.1. Інструменти використані в аналізі

Для аналізу емоційної забарвленості тексту я обрала мову програмування Python. Python - це інтерпретована об’єктно орієнтована мова програмування, вона досить високого рівня та має строгу динамічну типізацію.

Середовищем розробки використала платформу Anaconda. Anaconda є популярною платформою для науки про дані та “наукового обчислення” на мові Python. Вона надає всеохоплююче та інтегроване середовище, яке включає менеджер пакетів, менеджер середовища та різні інструменти для аналізу даних, машинного навчання та візуалізації.

У цій роботі я зосередила увагу на двох алгоритмах, а саме Naive Bayes Classifier та Logistic Regression.

Наївний Баєсів Класифікатор заснований на теоремі Баєса. Це швидкий, точний і надійний алгоритм для роботи з великими наборами даних.

Логістична регресія - це статистичний метод прогнозування бінарних класів. Результат або цільова змінна має дихотомічний(існування лише двох можливих класів) характер. Цей метод обчислює ймовірність настання події.

Для обробки текстових даних було обрано бібліотеку NLTK. ЇЇ я використала для класифікації, токенізації, стемінгу, видалення стоп-слів, категоризації, синтаксичного аналізу та семантичного аналізу даних.

```
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
```

Рисунок 6 – Підключення модулів бібліотеки NLTK

Модуль re в Python надає функціональність для роботи з регулярними виразами, що дозволяє виконувати складні операції пошуку, валідації та маніпуляції текстом. З його допомогою можна виконувати шаблонний пошук та заміну тексту в рядках, використовуючи регулярні вирази. У своїй роботі я

використала цей модуль для видалення URL адреси, хештегів, згадок та пунктуації з даних.

Для зчитування даних з CSV-файлу та маніпуляцій з ними використовую бібліотеку pandas.

Модуль sklearn (Scikit-learn) є одним з найпопулярніших пакетів для машинного навчання і аналізу даних. Він надає багато функцій і інструментів для завантаження, підготовки, візуалізації та моделювання даних. Він підтримує багато алгоритмів машинного навчання, надає зручний інтерфейс для роботи з даними та моделями, і дозволяє легко виконувати операції навчання, прогнозування і оцінки моделей. У своїй роботі використала цей модуль для таких операцій[8]:

- витягування ознак з тексту, використовуючи метод TF-IDF (term frequency - inverse document frequency). Необхідний для отримання числових ознак з обробленого тексту.
- тренування моделі Логістичної регресії на основі отриманих ознак та міток.
- поділ даних на тренувальний та тестовий набори
- обчислення точності моделі, яка використовується для оцінки якості класифікації.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Рисунок 7 – Підключення модулів бібліотеки Sklearn

Бібліотека LogisticRegression, яка є частиною бібліотеки scikit-learn надає реалізацію логістичної регресії, яка є популярним алгоритмом машинного навчання для бінарної класифікації.

Логістична регресія використовує логістичну функцію для оцінки ймовірності віднесення прикладу до певного класу. Вона є лінійною моделлю, де

вхідні ознаки зважуються відповідними коефіцієнтами, а потім подаються на вхід логістичній функції для отримання імовірності.

Клас LogisticRegression забезпечує різні параметри для керування моделлю, такі як регуляризація, тип оптимізації, максимальна кількість ітерацій та інші. Він має методи для навчання моделі на вхідних даних, прогнозування класів для нових даних та оцінки точності моделі.

Для обчислень на великих масивах даних використовую бібліотеку NumPy(Numerical Python). Ця бібліотека є однією з основних для наукових обчислень та роботи з числовими даними у мові програмування Python. Вона надає потужні інструменти для виконання обчислень на великих масивах даних, включаючи математичні, лінійну алгебру, обробку образів, роботу з даними у форматі таблиць та багато іншого.

Для візуалізації даних обрала такі бібліотеки: matplotlib, plotly та wordcloud.

2.2. Обробка даних для роботи з алгоритмами

Для роботи з Наївним Баєсовим Класифікатором та Логістичною регресією дані потрібно обробити.

Попередня обробка даних є одним із важливих кроків у будь-якому проекті машинного навчання і цей проект не виняток. Вона включає в себе очищення та форматування даних перед подачею їх в алгоритм машинного навчання.

Етапи обробки складаються з таких завдань:

- токенізація;
- видалення стоп-слів;
- видалення URL адрес, хештегів, згадок та пунктуації;
- стемінг

```python
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()

def preprocess_text(text):
    # Remove URLs
    text = re.sub(r"http\S+|www\S+|https\S+", "", text)
    # Remove hashtags and mentions
    text = re.sub(r"\#\w+|\@\w+", "", text)
    # Remove punctuation and convert to lowercase
    text = re.sub(r'[^\w\s]', '', text.lower())
    # Tokenize
    tokens = word_tokenize(text)
    # Remove stopwords
    tokens = [token for token in tokens if token not in stop_words]
    # Remove stopwords and apply stemming
    tokens = [stemmer.stem(token) for token in tokens if token not in stop_words]
    # Join tokens back to text
    text = ' '.join(tokens)
    return text
```

Рисунок 8 – Реалізація попередньої обробки даних

Приклад обробки тексту зображено на рис. 9.

```
[30] preprocess_text("A few separatists along with their BMP-2 IFV were captured in"+
     "Stanitca Luganskaya, #Luhansk by Ukrainian Forces.  https://t.co/nrN4Zu0qK9")

'separatist along bmp2 ifv captur instanitca luganskaya ukrainian forc'
```

Рисунок 9 – Приклад обробки тексту

2.3. Тренування та тестування алгоритмів

Наступним кроком після обробки є тренування та тестування алгоритмів. Для Наївного Баєсового Класифікатора я розписала роботу класу “NaiveBayesClassifier”, а для Логістичної регресії використала вбудовану в модуль sklearn бібліотеку LogisticRegression.

2.3.1. Тренування Наївного Баєсового Класифікатора

У конструкторі класу NaiveBayesClassifier спочатку ініціалізуються різні змінні, такі як class_counts, token_counts, total_documents, vocab та stemmer. Вони використовуються для збереження рахунків класів, рахунків токенів, загальної кількості документів, словника слів та об'єкту стеммера.

```
class NaiveBayesClassifier:
    def __init__(self):
        self.class_counts = defaultdict(int)
        self.token_counts = defaultdict(lambda: defaultdict(int))
        self.total_documents = 0
        self.vocab = set()
        self.stemmer = PorterStemmer()
```

Рисунок 10 – Ініціалізація змінних класу NaiveBayesClassifier

Далі для тренування класифікатора прописую функцію train(), яка приймає текст(text) та відповідні мітки(labels) в якості вхідних даних для навчання класифікатора. Для кожного документа та відповідної мітки виконуються наступні дії:

- Збільшується лічильник класу для відповідної мітки.
- Збільшується загальна кількість документів.
- Текст документа токенізується за допомогою методу tokenize(), а потім для кожного токена виконуються наступні дії:
    - Збільшується лічильник токена для відповідної мітки.
    - Додається токен до словника слів.

Цей процес навчання дозволяє класифікатору зібрати статистику про кількість документів у кожному класі, кількість токенів у кожному класі та загальну кількість документів.

```
def train(self, text, labels):
    for document, label in zip(text, labels):
        self.class_counts[label] += 1
        self.total_documents += 1

        tokens = self.tokenize(document)
        for token in tokens:
            self.token_counts[token][label] += 1
            self.vocab.add(token)
```

Рисунок 11 – Програмна реалізація функції train()

Після тренування йде процес класифікації для якого необхідно спершу прописати функції calculate_score() та tokenize().

Функція calculate_score() приймає токени документа tokens та мітку класу label в якості вхідних даних і обчислює бал оцінки для цього документа та класу.

Ініціалізується змінна score зі значенням 0 та для кожного токена в документі виконується наступне:

- Отримується кількість входжень токена у вказаний клас (count_token_label).
- Отримується загальна кількість входжень токена у всі класи (count_token).
- Розраховується вірогідність токена у вказаному класі (prob_token_label), додаючи 1 до чисельника та додавши довжину словника слів до знаменника.
- Розраховується вірогідність токена у всіх класах (prob_token), додаючи 1 до чисельника та додавши загальну кількість документів до знаменника.
- До score додається різниця між логарифмом prob_token_label та логарифмом prob_token.
- Повертається отриманий бал оцінки.

```
def calculate_score(self, tokens, label):
    score = 0

    for token in tokens:
        count_token_label = self.token_counts[token][label]
        count_label = self.class_counts[label]
        prob_token_label = (count_token_label + 1) / (count_label + len(self.vocab))

        count_token = sum(self.token_counts[token].values())
        prob_token = (count_token + 1) / (self.total_documents + len(self.vocab))

        score += math.log(prob_token_label) - math.log(prob_token)

    return score
```

Рисунок 12 – Програмна реалізація функції calculate_score()

Функція tokenize() приймає документ document в якості вхідних даних і розбиває його на токени. Виконуються наступні кроки:

- Ініціалізується змінна stop_words зі списком стоп-слів англійської мови.
- Документ перетворюється до нижнього регістру і розбивається на слова за допомогою регулярного виразу.
- Відфільтровуються стоп-слова

```
    def tokenize(self, document):
        stop_words = set(stopwords.words('english'))
        tokens = re.findall(r'\w+', document.lower())
        tokens = [token for token in tokens if token not in stop_words]
        tokens = [self.stemmer.stem(token) for token in tokens]  # Stemming
        return tokens
```

Рисунок 13 – Програмна реалізація функції tokenize()

Після цього переходимо безпосередньо до класифікації з функцією classify(). Вона приймає документ document в якості вхідних даних і виконує класифікацію цього документа. Виконуються такі кроки:

- Документ токенізується за допомогою методу tokenize().
- Для кожного класу розраховується бал оцінки за допомогою методу calculate_score(). Бал оцінки обчислюється на основі логарифму вірогідності токенів у класі та логарифму вірогідності токенів у всіх класах.
- Повертається мітка класу з найвищим балом оцінки.

```
    def classify(self, document):
        tokens = self.tokenize(document)

        scores = {}
        for label in self.class_counts:
            score = self.calculate_score(tokens, label)
            scores[label] = score

        return max(scores, key=scores.get)
```

Рисунок 14 – Програмна реалізація функції classify()

2.3.2. Тренування Логістичної Регресії

Починаю з визначення функції preprocess_text для попередньої обробки текстових даних, реалізація якої зображена на рис.8.

З допомогою функції train_test_split(), яка є частиною модуля sklearn.model_selection ділю дані на навчальні та тестові набори. Використовую 80% даних для навчання, а решту 20% для тренування. Встановлює випадкове початкове число для забезпечення відтворюваності. Це дозволяє отримувати той самий розподіл кожного разу, коли запускається код.

```
X_train, X_test, y_train, y_test = train_test_split(df['preprocessed_tweet'],
                                   df['label'], test_size=0.2, random_state=42)
```

Рисунок 15 – Програмна реалізація поділу даних на навчальний та тестовий набори

Далі використовую бібліотеку TfidfVectorizer з модуля sklearn.feature_extraction.text, щоб виконати векторизацію тексту за допомогою методу TF-IDF (Term Frequency-Inverse Document Frequency).

TF-IDF є методом вагового коефіцієнта, який використовується для визначення важливості термів (слів) у колекції документів. Він широко застосовується у векторній моделі подання документів для обробки тексту і інформаційного пошуку.

TF (Частота термів) відображає, як часто термін з'являється в документі. Це просто кількість входжень терма у документ.

IDF (Обернена частота документів) відображає, наскільки рідкісним є термін у колекції документів. Вона обчислюється як логарифм зворотної частоти входження терма в документи колекції.

TF-IDF обчислюється шляхом множення значень TF і IDF для кожного терма. Це дає більш вагоме значення тим термам, які зустрічаються часто у документі, але не часто входять в інші документи колекції. Значення TF-IDF відображає важливість терміну для конкретного документа в контексті колекції.

TF-IDF може бути використаний для векторизації тексту, де кожен документ представляється вектором, де кожна компонента вектора відповідає значенню TF-IDF для відповідного терма. Це дозволяє використовувати числові ознаки для навчання моделей машинного навчання для класифікації тексту, кластеризації, пошуку інформації та багатьох інших завдань обробки тексту.

Отже, створюється об'єкт векторизатора TF-IDF TfidfVectorizer(). Він дозволяє перетворити текстові дані на числові вектори, які можуть бути використані для навчання моделі машинного навчання.

Функція fit_transform(X_train) виконує дві дії: навчання векторизатора і перетворення тренувального набору тексту X_train на матрицю ознак X_train_features. Під час навчання векторизатора він аналізує тренувальний текст,

обчислює значення TF-IDF для кожного слова і будує словник унікальних слів (термів). Потім кожен текст перетворюється на вектор, де кожна компонента вектору представляє значення TF-IDF для відповідного терму.

Функція transform(X_test) перетворює тестовий набір даних X_test на матрицю ознак X_test_features, використовуючи вже навчені значення TF-IDF та словник, визначені під час навчання векторизатора. Таким чином, векторизатор застосовує ті ж самі перетворення до тестового тексту, які були виконані для тренувального тексту.

Отримані X_train_features і X_test_features є матрицями ознак, де кожен рядок представляє вектор TF-IDF для відповідного тексту з тренувального і тестового набору відповідно. Ці ознаки потрібні для навчання моделі машинного навчання, такої як логістична регресія та для класифікації або інших завдань обробки тексту.

```
vectorizer = TfidfVectorizer()
X_train_features = vectorizer.fit_transform(X_train)
X_test_features = vectorizer.transform(X_test)
```

Рисунок 16 – Програмна реалізація векторизації тексту

Після цього перетворюю розріджену матрицю ознак X_train_features у масив за допомогою методу toarray(). Це потрібно для подальшого використання матриці в моделі машинного навчання.

```
X_train_features = X_train_features.toarray()
X_test_features = X_test_features.toarray()
```

Рисунок 17 – Програмна реалізація перетворення розрідженої матриці у масив

Оголошую функцію sigmoid(), яка використовується для обчислення значення логістичної функції. Ця функція використовується для класифікації нових текстових даних.

```
def sigmoid(z):
    return 1 / (1 + np.exp(-z))
```

Рисунок 18 – Програмна реалізація функції sigmoid()

2.3.3. Тестування роботи алгоритмів

Для роботи обох алгоритмів дані ділилися на навчальні(80%) та тестові(20%) набори. Отож, після тренування я отримала такі результати точності роботи алгоритмів:

Наївний Баєсів Класифікатор - 0.7953074433656958

Логістична Регресія - 0.819848975188781

Як бачимо, Логістична Регресія працює краще ніж Наївний Баєсів Класифікатор.

Також хочу навести приклади класифікації нового тексту – рис. 19.

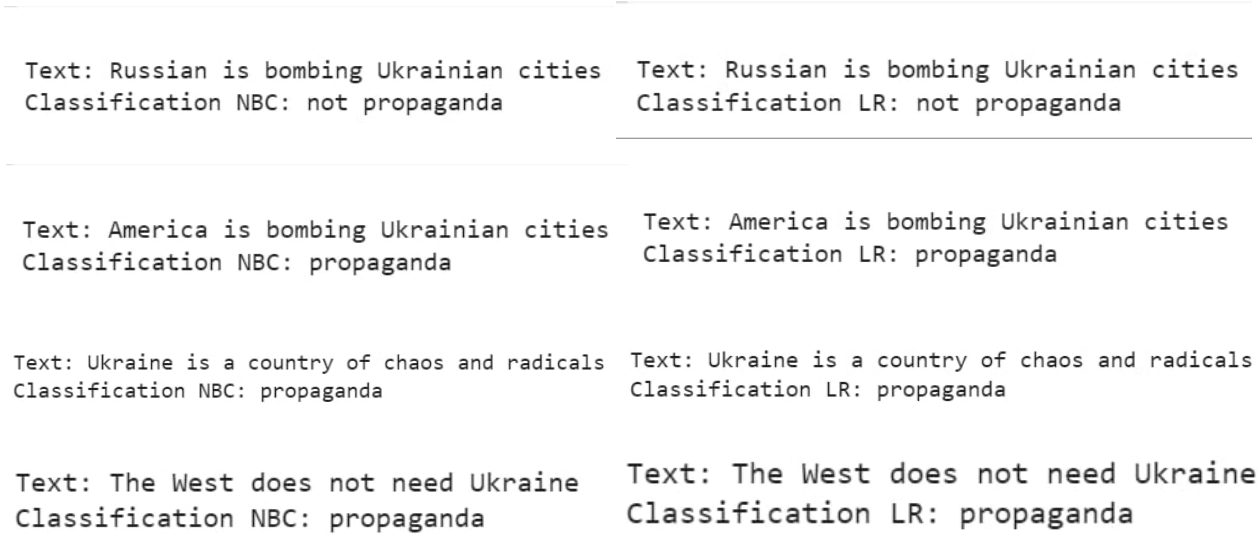

Рисунок 19 – Приклад класифікації нового тексту

2.3.4. Візуалізація роботи

Для обрахунку формули Баєса необхідні такі складові як апріорні ймовірності, ймовірність розподілу ознак та апостеріорна ймовірність, графіки яких зображені на рис. [20 – 22].

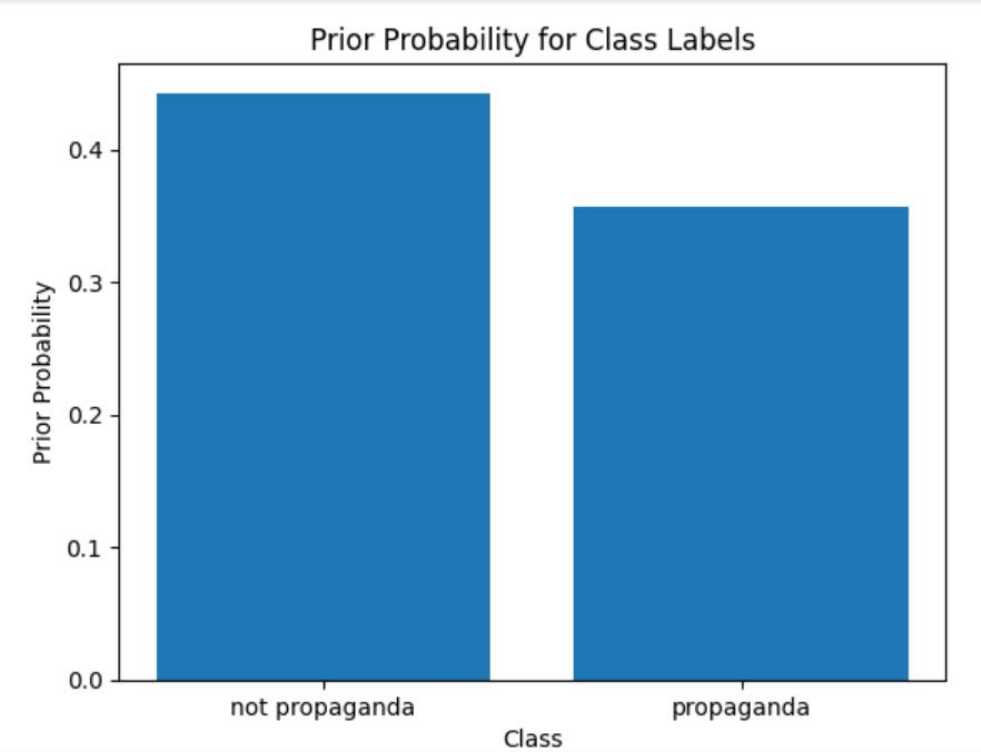

Рисунок 20 – Графік апріорних ймовірностей для кожного класу

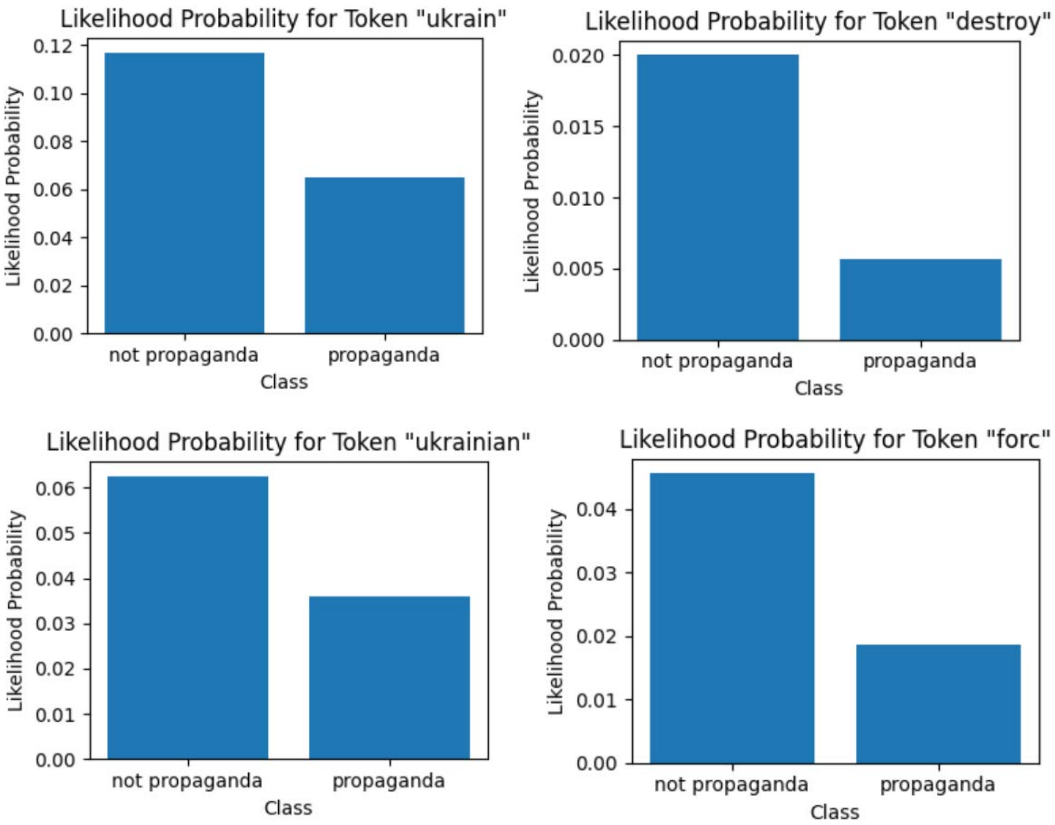

Рисунок 21 – Графіки ймовірності розподілу ознак

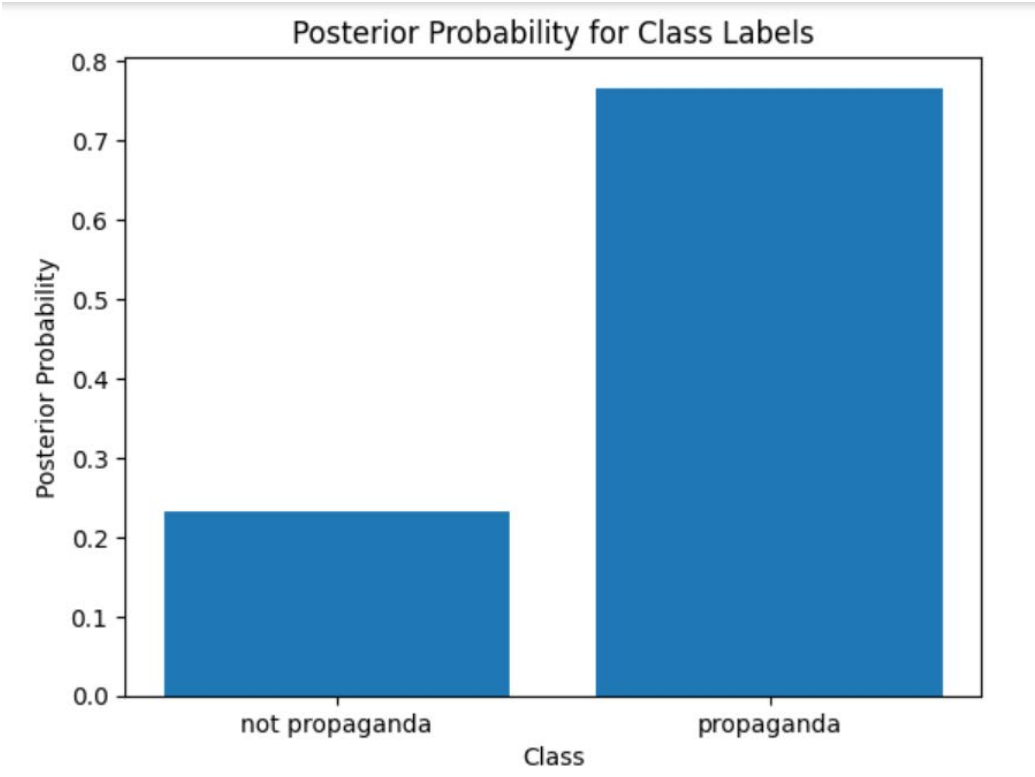

Рисунок 22 – Графіки апостеріорних ймовірностей

Для швидкої оцінки розподілу даних між класами було створено кругову діаграму зображену на рис. 23.

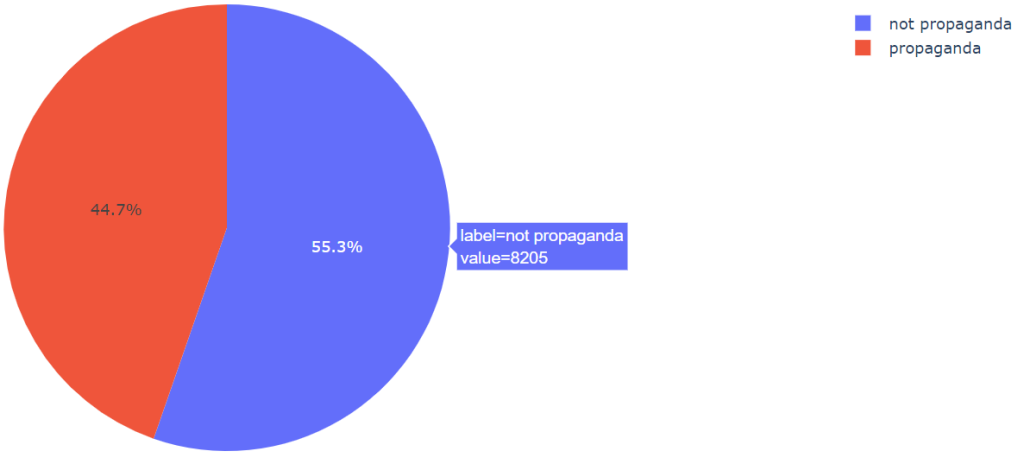

Рисунок 23 – Діаграма розподілу даних

Щоб швидко визначити які токени найчастіше зустрічаються в оброблених даних було створено «хмару слів» - рис. 24.

Рисунок 24 – Візуалізація хмари слів

Зображений на рис. 25 графік показує проміжний етап аналізу емоційної забарвленості тексту з використанням алгоритму Логістична регресія. На цьому графіку відображені ймовірності класифікації твітів, вибраних випадковим чином з тестового набору даних.

Кожен стовпець показує ймовірність належності відповідного твіту до певного класу ("пропаганда" або "не пропаганда"). Горизонтальна червона лінія, яка перетинає графік, відображає порогове значення ймовірності (0.5). Твіти, які мають ймовірність класифікації нижче цього порогу, вважаються належними до класу not propaganda, а ті, що мають ймовірність вище порогу, до класу propaganda.

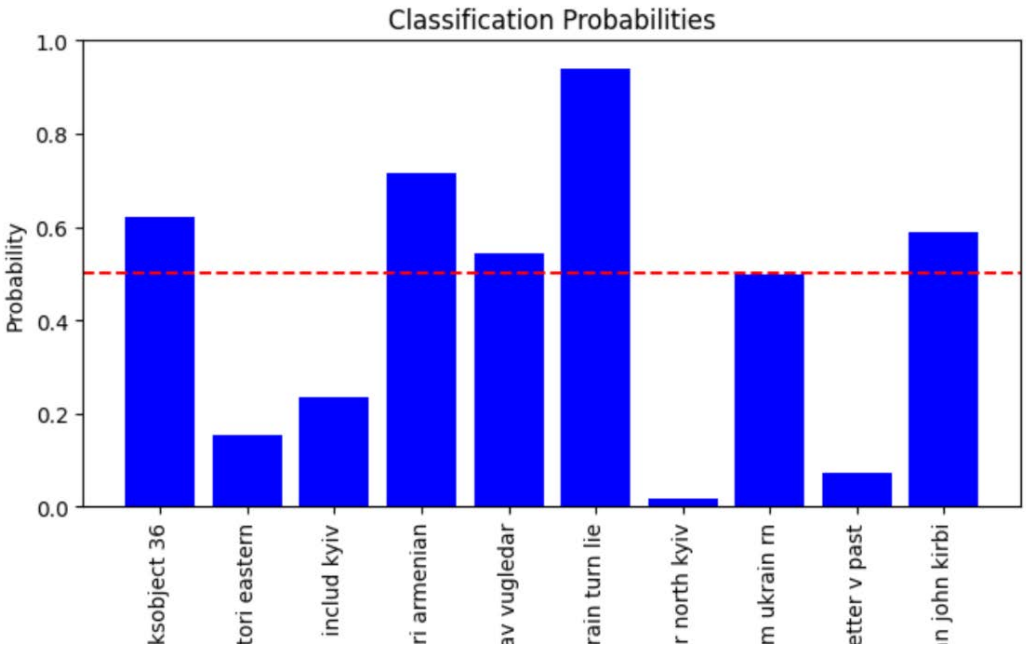

Рисунок 25 – графік ймовірностей класифікації вибраних випадковим чином твітів

ВИСНОВКИ

Аналіз емоційного забарвлення тексту є активною галуззю досліджень в аналізі даних, пошуку інформації та веб-інтелекту. Він знаходить застосування у багатьох сферах.

Тема боротьби з російською пропагандою актуальна вже десятки років і, на жаль, зараз відіграє особливо важливу роль в нашому житті. Саме по цій причині темою для дослідження було обрано аналіз емоційного забарвлення тексту з метою пошуку російської пропаганди.

У цій праці порівняно ефективність роботи таких двох алгоритмів як Наївний Баєсів Класифікатор та Логістична Регресія. Результати показали, що обидва алгоритми непогано підходять для виконання поставленого завдання, проте алгоритм Логістична Регресія для аналізу великого об’єму даних є ефективнішим. Такі висновки були зроблені на основі оцінки точності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Беккауер А.О. Використання технологій data mining для автоматизації бізнес-процесів на виробництві// Моделювання в економіці, організація виробництва та управління проектами. –2016.– Вип. 1(138).–С. 161-164
2. Бойко, Н. І., and О. В. Лукаш. "Сучасні тенденції процесу вилучення відкритої інформації з веб-сайтів інструментами web scraping."
3. Мельничук, Я. О., and С. М. Кравченко. "АНАЛІЗ ДАНИХ ТА ВІЗУАЛІЗАЦІЯ ЗА ДОПОМОГОЮ МОВИ PYTHON."
4. Могильний С. Б. Машинне навчання з використанням мікрокомп’ютерів: навч.-метод. посіб. / за ред. О. В. Лісового та ін. Київ, 2019. 226с.
5. Петренко, А. І. "Grid та інтелектуальна обробка даних Data Mining."
6. Ткаченко, Олександра Олексіївна, Олена Володимирівна Олійник. "МОЖЛИВОСТІ ТА ТРУДНОЩІ ВИКОРИСТАННЯ ОБРОБКИ ПРИРОДНОЇ МОВИ." Практичні та теоретичні питання розвитку науки та освіти (частина І): матеріали ІІ Міжнародної науково-практичної конференції м. Львів, 19-20 грудня 2020 року.–Львів: Львівський науковий форум, 2020.–74 с.: 73.
7. Agresti, A. (2013). "Categorical Data Analysis. Wiley." (стор. 142-171)
8. Aurélien Géron "Hands-On Machine Learning with Scikit-Learn and TensorFlow"
9. Bing Liu "Sentiment Analysis and Opinion Mining"
10. Bishop, C. M. (2006). "Pattern Recognition and Machine Learning."
11. Carson Sievert, "Interactive web-based data visualization with R, plotly, and shiny", 2019. – 208p.
12. Hastie, T., Tibshirani, R., & Friedman, J. (2009). "The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer." (стор. 119-132)
13. Ian Goodfellow, Yoshua Bengio, Aaron Courville "Deep Learning"
14. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). "An Introduction to Statistical Learning: with Applications in R. Springer." (стор. 131-151)
15. Larose D.T. Discovering knowledge in data: an introduction to data mining. – NewJersey: JohnWiley&Sons, Inc., 2005. – 240 p.
16. Steven Bird, Ewan Klein, Edward Loper "Natural Language Processing with Python"

17. Waltman, L., Van Eck, N.J., & Noyons, E.C.M. (2010). A unified approach to mapping and clustering of bibliometric networks. Journal of Informetrics, 4(4), 629-635.
18. https://uk.wikipedia.org/wiki/Kaggle
19. https://www.kaggle.com/datasets/dariusalexandru/russian-propaganda-tweets-vs-western-tweets-war?resource=download&select=russian_propaganda_tweets.csv
20. https://www.kaggle.com/datasets/dariusalexandru/russian-propaganda-tweets-vs-western-tweets-war?resource=download&select=western_analysts_tweets.csv