Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки та захисту інформації
_____ Іван ПАРХОМЕНКО
«____» червня 2023 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)
спеціальність 125 Кібербезпека
(код і назва спеціальності)
освітній ступень бакалавр
освітня програма Кібербезпека
(назва освітньо-професійної програми)
на тему: «Метод ідентифікації ворожих медіа-ресурсів в кіберпросторі»

Виконавець: студент IV курсу, групи КБ-41

_____ **Вадим БЕЗДОЛЬНИЙ**
(підпис) (ім'я, прізвище)

	Ім'я, прізвище	**Підпис**
Керівник	Лариса МИРУТЕНКО	

Нормоконтроль	Андрій ФЕСЕНКО	

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки та захисту інформації
_____ Сергій ТОЛЮПА
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньо-професійної програми)

Студенту **КБ-41** (група) **Бездольному Вадиму Андрійовичу** (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Метод ідентифікації ворожих медіа-ресурсів в кіберпросторі

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №3 від 20.10.2022 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Програмне забезпечення WeKa для тренування моделей нейронних мереж, середовище розробки PyCharm для написання скриптів на мові Python, для створення датасетів, теорія створення моделей нейронних мереж.

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Теоретичні підходи до аналізу текстового контенту, підготовка даних для навчання, створення словників для обчислення характеристик тексту, розробка інтелектуальної моделі ідентифікації текстового контенту.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Розробка інтелектуальної моделі ідентифікації ворожих медіа-ресурсів

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 24 жовтня 2022 року

Завдання видала _____ (підпис) Лариса МИРУТЕНКО (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Вадим БЕЗДОЛЬНИЙ (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	24.10.2022 – 22.01.2023	*виконано*
2	Аналіз літератури	29.01.2023 – 20.02.2023	*виконано*
3	Вибір архітектури нейронної мережі	24.02.2023 – 04.03.2022	*виконано*
4	Дослідження характеристик тексту та доцільності їх використання	05.03.2023 – 24.03.2023	*виконано*
5	Створення навчальної та тестової вибірок, пошук текстових джерел	25.03.2023 – 07.04.2023	*виконано*
6	Дослідження процесу попередньої обробки тексту	07.04.2023 – 16.04.2023	*виконано*
7	Формальне визначення моделі та вибір класифікатора	16.04.2023 – 20.04.2023	*виконано*
8	Реалізація та тестування моделі	20.04.2023 – 10.05.2023	*виконано*
9	Оформлення пояснювальної записки	11.05.2023 – 27.05.2023	*виконано*

Завдання видала _____ (підпис) Лариса МИРУТЕНКО (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Вадим БЕЗДОЛЬНИЙ (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 12 червня 2023 року

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи складається зі вступу, трьох розділів, загальних висновків, списку джерел та додатків. Основний текст займає 72 сторінки, включає в себе зміст, вступ, три розділи кваліфікаційної роботи, висновки та список джерел, що містить 26 найменувань та займає 4 сторінки. Крім того, робота містить 3 додатки із загальною кількістю сторінок 11, кількість рисунків дорівнює 55, літературних джерел 26.

Метою роботи є розробка моделі ідентифікації ворожих медіа-ресурсів шляхом використання нейронних мереж.

Об’єктом дослідження є процес класифікації і кластеризації економічно та політично орієнтованого контенту за допомогою нейронної мережі.

Предметом дослідження є моделі класифікації економічно та політично орієнтованого медіа-контенту, виконані за допомогою апарату нейронних мереж на основі класифікатора K-Star.

Методи дослідження:

- аналіз відкритих джерел;
- порівняння;
- імітаційне моделювання.

Практичною цінністю є розробка інтелектуальної моделі ідентифікації ворожих медіа-ресурсів.

Ключові слова: ідентифікація, кіберпростір, кіберзахист, моделювання, нейронна мережа, кластеризація.

Відповідно до мети роботи, були поставлені наступні завдання:

- синтез інтелектуальної моделі мережі;
- збір та обчислення характеристик текстового контенту;
- розробка алгоритму попередньої підготовки даних;
- синтез моделі нейронної мережі та її навчання;
- перевірка роботи мережі і аналіз її інформативності.

ЗМІСТ

ВСТУП

Інформаційна безпека – стан захищеності життєво важливих інтересів людини і громадянина, суспільства і держави, при якому запобігається завдання шкоди через неповноту, несвоєчасність та недостовірність поширюваної інформації, порушення цілісності та доступності інформації, несанкціонований обіг інформації з обмеженим доступом, а також через негативний інформаційно-психологічний вплив та умисне спричинення негативних наслідків застосування інформаційних технологій [1]. Легкодоступність та масовість інформаційних джерел призводить до різкого збільшення потенційних шкідливих інформаційних впливів. Шкідливі інформаційні впливи використовують ударний інформаційний контент для підсилення або послаблення переконань людини. Наразі не існує системи класифікації контенту, яка б автоматизовано вирішувала проблему інформаційної безпеки людини. Слід зазначити, що дане дослідження зосереджено на економічно та політично орієнтованому контенті.

Наукові роботи, що стосувалися теми класифікації пропагандистського контенту, присутні в недостатньому об’ємі для побудови на них власної методики. Попередні дослідження стосовно виявлення пропаганди в тексті зосереджувались на задачі класифікації пропаганди та її різновидів [2-5], але рідко розглядали вирішення задачі автоматизації цього процесу. Деякі автори вирішували задачу класифікації текстів за допомогою використання алгоритмів машинного навчання з вчителем [6]. Тексти були розділені у два класи: пропагандистські та не пропагандистські. В якості текстового матеріалу був використаний контент з Твіттеру з хештегами #propaganda, #Hoaxes, #Falseflag тощо. Контент був попередньо оброблений використовуючи підрахунок частоти слів (term frequency, inverse document frequency) та bag of words. Ці параметри розглядались під час класифікації тексту за допомогою Support Vector Machine (SVM) та Multinomial Naïve Bayes. Fine tuning для SVM був виконаний через kernel Linear, Poly та RBF. В результаті, на даному датасеті SVM показав кращі результати ніж MNB з середньою точністю у 69.2%.

Апарат нейронних мереж широко використовується для задач класифікації [7], в тому числі для текстового контенту.

Нейронних мережі демонструють високу точність у вирішенні задачі аналізу мови (sentiment analysis task) на прикладі датасету з оглядами фільмів IMDB [8]. Між собою порівнювалася точність класифікації текстового матеріалу за допомогою Spiked Neural Network та Artificial Neural Network. В даному контексті важливо, що обидва підходи мали точність у 85.88% та 84.86% відповідно.

Для вирішення завдання категоризації тексту існують приклади розробки складної системи, що складалася з bidirectional cyclic and convolutional neural networks – BRCNN, cyclic neural network та convolution neural network [9]. Завданням мережі було правильно визначити категорію, до якої відносився текстовий контент. Продуктивність цієї моделі оцінювалась за допомогою 7 датасетів. Датасети складалися з різного типу текстового контенту: огляди фільмів, об'єктивних та суб'єктивних речень, помічені по рівню своєї емоційності. Точність роботи моделі могла досягати 96.3%, з середніми значеннями у 78.8-96.3%.

Враховуючи відносно високу точність класифікації, використання нейронних технологій можна вважати перспективним напрямком для створення системи класифікації тексту за наявністю пропагандистських матеріалів. В якості вхідних параметрів для мережі вказана робота використовує частоту появи певних слів у текстовому контенті. Можливо, використання інших характеристик могло б позитивно вплинути на точність результатів.

Для вирішення задачі класифікації можна використати двоетапний підхід [10]. Нейронна мережа класифікувала контент на форумах у два класи: пропаганда, не пропаганда. Спершу нейронна мережа навчається помічати текстовий контент певними флагами на створеною людиною датасеті. Після цього ця нейронна мережа повинна була визначити, чи є текст пропагандистським, вже використовуючи флаги, а не сам текстовий матеріал. Автори дійшли наступного висновку:

"Overall, metadata (particularly a low comment rating) are more predictive of perceived propaganda than textual features. The method can be extended to monitor

suspicious activity in other online environments, but the results should not be interpreted as detecting actual propaganda."

Цей висновок дозволяє припустити, що використання числових характеристик тексту, а не текстових флагів (textual features) буде більш доцільним для вирішення завдання класифікації контенту.

Деякі автори порівнювали емоційний відгук читачів від прочитаної статті, змінюючи на чому концентрувався її зміст [12]. Результат їх роботи дозволяє підтримати припущення про доцільність використання числових характеристик. У роботі, в якості варіантів однієї і тієї самої статті використовувалась персоніфікована історія, перемога над складними життєвими умовами (triumph over adversity) та фокус на статистичній інформації. Емоційний відгук перевірявся через бажання читача фінансово допомогти вирішенню проблеми, описаній у статті. Автори прийшли до висновку, що позитивний відгук не залежав від використання одної з технік:

"While Kristof clearly uses reporting methods intended to overcome compassion fatigue, the effect of these efforts for the most part is not apparent. None of the metrics studied showed a significant positive relationship with personification of story, triumph over adversity, or mobilizing information. The interplay of reader response with statistics also contradicted what experimental psychology posits; this study found that reader response positively corresponds with quantitative information, though the results generally were not statistically conclusive."

Сприйняття статті не залежало від стилю її написання, аналогічно тому як комп'ютер здатен аналізувати контент, використовуючи його ключові слова, а не текст в цілому [13].

Результати наукових робіт з класифікації текстових даних з використанням апарату нейронних мереж є відносно точним. Використання нейронних мереж для класифікації тексту також є більш продуктивним ніж класифікація тексту стандартними засобами програмування. Ці результати, а також результати вказаних попередньо досліджень, дозволяють вказати, що саме використання апарату нейронних мереж буде найбільш доцільним для вирішення поставленого завдання.

Метою роботи є розробка моделі ідентифікації ворожих медіа-ресурсів шляхом використання нейронних мереж.

Об’єктом дослідження є процес класифікації і кластеризації економічно та політично орієнтованого контенту за допомогою інтелектуальних моделей.

Предметом дослідження є моделі класифікації економічно та політично орієнтованого медіа-контенту, виконані за допомогою апарату нейронних мереж на основі класифікатора K-Star.

Методи дослідження:

- аналіз відкритих джерел;
- порівняння;
- синтез;
- імітаційне моделювання.

Практичною цінністю є розробка інтелектуальної моделі ідентифікації ворожих медіа-ресурсів.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ПЕРЕДУМОВИ ДЛЯ СТВОРЕННЯ НЕЙРОННОЇ МЕРЕЖІ

1.1 Огляд парадигм машинного навчання та його алгоритмів

Задача класифікації текстового контенту є спільною задачею таких галузей, як Artificial Intelligence, Machine Learning, Data Science та Natural Language Processing (NLP). Важливо розуміти різницю між трьома галузями:

- Штучний інтелект (AI) – це загальна категорія технологій, які дозволяють комп'ютерам робити розумові операції, які раніше вважалися виключно людською прерогативою. AI може включати в себе такі різні техніки, як машинне навчання, глибоке навчання, обробку природної мови, комп'ютерне зорове сприйняття та інші.
- Машинне навчання (Machine Learning) – це конкретна підгалузь AI, яка дозволяє комп'ютерам вчитися на основі даних, замість того, щоб програмувати їх наперед. Комп'ютерний алгоритм навчається на прикладах і даних, здатних зробити прогнози або приймати рішення в майбутньому. Машинне навчання використовується для розв'язання складних задач, таких як розпізнавання образів, розпізнавання мови, рекомендації товарів і послуг, прогнозування погоди та інше.
- Data Science – це наука, що досліджує, як отримати інформацію з даних, зокрема, як обробляти, аналізувати, визначати закономірності і робити висновки на основі даних. Data Science також включає розробку алгоритмів та інструментів для обробки даних, щоб зробити їх доступними для машинного навчання та інші застосування.
- Natural Language Processing (NLP) або Обробка природньої мови – це галузь комп'ютерних наук, яка займається розробкою методів та алгоритмів для обробки та аналізу людської мови в її природному вигляді. Це охоплює різні аспекти мови, такі як розпізнавання та синтез мовлення, семантичний аналіз, машинний переклад, класифікація текстів, виявлення емоцій та інше.

В залежності від призначення нейронної мережі, використовуються різні методи машинного навчання.

Рисунок 1.1 – Існуючі методи машинного навчання.

- Навчання з вчителем (Supervised Learning) – це метод навчання машинного навчання, в якому маємо вхідні дані (вектори або таблиці) та відповідні вихідні значення (мітки). Метою є навчити алгоритм прогнозувати вихідні значення на основі вхідних даних. Наприклад, при класифікації зображень, вихідним значенням може бути тег, який позначає клас зображення.
- Навчання без вчителя (Unsupervised Learning) – це метод машинного навчання, в якому немає міток для вихідних даних. Метою є відкриття внутрішньої структури даних, знайти закономірності та групування. Наприклад, при кластеризації, метою є групування схожих об'єктів.
- Навчання з підкріпленням (Reinforcement Learning) – це метод машинного навчання, в якому агент взаємодіє з середовищем, отримуючи від нього нагороди або покарання, залежно від своїх дій. Метою є навчити агента знаходити оптимальну стратегію дій для максимізації нагороди. Наприклад, при навчанні комп'ютерної

програми, яка грає в гру, метою є навчити її здійснювати оптимальні дії для максимізації очок.

- Напівавтоматичне навчання (Semi-Supervised Learning) – це метод машинного навчання, який поєднує в собі навчання з вчителем (Supervised Learning) та навчання без вчителя (Unsupervised Learning). У цьому методі лише частина вхідних даних має мітки, тоді як інша частина не має. Метою цього методу є використовувати інформацію з мітками, щоб навчити алгоритм, а потім використовувати цей алгоритм для передбачення значень на даних без міток.

Для вирішення задачі класифікації в даній роботі найбільше підходить метод навчання з вчителем. Для створення навчальної вибірки використовується додаток, написаний на мові Python, що формує датасет з числовими характеристиками тексту. Вони, як і додаток, будуть розглянуті далі.

На рисунку 1.2 наведена детальна інформація по кожному з існуючих алгоритмів побудови моделей машинного навчання і їх роботи.

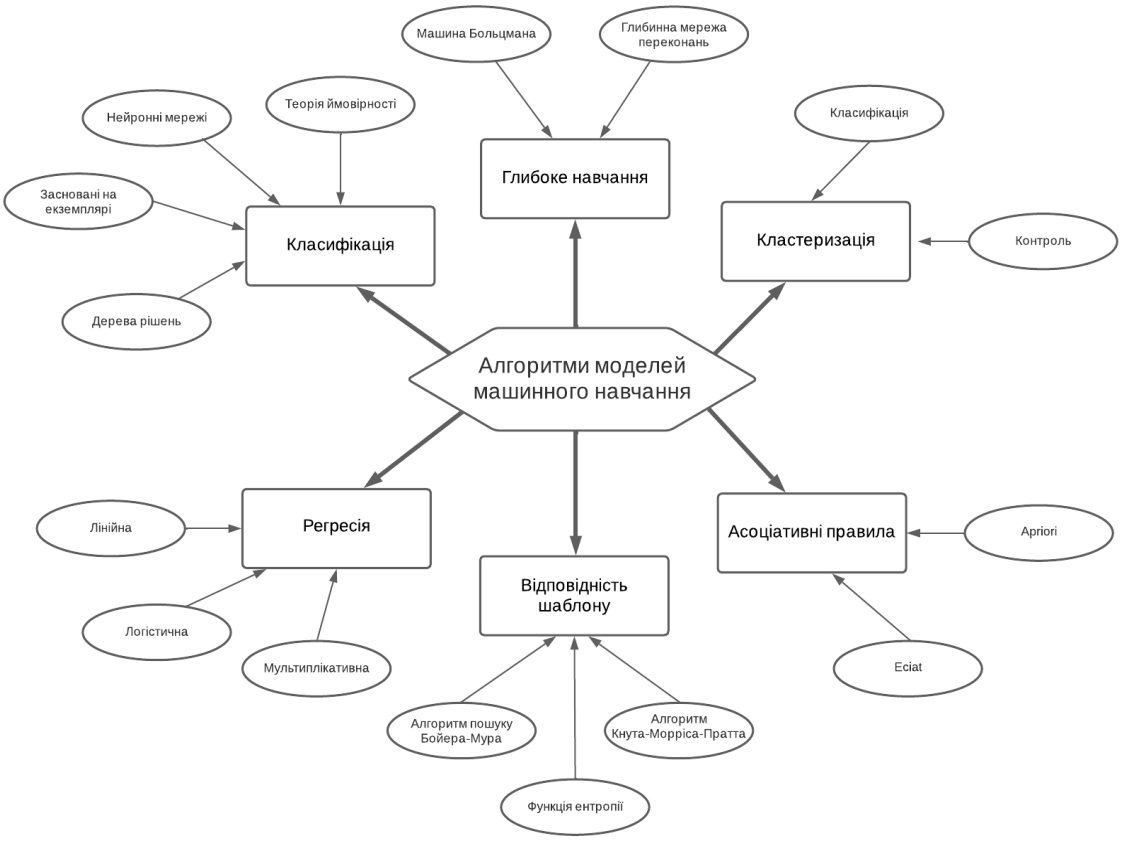

Рисунок 1.2 – Існуючі алгоритми моделей машинного навчання.

- Регресія – використовується для прогнозування числових значень. Наприклад, цей алгоритм може використовуватися для прогнозування ціни на нерухомість, враховуючи її параметри.
- Кластеризація – використовується для групування схожих об'єктів в класи. Наприклад, цей алгоритм може використовуватися для аналізу споживацьких звичок та групування споживачів за схожими характеристиками.
- Дерева рішень – використовується для класифікації та прогнозування подій на основі вирішення послідовності рішень. Наприклад, цей алгоритм може використовуватися для класифікації клієнтів банку на основі їхніх фінансових характеристик та інших даних.
- Наївний Байєс – використовується для класифікації текстової інформації, такої як електронні листи, на основі ймовірності того, що деякі слова з'являться у тексті. Наприклад, цей алгоритм може використовуватися для фільтрації спаму в електронній пошті.
- Метод опорних векторів – використовується для класифікації об'єктів на два або більше класів. Наприклад, цей алгоритм може використовуватися для класифікації зображень на основі їхніх характеристик.

На рис. 1.3 зображений процес реалізації нейронної мережі.

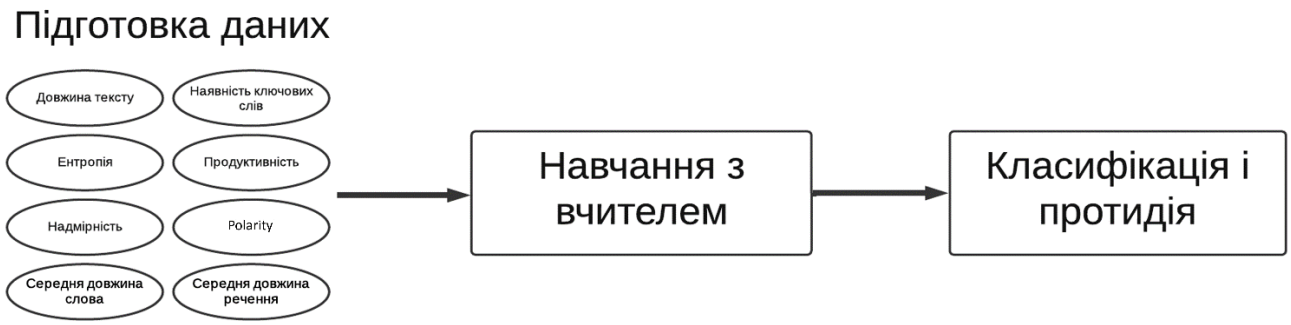

Рисунок 1.3 – Базова модель побудови системи ідентифікації контенту.

1.2 Метрики програмного забезпечення WeKa для оцінки функціонування нейронної мережі

У програмному застосунку WeKa є декілька метрик для візуалізації характеристик мережі.

- MarginCurve – це інструмент для оцінки якості багатокласового класифікатора. Він створює точки, що ілюструють похибку прогнозу. Похибка визначається як різниця між ймовірністю, передбаченою для фактичного класу, і найвищою ймовірністю, передбаченою для інших класів. Одна з гіпотез, яка пояснює високу ефективність алгоритмів бустінгу, полягає в тому, що вони збільшують маржу на навчальних даних, і це дає кращу продуктивність на тестових даних. MarginCurve побудована за наступним алгоритмом: спочатку всі екземпляри сортуються за зростанням маржиналів і створюється список екземплярів, який відсортований за їх оцінкою. За допомогою порогового значення (threshold) з лівої до правої частини списку класифікуються всі екземпляри. На осі X відкладається відношення кількості позитивних (дійсно позитивних та помилково позитивних) зразків до загальної кількості позитивних зразків, а на осі Y відкладається відношення кількості негативних (дійсно негативних та помилково негативних) зразків до загальної кількості негативних зразків. Отримана крива дозволяє зрозуміти, який поріг найбільш оптимальний для класифікації, тобто як можна отримати максимально точний результат при мінімізації кількості помилок. MarginCurve може бути корисним для знаходження порогового значення, яке забезпечує оптимальний баланс між точністю та повнотою класифікатора K-Star.
- ThresholdCurve – це графік, що демонструє взаємозв'язок між точністю і повнотою моделі класифікації на різних порогових значеннях. Він генерує точки, що ілюструють компроміси в прогнозуванні, які можна отримати, змінюючи порогове значення між класами. Наприклад, типове порогове значення 0.5 означає, що прогнозована ймовірність “позитивного” результату має бути вищою за 0.5 для того, аби приклад був визнаний “позитивним”. Отриманий набір даних можна використовувати для візуалізації компромісу між точністю і пригадуванням (recall)

або для аналізу ROC-Curve (співвідношення істинних і хибнопозитивних результатів). Weka просто змінює поріг для оцінок ймовірності класу в кожному випадку. Для розрахунку AUC використовується статистика Манна-Уітні. У класифікатора K-Star є параметр порогу (threshold), який визначає, який клас повинен бути визнаний як позитивний, а який як негативний. ThresholdCurve дозволяє оцінити вплив цього порогового значення на точність і полноту класифікації. Вона побудована наступним чином: спочатку вибирається серія порогових значень, потім для кожного з них обчислюється точність і полнота моделі. Отримані значення відображаються на графіку, який демонструє, як точність і полнота змінюються в залежності від порогового значення. Цей графік дозволяє вибрати оптимальне порогове значення для класифікатора K-Star, яке забезпечить максимальну точність і полноту.

- Cost/Benefit Analysis – це метод, який використовується для оцінки правильності роботи класифікатора, він візуалізує кількість витрат для досягнення певного відсотку точності моделі. Кожен класифікатор описує кожен окремий набір даних нерівномірно, і деякі класифікатори будуть описувати дані більш точно. Метод дозволяє порівнювати результати класифікації між собою та обирати найбільш правильний метод класифікації.
- CostCurve – це метод, що генерує точки, що ілюструють різницю у точності (probability cost tradeoffs), яку можна отримати, змінюючи порогове значення (threshold) між класами. Наприклад, типове порогове значення 0.5 означає, що прогнозована ймовірність “позитивного” результату має бути вищою за 0.5 для того, щоб випадок був визнаний “позитивним”. Для класифікатора K-Star CostCurve можна використовувати для оцінки витрат, пов'язаних з визначенням оптимального порогу відсічення для прийняття рішення про класифікацію. Крива відображає витрати (наприклад, час, гроші тощо) в залежності від відсічення рішень на різних рівнях точності і повноти. Вона може допомогти знайти баланс між точністю і наданій мережі кількістю даних.

1.3 Числові характеристики тексту та оцінка доцільності їх використання

Кількість характеристик тексту, які є доцільно використовувати для побудови моделі, обмежена. Самі характеристики тексту різняться в залежності від використання контенту. Наприклад, Search Engine Optimization – це процес оптимізації веб-сайту для поліпшення його рейтингу та позиції в результатах пошукових систем, таких як Google, Bing, Yahoo і т.д. У цій галузі контент характеризується наступними параметрами.

- Унікальність – наскільки даний текст повторює або не повторює тексти на аналогічну тему.
- Наявність стоп-слів – частота появи слів, які не несуть смислового навантаження (сполучники або займенники).
- Довжина тексту.
- Інформативність – характеризує, наскільки текст повно розкриває задану тему.
- Наявність ключових слів – частота появи слів, які оброблюються пошуковими системи, вони впливають на знаходження саме цього тексту через відповідний запит.

В теорії інформації для характеристики дискретного джерела інформації використовуються наведені нижче характеристики.

- Ентропія – міра невизначеності ситуації. Чим вона більша, тим інформативність повідомлення більша. Ентропія досягає максимального значення, коли ймовірності появи кожного повідомлення з множини повідомлень дорівнюють одна одній.
- Продуктивність – відношення ентропії до середньої тривалості символу.
- Надмірність – дає відносну оцінку використання потенційних можливостей джерела з алфавітом заданої потужності. Алфавіт будь–якої людської мови має деяку надмірність, через що не кожен символ у слові є однаково інформативним. Надмірність може приймати значення від 0 до 1.

Галузь Natural Language Processing використовує subjectivity, polarity, а також статистичні характеристики тексту [13].

- Subjectivity – характеристика визначає об’єктивність написаного. Об’єктивні факти не несуть емоційного забарвлення, тому не мають корисної інформації для класифікації тексту.
- Polarity – характеристика визначає емоційне забарвлення написаного, від повністю негативного до повністю позитивного (від -1 до 1).
- Довжина тексту.
- Середня довжина слова.
- Середня довжина речення.

Не кожна з цих характеристик є оптимальною для розрахунку та/або навчання, тестування моделі.

Унікальність тексту вимірюється у порівнянні з наявним в мережі текстовим контентом, попередня обробка такої характеристик займає кількість ресурсів, що перевищує додаткову точність, яку з цієї характеристики можливо було б отримати.

Наявність стоп-слів є легко обчислювальною характеристикою. Проте, будь-який текст має стоп-слова, і відношення цих слів до довжини тексту можна вважати сталим. Дана характеристика не є корисною для вирішення поставленої задачі класифікації.

Характеристика під назвою “інформативність” не є комп’ютерно обчислювальною.

Продуктивність дискретного джерела оцінити неможливо через відсутність середньої тривалості символу.

Subjectivity не наявна в списку вхідних характеристик, тому що вона автоматично обчислюється при визначенні polarity, дана характеристика не є інформативною.

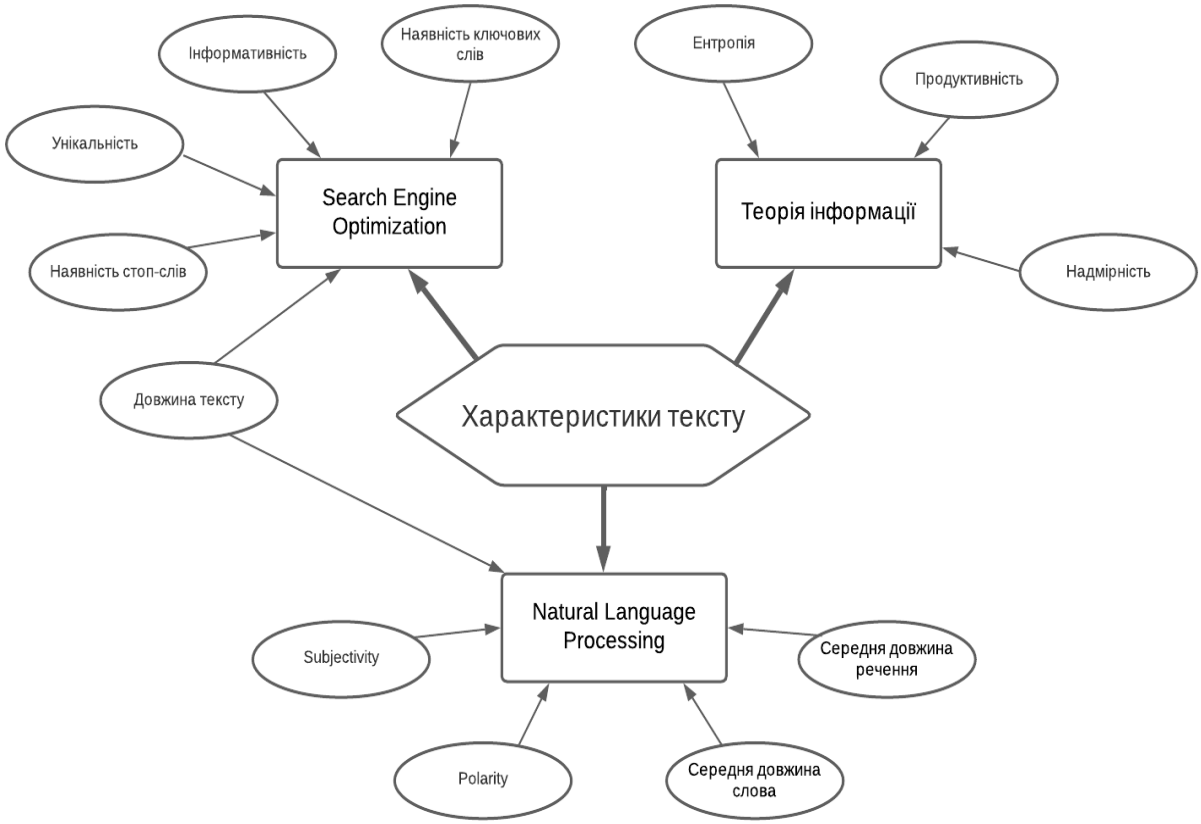

Рисунок 1.4 – Характеристики тексту.

Висновки до розділу 1

В даному розділі для вирішення задачі класифікації текстового контенту були розглянуті теоретичні засади машинного навчання та нейронних мереж. Були вказані методи машинного навчання, а саме: навчання з вчителем, навчання без вчителя, навчання з підкріпленням, напівавтоматичне навчання, що відрізняються між собою наявністю заздалегідь промаркованих людиною елементів.

Алгоритми машинного навчання визначають, як саме нейронна мережа будує зв’язки між елементами, це поняття слід відрізняти від методів машинного навчання. До алгоритмів машинного навчання належать регресія, кластеризація, дерева рішень, наївний Байєс, метод опорних векторів.

У програмному середовищі WeKa наявні власні характеристики, що описують нейронну мережу. Їх використовують для оцінки якості нейронної мережі та вхідних даних на яких вона побудована. До цих характеристик належать MarginCurve –

різниця між найбільш ймовірною та наступною за ймовірністю мітками, ThresholdCurve – характеризує точність моделі на різних порогових значеннях, Cost/Benefit Analysis – точність моделі по відношенню до кількості вхідних даних, CostCurve – ілюструює компроміси ймовірних витрат, які можна отримати, змінюючи граничне значення між класами.

Числові характеристики тексту існують в кожній сфері діяльності, і є доволі унікальними саме для неї. До таких характеристик відносяться унікальність, наявність стоп-слів, довжина тексту, інформативність, наявність ключових слів, ентропія, продуктивність, надмірність, subjectivity, polarity, середня довжина слова, середня довжина речення. Не всі з них є доречними при використанні у нейронній мережі. Питання доцільності їх використання розглянуто у розділі 2 кваліфікаційної роботи.

РОЗДІЛ 2
РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ МОДЕЛІ КЛАСИФІКАЦІЇ ТЕКСТОВОГО КОНТЕНТУ

2.1 Синтез моделі класифікації текстового контенту

В моделі визначено 8 входів, відповідно до кількості комп'ютерно обчислювальних характеристик тексту. Вони будуть розглянуті пізніше.

Кількість виходів дорівнює 1. На виході мережі ми повинні отримати одне з двох значень – 0 або 1, що відповідно означає "не маніпуляція", "маніпуляція".

Кількість шарів дорівнює 5.

Для побудови та оцінки інформативності моделі, яка займається класифікацією текстового матеріалу, необхідно зробити попередню підготовку вибірок даних для навчання, контролю та тестування.

Процес обробки даних про стан системи складається з 4-ох основних етапів.

1) Збір текстового матеріалу.
2) Обчислення характеристик текстового матеріалу та привласнення йому відповідного класу.
3) Формування таблиці з обчислених даних, з визначеним полем для класу.
4) Навчання та тестування нейронної мережі.
5) Аналіз отриманих результатів.

При створенні моделі використовується метод навчання з вчителем, адже по-перше, класи повинні бути визначені заздалегідь, по–друге, навчальна та тестова вибірка повинні бути попередньо класифіковані.

Для використання парадигми зі вчителем, вхідні дані повинні складатись з вектору незалежних змінних та значень, які має видавати модель після навчання. Навчальні та тестові записи мають вигляд: $\{(x_1,y_1),...([x_N,y_N])\}$, де x_i – вектор вхідних ознак і-го запису, y_i – цільове значення і-го запису.

Обраною моделлю зумовлене те, що набір вхідних та вихідних характеристик повинен бути чисельним типом даних. Визначаємо, що на виході моделі для

вирішення задачі класифікації можливо отримати два значення – 0 та 1, які відповідно означають “не маніпуляція”, “маніпуляція”.

Були обрані декілька характеристик, які доцільно використовувати.

Ентропія є основною мірою інформативності повідомлення, одночасно з новизною контенту зростає його інформативність. Це головна характеристика в теорії інформації, її використання може позитивно вплинути на точність моделі.

Polarity важлива як показник емоційного забарвлення контенту. Кожна маніпуляція за своєю суттю є емоційною маніпуляцією [14-16]. Емоційним може бути сам зміст, або текст може бути спрямований на певний емоційний відгук. Надмірно негативний або надмірно позитивний текст може вплинути на настрій людини та її здатність приймати раціональні рішення. Крім того, пропагандистські статті відходять від простого викладу фактів і натомість звертаються до більш упередженої розповіді.

Наявність ключових слів є загальноприйнятим засобом оцінити зміст статті. Ключові слова допомагають сприйняти направленість тексту, в умовах данного обчислення в якості ключових слів буде використовуватися словник пропагандистської лексики.

З використання polarity як вхідного параметра до мережі випливає ще декілька можливих показників висловлення емоції в тексті. Перший показник – це пунктуація, окличні або питальні речення за визначенням є упередженими, виражають певну позицію.

Текст, написаний великими літерами сприймається читачем як крикливий, той, який необхідно читати з підвищеною інтонацією. Він може стати одним з показників емоційності тексту.

Статистичні показники тексту в контенті, такі як середня довжина слова та середня довжина речення використовуються в нашому датасеті через свою роль у Natural Language Processing, тобто вона є доречною при визначенні емоційності тексту.

Рекламний політичний контент має більшу довжину, ніж не рекламний, з тих причин що ставить ціль підкріпити правильність позиції, що описуються у рекламі.

Таким чином, довжина тексту може бути використана як потенційний показник наявності пропаганди у тексті.

Зображення моделі нейронної мережі наведено на рис. 2.1. На ньому зображені вхідні характеристики для нейронної мережі, кількість шарів та можливі вихідні класи.

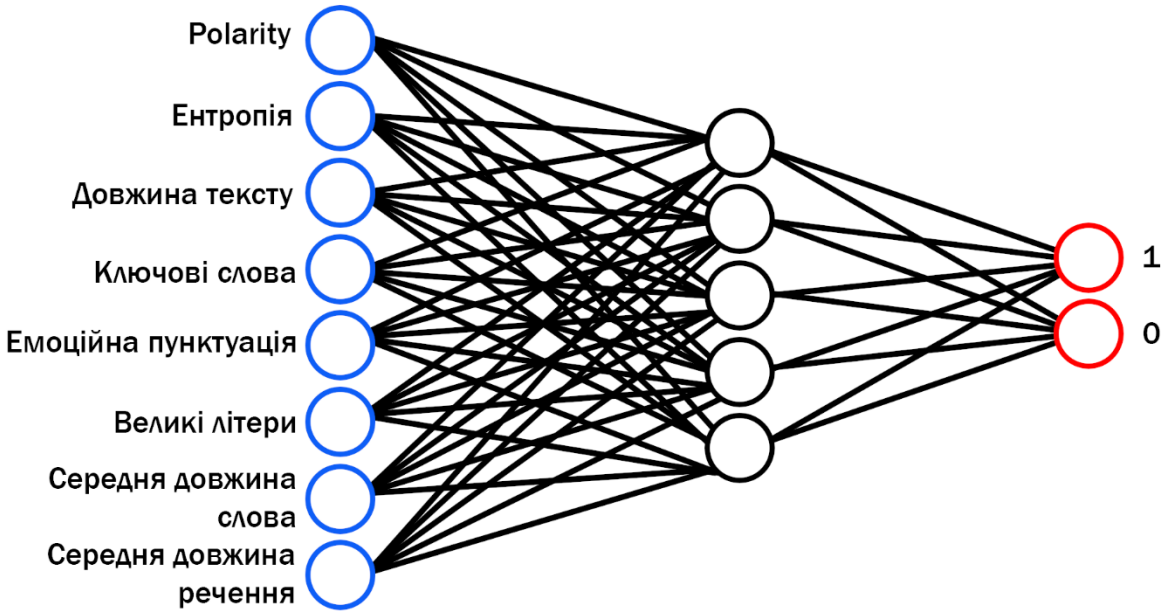

Рисунок 2.1 – Модель нейронної мережі для класифікації текстового контенту.

2.2 Розробка формул для обчислення характеристик тексту

В якості вхідних параметрів мережів, доречними для вирішення задачі класифікації були обрані:

- Polarity.
- Ентропія.
- Довжина тексту.
- Частота вживання ключових слів.
- Частота появи емоційної пунктуації.
- Частота появи слів, написаних лише великими літерами.
- Середня довжина слова.
- Середня довжина речення.

Підготовка навчальних та тестових вибірок виконана за допомогою рішення на мові Python. Мову Python обрано через її підтримку модулів NLP. Модулі NLP були необхідні для розрахунку значень polarity. У якості модуля NLP обрана бібліотека TextBlob. Код програми для попередньої обробки даних буде наведений у Додатку Б.

В якості вибірки для навчання були обрані сумарно 5062 статей з Телеграм-каналів, які відзначаються як маніпулятивні [25] та 4850 статей з "білого списку" [26]. Список ресурсів буде наведено у Додатку В.

Окремо була складена вибірка для тестування, яка містить 22335 статей, з яких 17485 визначені як маніпулятивні.

Перед обчисленням характеристик тексту, він проходить стадію попередньої обробки. З тексту вилучаються символи табуляції та переносу рядка "\n", "\t", "\r", нестандартні лапки "«" "»", крапка з комою ";", а також повторювані пробіли " ", " ", "   ". Ці символи замінюються на один пробіл " ".

Для обчислення значення polarity використовується модуль TextBlob. Модуль TextBlob надає власну версію класифікатора Naïve Bayes Classifier (NBC), попередньо навчену для англійської мови з можливістю навчання для інших мов. Для визначення polarity для української мови, були розроблені словники, що складалися з пар слів та їх емоційного забарвлення (два значення: позитивне, негативне). Для навчання NBC ми модифікували існуючий .csv-словник негативно і позитивно забарвлених українських слів [20]. Потім за допомогою методу prob.classify() навченого NBC обчислюється ймовірність того, що контент позитивний або негативний. В залежності від того, який варіант більш вірогідний, програма повертає значення змінної rating.prob("pos") або -1*rating.prob("neg"). Оцінка контенту може приймати значення від -1 до 1, тобто від надмірно негативного до надмірно позитивного.

Для обчислення ентропії використовується формула Шеннона.

$$H = -\sum_{i=1}^{m} p(i) * \log_2 p(i) \qquad (2.1)$$

де m – кількість символів у тексті контенту;

$p(i)$ – частота появи одного символу.

Довжина тексту обчислюється за допомогою функції len() без вилучення пробілів та знаків пунктуації.

У якості ключових слів використовується список найбільш вживаної проросійськими Телеграм-каналами лексики [27]. Ця характеристика вимірюється як співвідношення кількості наявних ключових слів до загальної кількості слів у тексті.

$$keywords\_frequency = \frac{n}{N} \tag{2.2}$$

де n – кількість наявних ключових слів;

N – загальна кількість слів у тексті.

Емоційною пунктуацією в Python-скрипті вважаються поєднання спеціальних символів “!!”, “!?”, “?!”, “!!!”, “!!!!”, “??”, “...”, “....”, “…”. Частота емоційної пунктуації вимірюється як відношення добутку довжини поєднання на кількість таких поєднань до кількості знаків пунктуації в тексті.

$$excessive\_punc = \frac{\sum_{i=1}^{m} l(i) * n(i)}{N} \tag{2.3}$$

де m – довжина масиву емоційної пунктуації;

$l(i)$ – довжина відповідної комбінації;

n – кількість випадків появи комбінації;

N – загальна кількість символів пунктуації в тексті.

Частота появи слів, написаних тільки великими літерами, обчислюється як відношення кількості таких слів до кількості слів у тексті. До таких слів не відносяться абревіатури НАСА, NASA, МЗС, ОАЭ, РПК, ФАС, MOL, НПЗ, ЦИК, PS, P.S, ЄС, ЕС, Ф, М, РНКБ, ЛДПР, СНГ, HSBC, G4S, HSBC, SWIFT, РСЗО, БТР, С-300, СИБЗ, ВУС, ВС, ШОС, IQ, СВПД, БРИКС, SDN, OFAC, СВР, ДРГ, СДД, SSD. HDD, RT, SEFE, ФРГ, FTX, ЕАЭС, ЄАЕС, ВАС, АСЕАН, ЗВР, ЖКХ, ВВП, ВНП, ВУЗ, ВНЗ, МИ-6, MI-6, КНР, ИПСО, GMLRS, ВДВ, ПМР, АТЭС, ATEC, MQ-1C, WSJ, US, КСИР, NLAW, SUN, UPD, РЭС, НПЗ, КНПЗ,СИЗО, ЗРК, ВВС, ВПС, РИ, ВГТРК, ВМС, ЗАЕС, ЗАЭС, МАГАТЕ, ЛЕП, ЧП, HIMARS, МИД, P&I, МО, САУ, МАГАТЭ, ТЯО, МВД, Я, ЛЭП, БПЛА, М-14, ВКС, Х-32, ГОЗ, ГРЭС, ГРЕС, ЕК, ЗВР, ЦБ, ВПК, NASAMS, M109L, МОЗ, MOS, УК, ОПЕК, ВС, СОБР, МО, CNN, СБУ, HIMARS, ТАСС, СНБО, РФ, СЦКК, СВО, ППО, ПВО, ЖД, ТЭЦ, ГЭС, СМИ, ЗМІ, ЧВК, ВСУ,

ЗСУ, ЖК, УПЦ, РПЦ, ТЦ, СПЖ, КП, ПЦУ, ЦДАЗУ, ВДНХ, СССР, СРСР, УССР, УПА, РСФСР, ЛГБТ, ООН, НАТО, США, КПСС, СПЖ, ЦСБ, ОВА, ФСБ, ФБР, ЦРУ, ГУР, ATACMS, ВР, ЕП, ТЭС, ГЭС, ТЕС, ГЕС, АЭС, АЕС, VESTI, ВГА, ДТЭК, ЕС, G7, НАБУ, САП, ОГА, G20, ГБР, ДТП, ДПС, ГИБДД, ЦАХАЛ, ХАМАС, ОПЗЖ, КНДР, КГГА, ЛНР, ДНР, ЛДНР, БМП, ТОЧКА-У, LED, DPA, ЗНО, ДПА, ДОР, IKEA, а також римські цифри.

$$caps = \frac{n}{N} \tag{2.4}$$

де n – кількості наявних слів, написаних тільки великими літерами;

N – загальна кількість слів у тексті.

Середня довжина слова дорівнює відношенню суми довжин слів на їх кількість. Кількість слів визначається розбиттям тексту контенту на частини по пробілу.

$$avg\_word = \frac{l}{n} \tag{2.5}$$

де l – сума довжин слів у тексті;

n – кількість слів у тексті.

Середня довжина речення обчислюється наступним чином. Текст контенту розбивається на речення по наявності крапки між ними. Одночасно цей текст розбивається на слова по наявності пробілу. Характеристика дорівнює загальної кількості речень на сумарну кількість слів.

$$avg\_sentence = \frac{l}{n} \tag{2.6}$$

де l – кількість слів у тексті;

n – кількість речень у тексті.

2.3 Вибір класифікатора для забезпечення потреб мережі

Синтез нейромережевої моделі виконували на базі класифікатора K-Star. Класифікатор K-Star – це алгоритм машинного навчання, що використовується для класифікації даних, заснований на k–найближчих сусідах (k–NN). Класифікатор K-Star є реалізацією k–NN, яка використовується в бібліотеці Weka для машинного

навчання. Weka – це відкрите програмне забезпечення, яке має набір алгоритмів машинного навчання для вирішення завдань інтелектуального аналізу даних, таких як підготовка даних, класифікація, регресія, кластеризація, виявлення правил асоціацій та візуалізація. Weka містить API, написане на мові Java, яке надає доступ до наявних алгоритмів машинного навчання з мінімальними налаштуваннями. Ліцензія програмного забезпечення Weka є GNU General Public License.

Основна ідея класифікатора K-Star полягає в тому, щоб зберегти найбільш вагомі атрибути (за допомогою статистики кількості голосів) з кращих k–найближчих сусідів і використовувати їх для класифікації нового зразка даних. Класифікатор K-Star простий у реалізації, і його легко використовувати для класифікації даних без додаткових обчислень. K-Star може бути використаний для класифікації даних з різних областей, таких як текстові дані, зображення, сигнали і т.д. Він дозволяє інтерпретувати результати класифікації, оскільки він використовує лише найбільш вагомі атрибути для прийняття рішень. Він використовує статистику голосів для прийняття рішень, що зменшує вплив шуму або помилок у даних.

Найголовніше, класифікатор K-Star може бути використаний для обробки великих обсягів даних, і він працює досить швидко для класифікації нових зразків даних.Даний алгоритм доступний у програмному забезпеченні Weka.

Висновки до розділу 2

В даному розділі була формально визначена модель майбутньої нейронної мережі та обрана парадигма машинного навчання, яка найбільш повно підходить до визначених даних – навчання з вчителем. В моделі буде визначено 8 входів, кількість виходів дорівнює 1, кількість шарів дорівнює 5.

Для моделі був обраний класифікатор, що найбільш повно описує представлені дані – класифікатор K-Star. K-Star дозволяє зберегти найбільш вагомі атрибути (за допомогою статистики кількості голосів) з кращих k-найближчих сусідів і використовувати їх для класифікації нового зразка даних. K-Star може бути використаний для класифікації даних з різних областей, таких як текстові дані,

зображення, сигнали і т.д. У порівнянні з іншими класифікаторами, він дає найбільш високий показник точності мережі при її валідації.

Із потенційних числових характеристик, що можуть бути використані для нейронної мережі такого типу, було обрано 8. Дані характеристики – це ентропія, polarity, наявність ключових слів, частота появи емоційної пунктуації, частоти появи слів, написаних лише великими літерами, середня довжина слова, середня довжина речення, довжина речення. Характеристики, що не були обрані, мали або недоцільно важке обчислення у порівнянні з точністю, що надають мережі, або їх обчислення неможливе з огляду на комплексність комп'ютерних систем. Були розроблені та наведені формули, що використовуються для обчислення кожної з фінальних характеристик.

РОЗДІЛ 3
ПІДГОТОВКА НАВЧАЛЬНИХ ВИБІРОК ТА РЕАЛІЗАЦІЯ МЕРЕЖІ

3.1 Створення датасетів шляхом парсингу Телеграм-каналів

Для задачі парсингу був використаний сторонній код, функції якого були дороблені відповідно до потреб створення датасету. Далі знаходяться прокоментовані фрагменти коду.

```
file = open('Telegram Main News Manipulation.txt', 'r')
urls = file.readlines()
user_input_channel = 'dummy'
for line in urls:
    url = line.replace('\n', '')
    user_input_channel = url
    if user_input_channel.isdigit():
        entity = PeerChannel(int(user_input_channel))
    else:
        entity = user_input_channel

    my_channel = await client.get_entity(entity)

    offset_id = 0
    limit = 100
    all_messages = []
    total_messages = 0
    total_count_limit = 1000
```

Рисунок 3.1 – Фрагмент коду для парсингу Телеграм-каналів.

Цей фрагмент коду відкриває файл з назвою “Telegram Main News Manipulation.txt” і зчитує його вміст за допомогою методу readlines(), який повертає список рядків з файлу. Були підготовлені два переліки каналів: “Telegram Main News Manipulation.txt” та “Telegram Main News Not Manipulation.txt” – в яких містяться назви каналів, що є, відповідно, маніпулятивними та не маніпулятивними. Вони

наведені в додатку В. Далі, використовуючи цикл for, він проходиться по кожному рядку із списку urls. Змінна line містить поточний рядок, який потрібно обробити.

Для кожного рядка, код виконує наступні кроки.

- Замінює символи нового рядка \n на порожній рядок, видаляючи їх з поточного рядка.
- Присвоює змінній user_input_channel значення, що міститься у поточному рядку.
- Якщо user_input_channel складається лише з цифр, код створює змінну entity, що містить об'єкт PeerChannel, який містить ці цифри у вигляді цілого числа.
- Якщо user_input_channel містить ще й символи крім цифр, то змінна entity просто отримує значення user_input_channel.
- За допомогою асинхронної функції await client.get_entity(entity) отримується сутність з телеграму (канал, чат, користувач і т.д.) згідно з отриманим значенням entity.
- Встановлює початкові значення для змінних offset_id, limit, all_messages, total_messages, та total_count_limit.
- Змінна all_messages використовуватиметься для зберігання всіх повідомлень, які будуть отримані з каналу.
- Змінна total_messages буде використовуватися для зберігання загальної кількості отриманих повідомлень.
- Змінна total_count_limit встановлює максимальну кількість повідомлень, які можна отримати (в даному випадку 1000).

```
for message in all_messages:
    if 'message' in message and message['message'] != '' and message['message'] is not None:
        print(message['message'])
        text = message['message']
        text = text.replace("⚡", "")
        text = text.replace("⚡", "")
        text = text.replace(" \n ", " ")
        text = text.replace("\n ", " ")
        text = text.replace(" \n", " ")
        text = text.replace("\n", " ")
        text = text.replace("\t", " ")
        text = text.replace("\r", " ")
        text = text.replace("«", "")
        text = text.replace("».", ".")
        text = text.replace("»", "")
        text = text.replace("\"", "")
        text = text.replace(";", "")
        text = text.replace("    ", " ")
        text = text.replace("   ", " ")
        text = text.replace("  ", " ")
        df.loc[len(df)] = text

name = 'C:\\Users\\wadym\\PycharmProjects\\PsychoOps\\Telegram Manipulation\\{}.csv'.format(
    user_input_channel[12:])
df.to_csv(name, sep=';', encoding='utf-8-sig', index=False)
```

Рисунок 3.2 – Фрагмент коду для попередньої обробки тексту.

Після отримання списку повідомлень all_messages з відповідного каналу, програма виконує дії з усіма окремими повідомленнями (messages) із масиву all_messages.

- Якщо повідомлення містить текст, то виводить його у консоль.
- З тексту видаляється різноманітна розділова пунктуація, символи та пробіли, щоб підготувати текст для аналізу.
- Очищений текст записується у новий рядок даних (df) за допомогою методу loc.
- Зберігає отриманий рядок даних у форматі .csv за допомогою методу to_csv.
- Назва кінцевого файлу формується зі шляху до директорії, де знаходиться проєкт (C:\Users\wadym\PycharmProjects\PsychoOps\Telegram Manipulation\), та імені каналу, з якого було отримано повідомлення (user_input_channel).

3.2 Обчислення характеристик текстів у створених датасетах

Після виконання цього коду створюються списки csv зі статтям кожного з каналів. Об’єднання цих списків в один для формування повного датасету відбувається пізніше. Для того, щоб обчислити характеристики статтей та занести їх у csv-файл використовується код файлу TextEvaluation.py. Вказати назву файлу важливо, так як інші файли будуть посилатися на нього.

```
import pandas as pd
import ArticleProcessing
import UkrainianProcessing
import RussianProcessing
from langdetect import detect

ua_classifier = UkrainianProcessing.trainClassifier()
ru_classifier = RussianProcessing.trainClassifier()

df = pd.read_csv('C:\\Users\\wadym\\PycharmProjects\\PsychoOps\\Test\\NovinachTest.csv', delimiter=";")
df2 = pd.DataFrame(columns=['message','polarity','entropy','length','keywords_frequency',
                            'excessive_punc', 'caps','avg_word','avg_sentence','output'])
print(df)
```

Рисунок 3.3 – Фрагмент коду, відкриття списку завантажених текстів.

Для роботи даного коду необхідні бібліотеки pandas – для роботи з csv-файлами, langdetect – для автоматичного визначення мови тексту, та створені для потреб даної роботи модулі ArticleProcessing, UkrainianProcessing, та RussianProcessing, що розглянуті нижче.

Наступні рядки створюють об'єкти класифікаторів ua_classifier та ru_classifier для обробки текстів на українській і російській мовах. Метод trainClassifier() тренує об’єкт (instance) класифікатора на заздалегідь створених датасетах. Вони наведені в додатку В.

Потім для обраного файлу .csv читаються дані за допомогою функції read_csv з бібліотеки pandas. Дані зберігаються у датафремі з назвою df. Після цього створюється новий порожній DataFrame з назвою df2 та задаються назви стовпців. Він використовується для збереження результатів обробки текстів.

```
for i in range(len(df)):
    try:
        text = df['message'][i]
        length = len(text)
        excessive_punc = ArticleProcessing.excessivePunctuation(text)
        entropy = ArticleProcessing.Entropy(text)
        text_for_keywords = text.lower()
        keywords_frequency = ArticleProcessing.keywordFrequency(text_for_keywords)
        caps = ArticleProcessing.capsLock(text)
        avg_word = ArticleProcessing.averageWord(text)
        avg_sentence = ArticleProcessing.averageSentence(text)

        polarity = 0.0
        lang = detect(text)
        if lang == 'ua':
            print('Ukrainian!')
            polarity = UkrainianProcessing.classifyArticle(text, ua_classifier)
        elif lang == 'ru':
            print('Russian!')
            polarity = RussianProcessing.classifyArticle(text, ru_classifier)
        else:
            print('Ukrainian!')
            polarity = UkrainianProcessing.classifyArticle(text, ua_classifier)
        # 1 = Маніпуляція
        # 0 = Не маніпуляція
        #list_row = [text, polarity, entropy, length, keywords_frequency, excessive_punc, caps, avg_word, avg_sentence, 1]
        list_row = [text, polarity, entropy, length, keywords_frequency, excessive_punc, caps, avg_word, avg_sentence, 0]
        print(list_row)
        df2.loc[len(df2)] = list_row
```

Рисунок 3.4 – Фрагмент коду для обчислення характеристик тексту.

Даний код складається з циклу for, який працює з кожним рядком вхідного датафрейму df. З кожного рядка отримується текст повідомлення, довжина тексту, частота вживання ключових слів, кількість надмірних знаків пунктуації, кількість слів у верхньому регістрі, середня довжина слова та середня довжина речення. Для обчислення цих характеристик використовуються відповідні функції модулю ArticleProcessing.

Далі за допомогою функції detect з пакету langdetect визначається мова тексту. Якщо мова є українською або російською, то за допомогою функцій classifyArticle з модулів. UkrainianProcessing та RussianProcessing відповідно визначається полярність (polarity) тексту. Остаточні дані додаються до нового датафрейму df2. Датафрейм df2 записується у csv-файл. Якщо виникає помилка, то цикл продовжується, і стаття не заноситься у датасет.

3.3 Реалізація формул для обчислення характеристик у середовищі PyCharm

Розглянемо вміст модулю ArticleProcessing. Для роботи модуля необхідні бібліотеки newspaper та її модуль Article для роботи з текстом статей, string та punctuation – для пошуку пунктуації у тексті, модуль re для регулярних запитів, модуль collections та Counter – для швидкого підрахунку ентропії, модуль math та функція log для логарифмічної функції.

```
from newspaper import Article
from string import punctuation
import re
from collections import Counter
from math import log

1 usage
def keywordFrequency(text):
    file = open('keywords.txt',encoding = 'utf-8', mode='r')
    words = file.readlines()
    frequency_rating = 0
    for word in words:
        word = word.strip('\n')
        #print(word)
        res = [i for i in range(len(text)) if text.startswith(word, i)]
        frequency_rating += len(res)
    return frequency_rating / len(text)
```

Рисунок 3.5 – Функція keywordFrequency().

Функція keywordFrequency приймає текст, який буде перевірений на наявність ключових слів, та повертає рейтинг частоти зустрічі цих слів у тексті. Список ключових слів зчитується з файлу keywords.txt, де кожне ключове слово записане в окремому рядку. Для кожного ключового слова функція шукає всі входження у тексті та додає їх кількість до загального рейтингу. У кінці виконується нормування рейтингу на довжину тексту, щоб врахувати різні довжини текстів.

```
1 usage
def Entropy(text):
    counts = Counter(text)
    frequencies = ((i / len(text)) for i in counts.values())
    return - sum(f * log(f, 2) for f in frequencies)
```

Рисунок 3.6 – Функція Entropy().

Функція Entropy обчислює ентропію тексту. Вона використовує бібліотеку Counter для підрахунку кількості кожного символу у тексті. Потім обчислюється частота кожного символу та вираховується значення ентропії шляхом використання формули Шеннона -1 * сума (частота символу * логарифм (частота символу, 2)).

```
1 usage
def averageWord(text):
    words = text.split(" ")
    average_word_length = sum(len(word) for word in words) / len(words)
    return average_word_length
```

Рисунок 3.7 – Функція averageWord().

Функція averageWord обчислює середню довжину слів у тексті. Вона розбиває текст на окремі слова за допомогою функції split() та обчислює суму довжини кожного слова. За допомогою цієї суми та загальної кількості слів у тексті (що дорівнює довжині розбитого тексту за пробілами) вона обчислює середню довжину слова.

```
1 usage
def averageSentence(text):
    sentences = text.split(".")
    words = text.split(" ")
    if (sentences[len(sentences) - 1] == ""):
        average_sentence_length = len(words) / len(sentences) - 1
    else:
        average_sentence_length = len(words) / len(sentences)
    return average_sentence_length
```

Рисунок 3.8 – Функція averageSentence().

Ця функція обчислює середню довжину речення в тексті. Функція приймає рядок тексту, розділяє його на окремі речення за допомогою символу крапки, а потім на окремі слова за допомогою пробілу. Далі функція рахує кількість слів та речень в тексті і обчислює середню довжину речення, яка є відношенням кількості слів до кількості речень. Якщо останній елемент списку речень є порожнім рядком, то з загальної кількості речень віднімається 1, оскільки останнє речення не закінчується крапкою. Результатом роботи функції є середня довжина речення в тексті.

```
1 usage
def deleteEmojis(article_text):
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  # emoticons
                               u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                               u"\U0001F680-\U0001F6FF"  # transport & map symbols
                               u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                               u"\U00002500-\U00002BEF"  # chinese char
                               u"\U00002702-\U000027B0"
                               u"\U00002702-\U000027B0"
                               u"\U000024C2-\U0001F251"
                               u"\U0001f926-\U0001f937"
                               u"\U00010000-\U0010ffff"
                               u"\u2640-\u2642"
                               u"\u2600-\u2B55"
                               u"\u200d"
                               u"\u23cf"
                               u"\u23e9"
                               u"\u231a"
                               u"\ufe0f"  # dingbats
                               u"\u3030"
                               "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', article_text)
```

Рисунок 3.9 – Функція deleteEmojis().

Функція deleteEmojis() видаляє емодзі та інші символи зі статті. Вона використовує регулярні вирази, щоб знайти символи, які потрібно видалити, та повертає текст статті без цих символів.

```
1 usage
def capsLock(text):
    abbr = "НАСА, NASA, МЗС, ОАЗ, РПК, ФАС, MOL, НПЗ, ЦИК, PS, P.S, ЄС, ЕС, Ф, М, РНКБ, ЛДПР," \
           " СНГ, HSBC, G4S, HSBC, SWIFT, РСЗО, БТР, С-300, СИБЗ, ВУС, ВС, ШОС, IQ, СВПД, БРИКС, " \
           "SDN, OFAC, СВР, ДРГ, СДД, SSD. HDD, RT, SEFE, ФРГ, FTX, ЕАЭС, ЄАЕС, ВАС, АСЕАН, ЗВР, " \
           "ЖКХ, ВВП, ВНП, ВУЗ, ВНЗ, МИ-6, МІ-6, КНР, ИПСО, GMLRS, ВДВ, ПМР, АТЭС, АТЕС, MQ-1C, WSJ, " \
           "US, КСИР, NLAW, SUN, UPD, РЭС, НПЗ, КНПЗ,СИЗО, ЗРК, ВВС, ВПС, РИ, ВГТРК, ВМС, ЗАЕС, ЗАЭС, " \
           "МАГАТЕ, ЛЕП, ЧП, HIMARS, С-300, МИД, P&I, МО, САУ, МАГАТЭ, ТЯО, МВД, Я, ЛЭП, БПЛА, М-14, " \
           "ВКС, Х-32, ГОЗ, ГРЭС, ГРЕС, ЕК, ЗВР, ЦБ, ВПК, NASAMS, M109L, МОЗ, MOS, УК, ОПЕК, ВС, СОБР, " \
           "МО, CNN, СБУ, HIMARS, ТАСС, СНБО, РФ, СЦКК, СВО, ППО, ПВО, ЖД, ТЭЦ, ГЭС, СМИ, ЗМІ, ЧВК, ВСУ, " \
           "ЗСУ, ЖК, УПЦ, РПЦ, ТЦ, СПЖ, КП, ПЦУ, ЦДАЗУ, ВДНХ, СССР, СРСР, УССР, I, II, III, IV, V, VI, VII, " \
           "VIII, IIX, IX, X, XI, XII, XIII, XIV, XV, XVI, XVII, XVIII, XIIX, XIX, XX, XXX, УПА, РСФСР, " \
           "ЛГБТ, ООН, НАТО, США, КПСС, СПЖ, ЦСБ, ОВА, ФСБ, ФБР, ЦРУ, ГУР, ATACMS, ВР, ЕП, ТЭС, ГЭС, ТЕС, " \
           "ГЕС, АЭС, АЕС, VESTI, ВГА, ДТЭК, ЕС, G7, НАБУ, САП, ОГА, G20, ГБР, ДТП, ДПС, ГИБДД, ЦАХАЛ, ХАМАС, " \
           "ОПЗЖ, КНДР, КГГА, ЛНР, ДНР, ЛДНР, БМП, ТОЧКА-У, LED, DPA, ЗНО, ДПА, ДОР, ІКЕА"
    text = re.sub(r'[0-9]+', '', text)
    pattern = r"\b(?=[MDCLXVII])M{0,4}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})([II]X|[II]V|V?[II]{0,3})\b\.?"
    text = re.sub(pattern, '', text)
    text = deleteEmojis(text)
    text = text.replace("_", '')
    text = text.replace("+", '')
    text = text.replace("№",'')
    text = text.replace("#", '')
    text = text.replace("%",'')
    text = text.replace(")", '')
    text = text.replace("(", '')
    text = text.replace(":", '')
    text = text.replace(", ", ' ')
    text = text.replace(" ,", ' ')
    text = text.replace(",", '')
```

Рисунок 3.10 – Функція capsLock(), винятки для роботи функції.

```
text = text.replace("??? ", " ")
text = text.replace("!!! ", " ")
text = text.replace("???", "")
text = text.replace("!!!", "")
text = text.replace("?? ", "")
text = text.replace("!! ", " ")
text = text.replace("!? ", " ")
text = text.replace("?! ", " ")
text = text.replace(" — ", " ")
text = text.replace(" - ", " ")
text = text.replace("— ", " ")
text = text.replace(" —", " ")
text = text.replace("- ", " ")
text = text.replace(" -", " ")
text = text.replace("??", "")
text = text.replace("!!", "")
text = text.replace("!?", "")
```

Рисунок 3.11 – Функція capsLock, вилучення розділових знаків.

```
    text = text.replace("?!", "")
    text = text.replace(". ","  ")
    text = text.replace("? ", "  ")
    text = text.replace("! ", "  ")
    text = text.replace("|", "  ")
    text = text.replace(".","")
    text = text.replace("?", "")
    text = text.replace("!", "")
    text = text.replace("—", "  ")
    text = text.replace("    ", "  ")
    text = text.replace("  ", "  ")
    print(text)
    words = text.split("  ")
    num_of_words = len(words)
    print(words)
    num_of_caps = 0
    for word in words:
        if word!=text[0] and word.upper() == word and word not in abbr:
            print(word)
            num_of_caps+=1
    print(num_of_caps)
    return num_of_caps/num_of_words
```

Рисунок 3.12 – Функція capsLock, відсоткове співвідношення слів, написаних великими літерами.

Функція capsLock() повертає кількість слів, написану виключно верхнім регістром у порівнянні зі всією кількістю слів. Спочатку визначається змінна abbr, яка містить перелік скорочень, які не будуть враховуватися як слова, написані виключно верхнім регістром, так як вони не несуть емоційного змісту. До цього переліку були додані найбільш популярні абревіатури, а також римські цифри. Далі, використовуючи регулярні вирази, текст очищається від римських чисел, пунктуації та емодзі. Метод звертається до списку words, в якому підраховуються кількість слів у верхньому регістрі. Кількість таких слів нормується у відповідністю з кількістю слів у всьому тексті.

```
2 usages
def deletePunctuation(article_text):
    article_text = article_text.translate(str.maketrans('', '', punctuation))
    article_text = article_text.replace('–', '')
    article_text = article_text.replace('\n', '')
    article_text = article_text.replace('«', '')
    article_text = article_text.replace('»', '')
    article_text = article_text.replace('|', ' ')
    article_text = article_text.replace('  ', ' ')
    article_list = article_text.split(" ")
    new_article_list = []
    for word in article_list:
        newword = ''.join([i for i in word if not i.isdigit()])
        new_article_list.append(newword)
    article_text = ' '.join(new_article_list)
    return article_text
```

Рисунок 3.13 – Функція deletePunctuation().

Функція deletePunctuation() призначена для обробки тексту з статті або іншого джерела шляхом видалення всіх знаків пунктуації та цифр.

Спочатку, всі знаки пунктуації видаляються з тексту за допомогою методу translate() та створенням таблиці заміни знаків за порожніми рядками. Далі, окрім знаку дефісу, який може вказувати на складне слово, всі інші знаки пунктуації, такі як крапки, коми, лапки тощо, також замінюються на порожні рядки.

Далі, кожне слово в рядку розділяється за допомогою методу split() з аргументом “ ”, тобто пробілом, і створюється список слів article_list. Далі відбувається створення порожнього списку new_article_list, який буде містити слова після обробки. У циклі for програма отримує кожне слово в article_list та виконує наступні дії:

- створює змінну newword;
- проходить по кожному символу у слові word;
- якщо символ не є цифрою, то додає його до newword;
- на кожній ітерації отримує newword без цифр;
- додає newword до списку new_article_list;

- знову перетворює new_article_list на рядок, розділяючи слова пробілами, та повертаєм цей рядок як результат роботи функції.

```
1 usage
def excessivePunctuation(text):
    expunc = {'!!', '!?', '?!', '!!!', '!!!!', '??', '...', '....','…'}
    count = lambda l1, l2: sum([1 for x in l1 if x in l2])
    allpunc = count(text, set(punctuation))
    if allpunc == 0:
        return 0.0
    expuncnum = 0
    m = 0
    for ex in expunc:
        res = [i for i in range(len(text)) if text.startswith(ex, i)]
        expuncnum += len(ex) * len(res)
    return (expuncnum / allpunc)
```

Рисунок 3.14 – Функція excessivePunctuation().

У функції excessivePunctuation() визначається надмірна пунктуація та її кількість у порівнянні зі всією пунктуацією. Спочатку створюється словник expunc, який містить набір можливих комбінацій зайвої пунктуації. Далі визначається кількість всієї пунктуації в тексті за допомогою лямбда-функції count, яка використовує вбудовану функцію sum для підрахунку кількості елементів в масиві.

Якщо кількість всієї пунктуації у тексті дорівнює нулю, то функція повертає значення 0.0, оскільки зайвої пунктуації в цьому випадку немає.

У протилежному випадку обчислюється кількість зайвої пунктуації в тексті expuncnum, перебираючи всі можливі комбінації зайвої пунктуації та знаходячи їх кількість. Також обчислюється загальна кількість пунктуації в тексті allpunc.

Нарешті, обчислюється відсоток зайвої пунктуації як частка expuncnum від allpunc та повертається це значення. Хоча зайва пунктуація шукається як комбінація відповідних символів, але для обчислення відсоткового значення кількість символів у комбінації множиться на частоту появи останньої.

3.4 Вирішення завдання семантичної обробки української та російської мов

Наступним розглянемо модулі UkrainianProcessing та RussianProcessing. Як було вказано, вони містять функції обробки української та російської мови. Для роботи UkrainianProcessing потрібен модуль pandas та класифікатор NaiveBayesClassifier з модулю textblob.

```
import pandas as pd
from textblob.classifiers import NaiveBayesClassifier
 # from textblob.sentiments import NaiveBayesAnalyzer

def deleteUAStopwords(article_text):
    words = article_text.split(" ")
    stopwords_ua = pd.read_csv("stopwords_ua.txt", header=None, names=['stopwords'])
    stopwords_ua_list = list(stopwords_ua.stopwords)
    newwords = ""
    first = True
    for word in words:
        if word not in stopwords_ua_list:
            if first:
                first = False
                newwords += word
            else:
                newwords += " " + word
    return newwords
```

Рисунок 3.15 – Функція deleteUAStopwords().

Функція deleteUAStopwords() отримує текст статті в якості вхідного параметру. Функція розділяє текст на окремі слова, зчитує список українських стоп-слів з файлу “stopwords_ua.txt” та перевіряє кожне слово на те, чи воно є стоп-словом. Якщо слово не є стоп-словом, то воно додається до змінної “newwords”.

Файл “stopwords_ua.txt” містить список слів, які не несуть особливої інформації та можуть бути відфільтровані без втрати смислового навантаження тексту.

```
3 usages
def trainClassifier():
    sentiment_ua_test = pd.read_csv('sentiment_ua_2.csv', delimiter=";")
    #print(sentiment_ua_test)
    sentiment_ua_test.drop('Unnamed: 0', axis=1, inplace=True)
    train = list(sentiment_ua_test.itertuples(index=False, name=None))
    #print(train)
    new_classifier = NaiveBayesClassifier(train)
    # new_analyzer = NaiveBayesAnalyzer(new_classifier)
    print("Classifier is trained!")
    return new_classifier
```

Рисунок 3.16 – Функція trainClassifier().

Функція trainClassifier() створює класифікатор текстів на основі алгоритму наївного Баєса. Він використовує бібліотеку pandas для завантаження даних для навчання класифікатора з файлу sentiment_ua_2.csv. Файл містить рядки з текстами та їх позитивні/негативні мітки. Завдяки функції drop() з пакету pandas, з датафрейму sentiment_ua_test видаляється колонка Unnamed: 0. Далі створюється список train з кортежів, що містять текст та мітку. Навчання класифікатора відбувається за допомогою функції NaiveBayesClassifier(train). Після навчання функція повертає новий класифікатор.

```
6 usages
def classifyArticle(text, new_classifier):
    rating = new_classifier.prob_classify(text)
    if rating.max() == 'pos':
        return 1 * float(rating.prob("pos"))
    if rating.max() == 'neg':
        return -1 * float(rating.prob("neg"))
```

Рисунок 3.17 – Функція classifyArticle().

Функція classifyArticle() отримує текст статті та натренований класифікатор і застосовує класифікацію на вхідному тексті.

Класифікатор приймає текст та повертає рейтинг (ймовірність) того, що текст належить до позитивного або негативного класу. Метод prob_classify повертає ProbDistI, який є інтерфейсом для роботи з ймовірнісними розподілами.

Якщо найбільш ймовірний клас – “pos” (позитивний), то функція повертає додатне число – ймовірність, що вхідний текст є позитивним. Якщо найбільш ймовірний клас – “neg” (негативний), то функція повертає від'ємне число – ймовірність, що вхідний текст є негативним.

Код для аналізу російської мови аналогічний, тільки використовує інші словники – “stopwords_ru.txt” та “sentiment_ru.csv” відповідно.

```
# -*- coding: utf-8 -*-
import pandas as pd
from textblob.classifiers import NaiveBayesClassifier
# from textblob.sentiments import NaiveBayesAnalyzer

def deleteRUStopwords(article_text):
    words = article_text.split(" ")
    stopwords_ru = pd.read_csv("stopwords_ru.txt", header=None, names=['stopwords'])
    stopwords_ru_list = list(stopwords_ru.stopwords)
    newwords = ""
    first = True
    for word in words:
        if word not in stopwords_ru_list:
            if first:
                first = False
                newwords += word
            else:
                newwords += " " + word
    return newwords
```

Рисунок 3.18 – Функція deleteRUStopwords().

```
1 usage
def trainClassifier():
    sentiment_ru_test = pd.read_csv('sentiment_ru_2.csv', delimiter=";")
    #print(sentiment_ua_test)
    #sentiment_ru_test.drop('Unnamed: 0', axis=1, inplace=True)
    train = list(sentiment_ru_test.itertuples(index=False, name=None))
    new_classifier = NaiveBayesClassifier(train)
    # new_analyzer = NaiveBayesAnalyzer(new_classifier)
    print("Classifier is trained!")
    return new_classifier
```

Рисунок 3.19 – Функція trainClassifier().

```
1 usage
def classifyArticle(text, new_classifier):
    rating = new_classifier.prob_classify(text)
    if rating.max() == 'pos':
        return 1 * float(rating.prob("pos"))
    if rating.max() == 'neg':
        return -1 * float(rating.prob("neg"))
```

Рисунок 3.20 – Функція classifyArticle().

3.5 Об’єднання датасетів для створення навчальних і тестових вибірок

Після створення .csv-файлів для кожного з каналів, необхідно об’єднати їх у датасет. Для цього використовується функція JoinMassiveCSV().

```
1 usage
def JoinMassiveCSV():
    list = []
    for i in range(1,365):
        list.append("C:\\Users\\wadym\\PycharmProjects\\PsychoOps\\"
                    "main_massive_not_propaganda\\articles_news_not_propaganda_{}.csv".format(i))
    df = pd.DataFrame(columns=['title', 'url', 'overly_emotional', 'missing_information',
                               'reverse_conclusion', 'sum',
                               'propaganda_effectiveness',
                               'algorithm_assessment', 'human_assessment'])
    for n in range(0,364):
        try:
            df1 = pd.read_csv(list[n], delimiter=";")
            df = pd.concat([df, df1], ignore_index=True)
        except:
            continue

    df.drop('Unnamed: 0', axis=1, inplace=True)
    print(df)
    df.to_csv('C:\\Users\\wadym\\PycharmProjects\\PsychoOps\\main_massive_not_propaganda\\joined.csv',
              sep=';', encoding='utf-8-sig')
```

Рисунок 3.21 – Функція JoinMassiveCSV().

Спочатку він створює список шляхів до файлів, які потрібно об'єднати, і потім за допомогою циклу for проходиться по цьому списку і зчитує кожний файл в окремий DataFrame за допомогою функції pd.read_csv().Потім використовуючи функцію pd.concat() він об'єднує всі DataFrame в один. У кінці він видаляє непотрібний стовпець і зберігає об'єднані дані у новому .csv-файлі. Фрагмент даних для навчальних та тестових вибірок зображений на рисунку 3.22.

	A	B	C	D	E	F	G	H	I
1	polarity	entropy	length	keywords_	excessive_	caps	avg_word	avg_sente	output
2	-0.98733	4.751395	376	0	0	0	6.25	17.33333	1
3	-0.86989	4.933556	55	0	0	0	8.333333	3	1
4	-0.90602	4.531453	203	0	0	0	4.230769	19.5	1
5	0.880552	4.638565	568	0	0	0	5.694118	28.33333	1
6	-0.82516	4.287946	92	0	0	0	6.75	6	1
7	-0.90602	4.310443	27	0	0	0	8.333333	3	1
8	-0.994	4.445103	72	0	0	0	5.636364	4.5	1
9	-0.79373	4.019879	45	0	0	0	5.571429	7	1
10	0.999465	4.609793	614	0.001629	0.133333	0	5.684783	17.4	1
11	-0.70987	4.671927	526	0	0	0	5.75641	12	1
12	-0.83901	4.775968	324	0	0	0.025	6.926829	10.25	1
13	-0.92143	4.091857	62	0	0	0	5.3	10	1
14	-0.99811	5.698528	345	0	0	0.121951	7.238095	10.5	1
15	-0.96528	4.463688	285	0	0	0	8.225806	9.333333	0
16	-0.94622	4.634061	124	0	0	0	8.615385	3.333333	0
17	-0.98458	4.633312	395	0	0	0	6.764706	9.2	0
18	-0.65076	4.865766	112	0	0	0	6.533333	6.5	0
19	-0.9425	4.573233	121	0	0	0	8.307692	3.666667	0
20	-0.70861	4.569536	244	0	0	0	7.448276	8.666667	0
21	0.945784	4.861795	333	0	0	0	6.422222	14	0
22	0.852282	4.644257	545	0	0	0.026667	6.090909	9.625	0
23	-0.8726	4.632573	269	0	0	0	5.923077	12	0
24	0.565308	4.646492	465	0	0	0	7.303571	8.5	0
25	-0.94622	4.591554	83	0	0	0	7.4	2.333333	0
26	-0.84215	4.548522	165	0	0	0.043478	6.5	6.666667	0
27	-0.85424	4.740809	92	0	0	0	8.2	2.666667	0
28	-0.93118	4.61247	223	0	0	0	7.296296	13.5	0
29	-0.9998	4.620012	573	0	0.136364	0.0125	5.833333	7.4	0

Рисунок. 3.22 – Фрагмент даних для навчальних та тестових вибірок.

3.6 Навчання моделі на створеній вибірці

На рисунках 3.23, 3.24 наведені вибірки для навчання і тестування, відформатовані для програмного забезпечення Weka.

```
@relation 'telegram_dataset_weka-weka.filters.unsupervised.attribute.Reorder-R1,2,3,4,5,6,7,8,9-weka.filters.unsupervised.attribute.Reorder-R1,2,3,4,5,6,7,8,9'

@attribute polarity numeric
@attribute entropy numeric
@attribute length numeric
@attribute keywords_frequency numeric
@attribute excessive_punc numeric
@attribute caps numeric
@attribute avg_word numeric
@attribute avg_sentence numeric
@attribute output {0,1}

@data
-0.906025,3.958229,27,0,0.75,0,8.333333,-0.25,0
-0.820514,3.235926,14,0,0,0,6.5,0,0
-0.820514,3.834963,24,0,0,0,7.333333,0,0
-0.906025,4.161978,26,0,0,0,26,0.5,0
-0.820514,4.213661,25,0,0,0,25,0.5,0
-0.820514,3.653757,21,0,1,0,6.333333,0.75,0
-0.906025,4.798269,62,0,0,0.142857,6.875,1,0
-0.906025,2.75,8,0,0,0,8,1,0
-0.906025,2,4,0,0,0,4,1,0
-0.906025,2.75,8,0,0,0,8,1,0
-0.906025,-0,1,0,0,0,1,1,0
-0.906025,2,4,0,0,0,4,1,0
-0.906025,3.121928,10,0,0,0,10,1,0
-0.820514,2,4,0,0,0,4,1,0
-0.820514,2,4,0,0,0,4,1,0
-0.820514,-0,1,0,0,0,1,1,0
-0.820514,3.121928,10,0,0,0,10,1,0
-0.820514,2.641604,9,0,0,0,9,1,0
-0.820514,3,8,0,0,0,8,1,0
-0.820514,2.807355,7,0,0,0,7,1,0
```

Рисунок 3.23 – Фрагмент даних для навчання моделі.

```
@relation telegram_dataset_full

@attribute polarity numeric
@attribute entropy numeric
@attribute length numeric
@attribute keywords_frequency numeric
@attribute excessive_punc numeric
@attribute caps numeric
@attribute avg_word numeric
@attribute avg_sentence numeric
@attribute output {0,1}

@data
-0.999997,4.500894,566,0,0,0,6.363636,11.833333,1
-0.999969,4.682291,522,0,0,0,5.5375,7.888889,1
-0.999968,4.459212,534,0,0,0,6.039474,18,1
-0.999967,4.566511,424,0,0.285714,0,5.854839,15.5,1
-0.999961,4.54158,506,0,0.0625,0,5.671053,12.666667,1
-0.999943,4.681172,353,0,0,0,5.555556,13.5,1
-0.999913,4.711387,607,0,0,0,6.238095,12,1
-0.999899,4.559827,336,0,0,0,6.659091,7.8,0
-0.999892,4.973442,1175,0,0,0.018634,5.758621,15.818182,1
-0.999866,4.943367,1083,0,0,0.014706,6.475862,12.083333,1
-0.999863,4.657127,1040,0,0,0,5.94,15,1
-0.999859,4.597374,287,0,0.4,0,6.2,13.333333,1
-0.999827,4.951491,742,0,0,0.03125,6.42,11.625,1
-0.999823,4.69323,1015,0.000985,0,0.013158,5.471338,22.428571,1
-0.999818,4.554557,409,0,0,0.015385,5.119403,16.75,1
-0.999803,4.590975,598,0,0.15,0.011765,5.655556,8,1
```

Рисунок 3.24 – Фрагмент даних для тестування моделі.

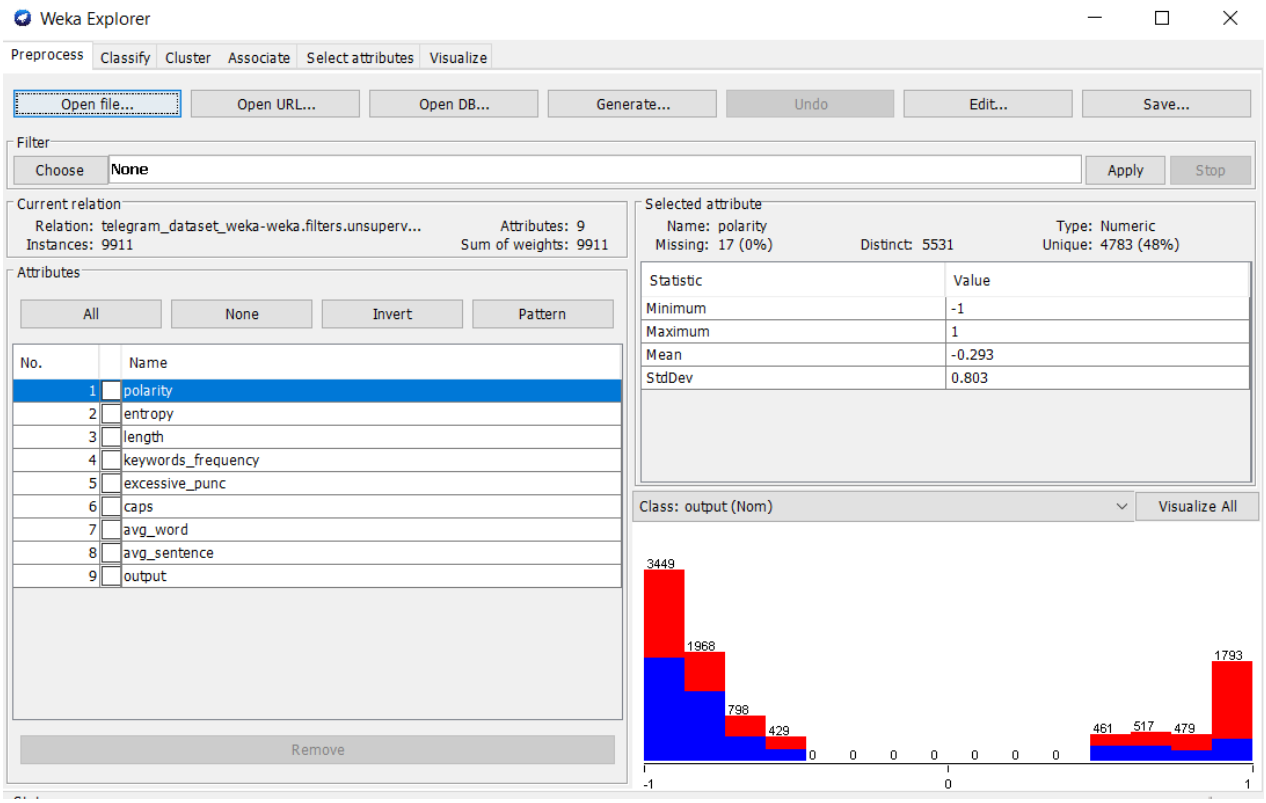

Рисунок 3.25 – Набір даних для навчання, завантажені у Weka.

Розподіл характеристик текстів у датасеті зображено на рис. 3.26.

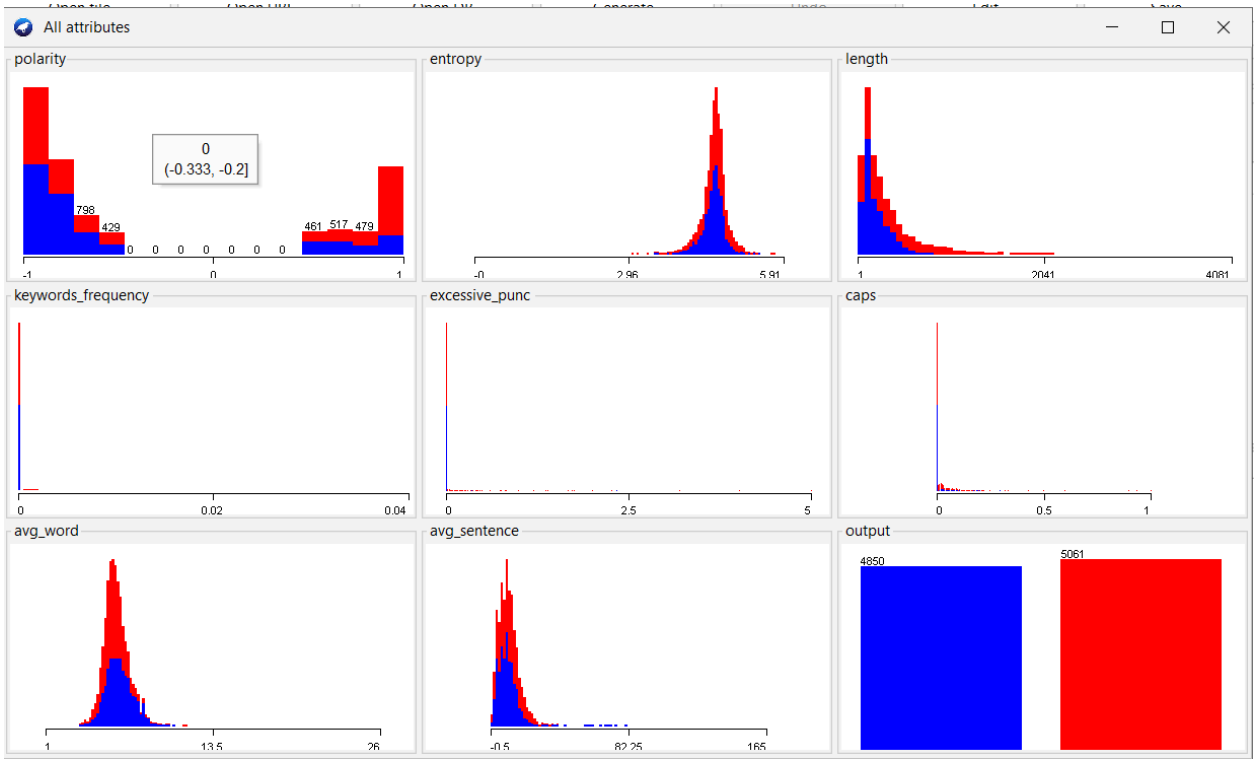

Рисунок 3.26 – Розподіл характеристик текстів у датасеті.

На рисунку 3.27 наведені налаштування класифікатора K-Star, використані для навчання мережі.

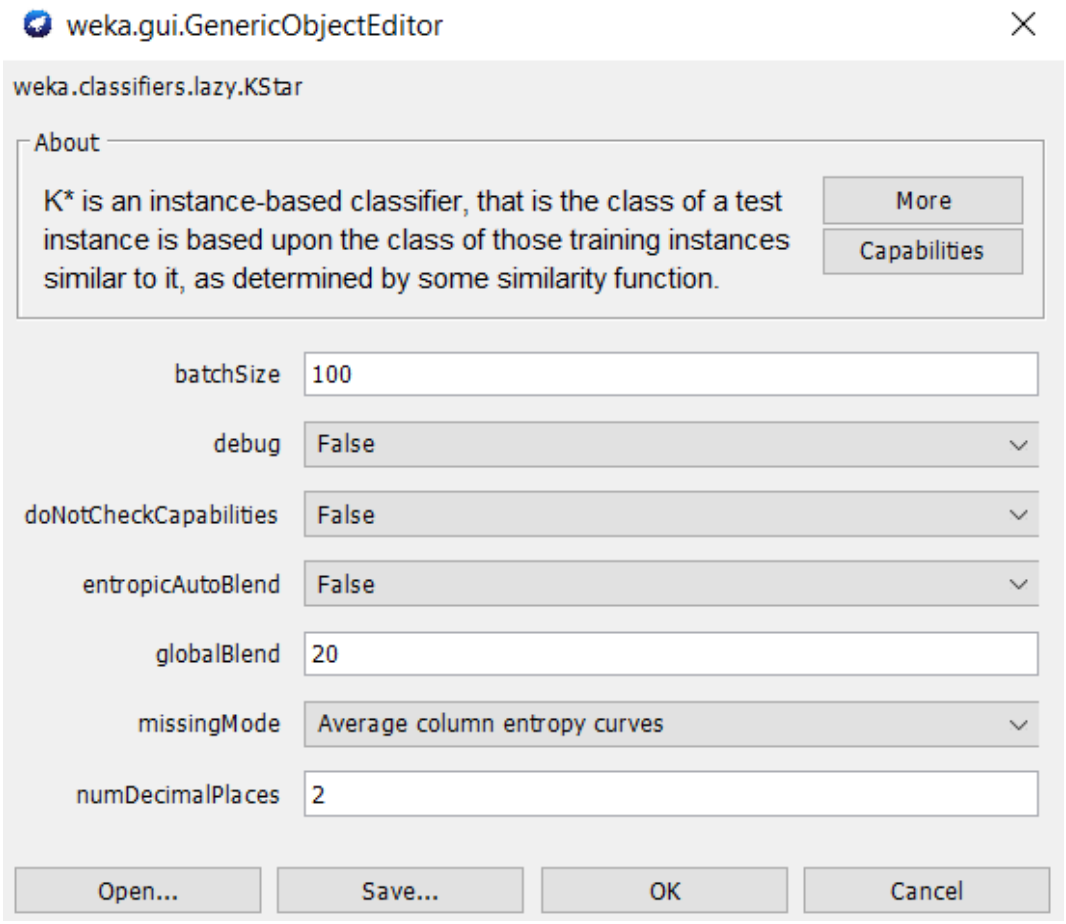

Рисунок 3.27 – Налаштування класифікатора K-Star.

Розмір даних дорівнює 100. Навчання мережі відбувається поетапно, 100 записів за одну ітерацію.

Параметри debug та doNotCheckCapabilities дорівнюють False. Параметр doNotCheckCapabilities визначає, чи має проводитися перевірка можливостей (capabilities) алгоритму перед використанням. Це процес перевірки того, чи підходять дані для аналізу та чи може використовуватися певний алгоритм для обробки цих даних. Якщо цей параметр встановлено на “true”, то перевірка можливостей не проводиться.

Параметр entropicAutoBlend встановлений як False. Параметр entropicAutoBlend дозволяє нейронній мережі автоматично розраховувати оптимальне значення параметру змішування між критерієм крос–ентропії та критерієм ентропії. Критерій

крос-ентропії та критерій ентропії – це два можливі підходи для підрахунку помилок в нейронній мережі під час тренування. Критерій крос-ентропії є більш складним та точним методом.

Параметр globalBlend дорівнює 20. У нейронних мережах існує концепція глобальних і локальних моделей. Глобальні моделі відповідають за загальну обробку вхідних даних та роблять загальні висновки про них. Локальні моделі відповідають за деталізацію роботи з вхідними даними і зазвичай використовуються для визначення деталей. Параметр globalBlend відповідає за те, яка частка вхідних даних повинна бути оброблена глобальною моделлю, а яка – локальними моделями.

Параметр missingMode відповідає за обробку відсутніх значень (missing values) в даних під час тренування та тестування моделі. Обидві вибірки повинні мати повністю заповнені поля, для запобігання помилок у навчанні була залишена опція за замовчуванням.

Кількість знаків після коми дорівнює 2.

Результат навчання моделі з набором даних в 100 000 записів наведений на рисунку 3.28.

```
KStar Beta Verion (0.1b).
Copyright (c) 1995-97 by Len Trigg (trigg@cs.waikato.ac.nz).
Java port to Weka by Abdelaziz Mahoui (am14@cs.waikato.ac.nz).

KStar options : -B 20 -M a

Time taken to build model: 0 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 390.9 seconds

=== Summary ===

Correctly Classified Instances       18219               81.5752 %
Incorrectly Classified Instances      4115               18.4248 %
Kappa statistic                          0.5847
Mean absolute error                      0.2173
Root mean squared error                  0.3487
Relative absolute error                 43.9927 %
Root relative squared error             70.5883 %
Total Number of Instances            22334

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.998    0.235    0.541      0.998    0.702      0.642    0.987     0.955     0
                 0.765    0.002    0.999      0.765    0.867      0.642    0.987     0.997     1
Weighted Avg.    0.816    0.053    0.900      0.816    0.831      0.642    0.987     0.988

=== Confusion Matrix ===

     a     b   <-- classified as
  4839    11 |     a = 0
  4104 13380 |     b = 1
```

Рисунок 3.28 – Результат навчання моделі і її перевірки на тестовій вибірці.

Програмою був створений файл з розширенням .model. Даний файл буде використовуватися для класифікації нових даних. Коректно прокласифікованими на тестовій вибірці виявились 81,5752% статей. Значення Area Under The Curve (ROC Area) дорівнює 0.987, що також свідчить про високу точність моделі.

Точність програми в результаті експерименту можна вважати задовільною, адже навряд чи кожна стаття на відповідному сайті може бути маніпуляторською або не маніпуляторською. Враховуючи це, відсоток співпадінь може бути вище.

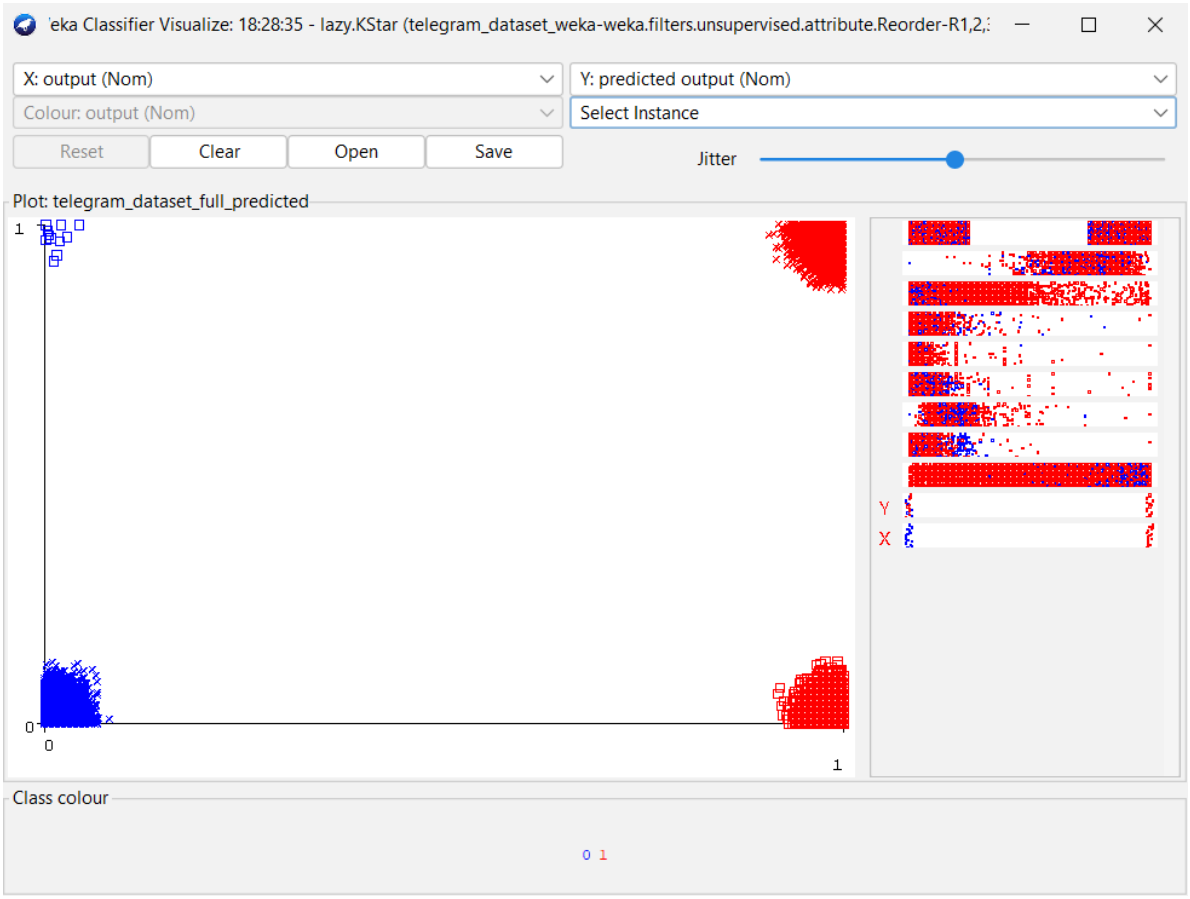

Рисунок 3.29 – Результат навчання моделі і її перевірки на тестовій вибірці.

На рисунках 3.30-3.36 наведена візуалізація характеристик натренованої моделі, включаючи MarginCurve, ThresholdCurve, Cost/Benefit Analysis та CostCurve.

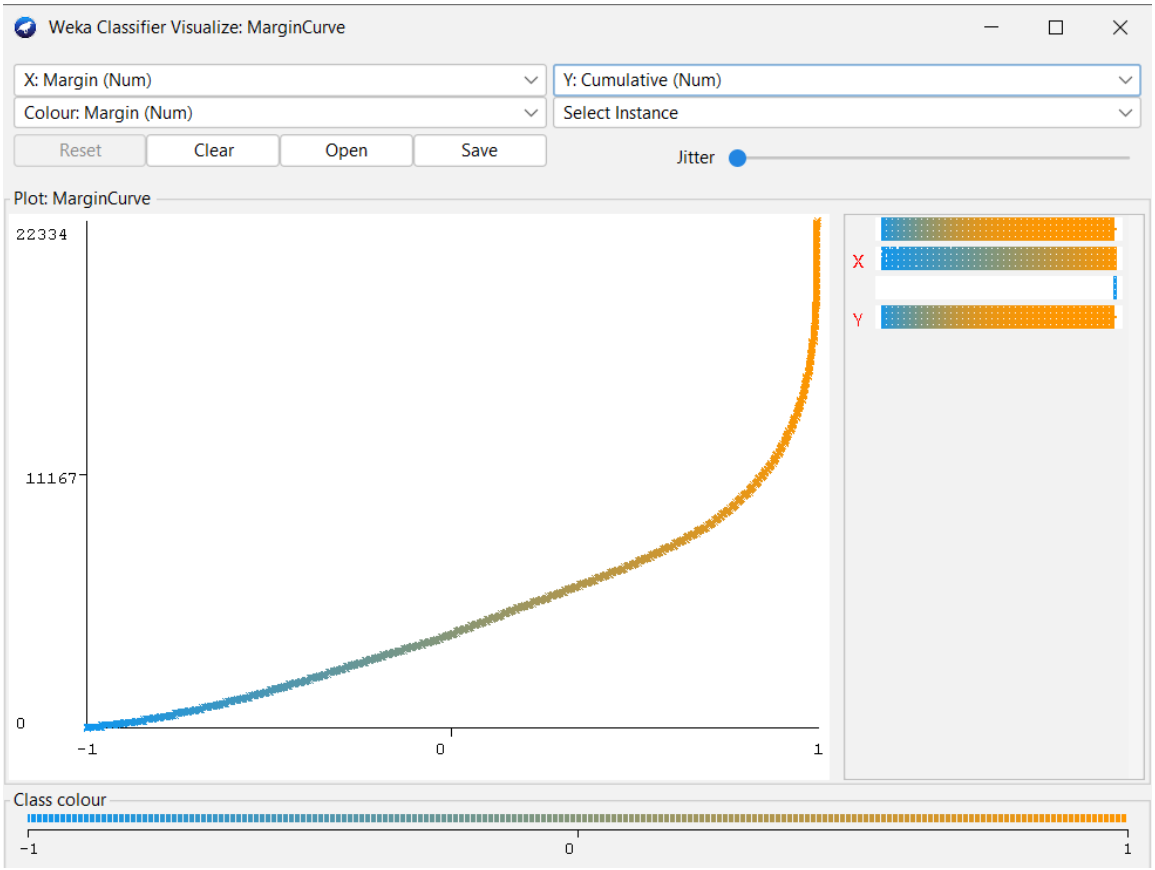

Рисунок 3.30 – Графік MarginCurve.

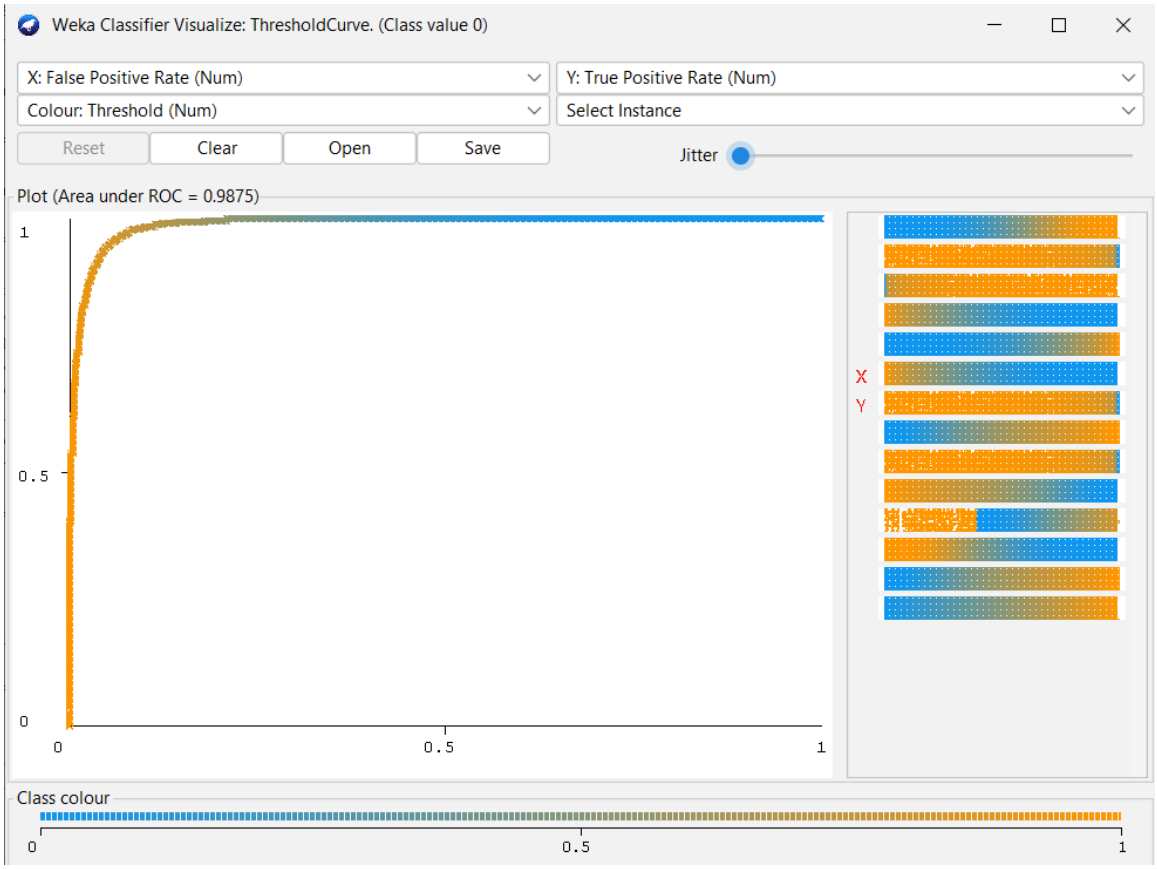

Рисунок 3.31 – Графік ThresholdCurve для класу 0.

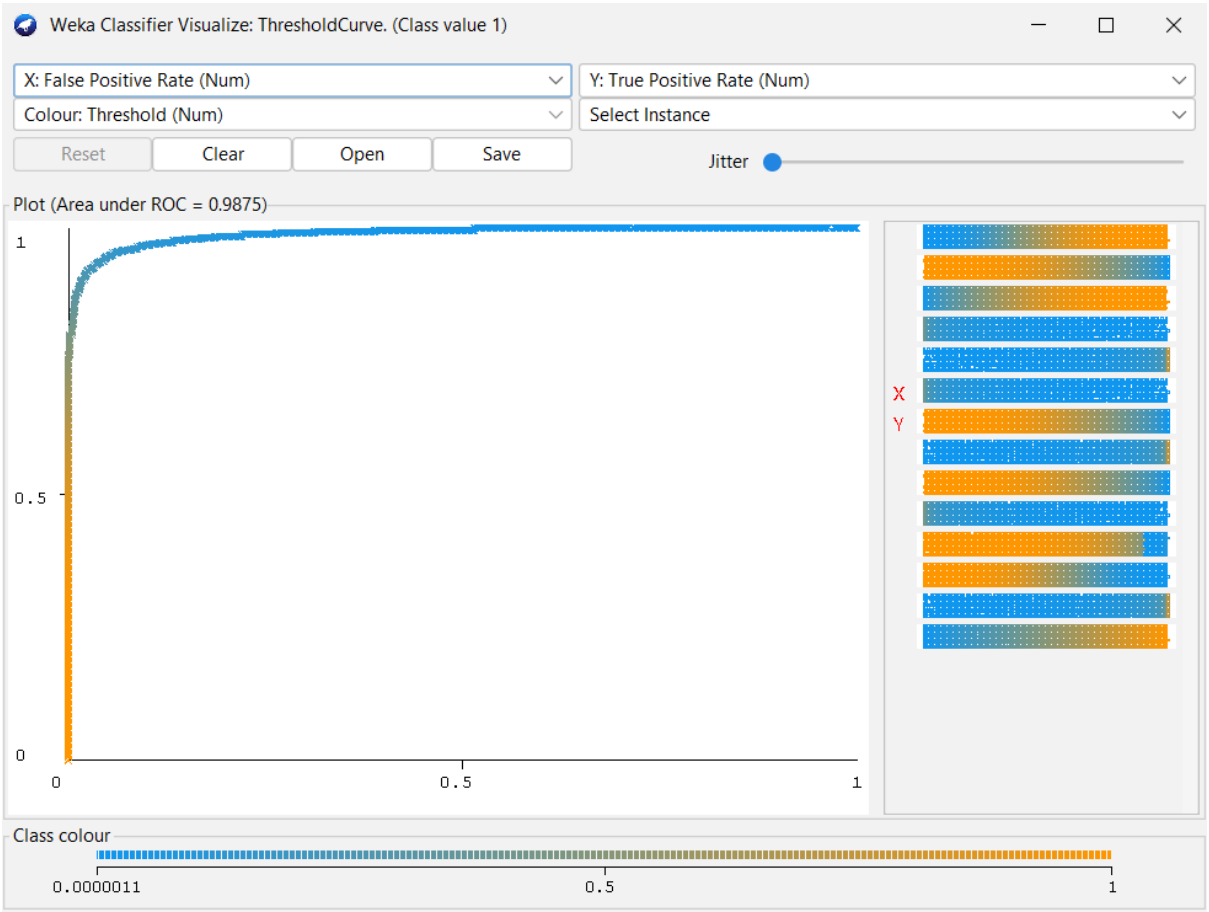

Рисунок 3.32 – Графік ThresholdCurve для класу 1.

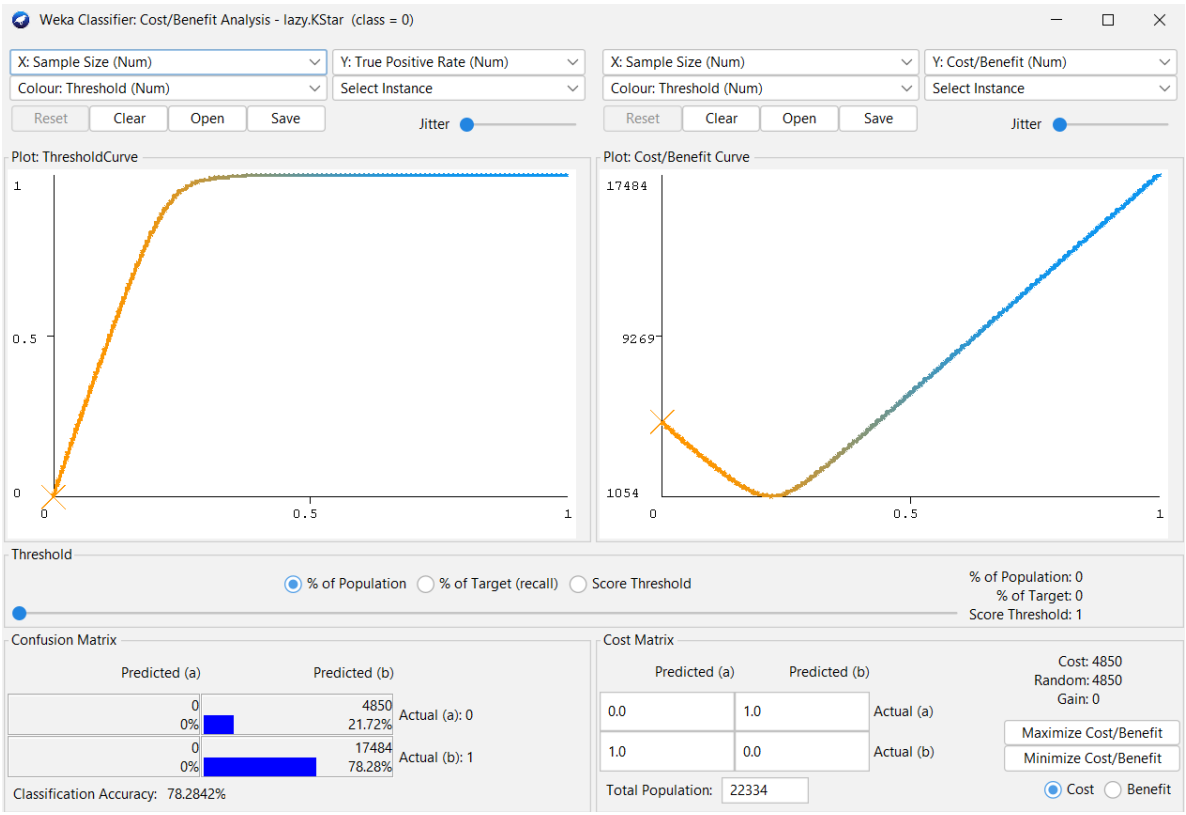

Рисунок 3.33 – Графік Cost/Benefit Analysis для класу 0.

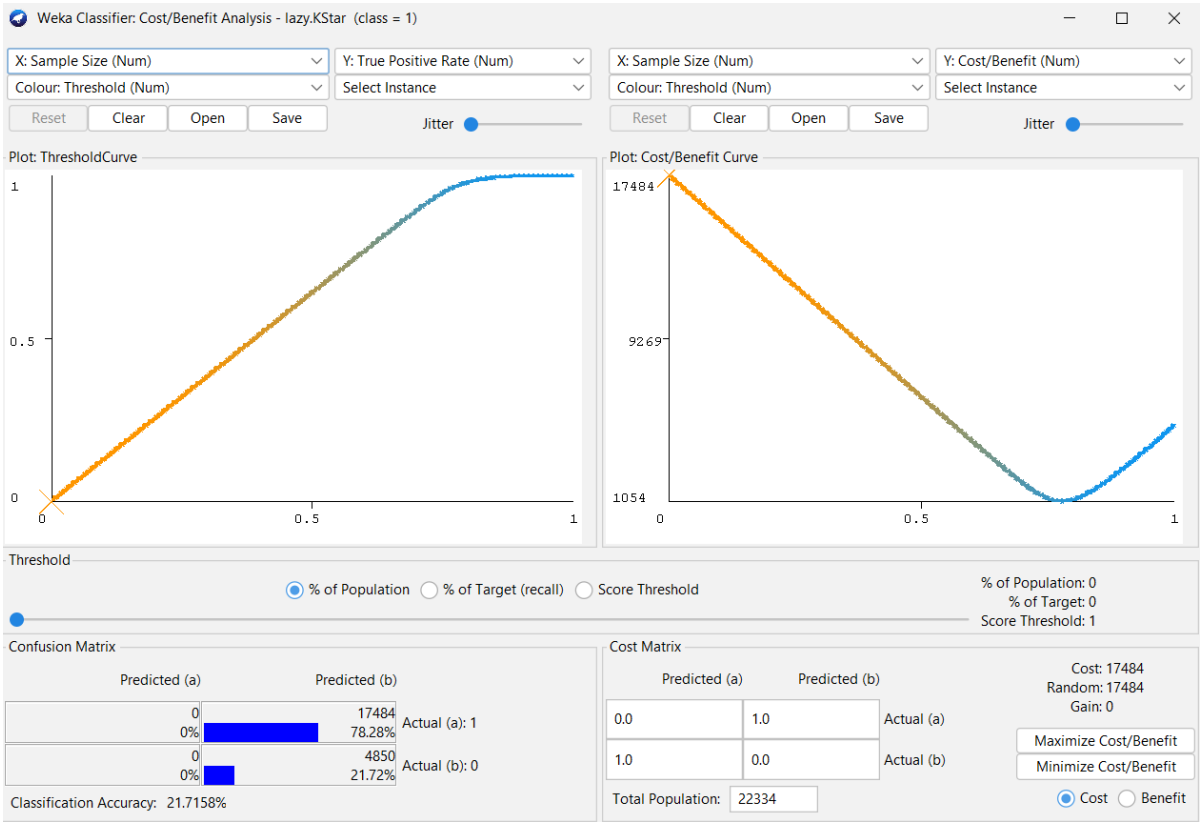

Рисунок 3.34 – Графік Cost/Benefit Analysis для класу 1.

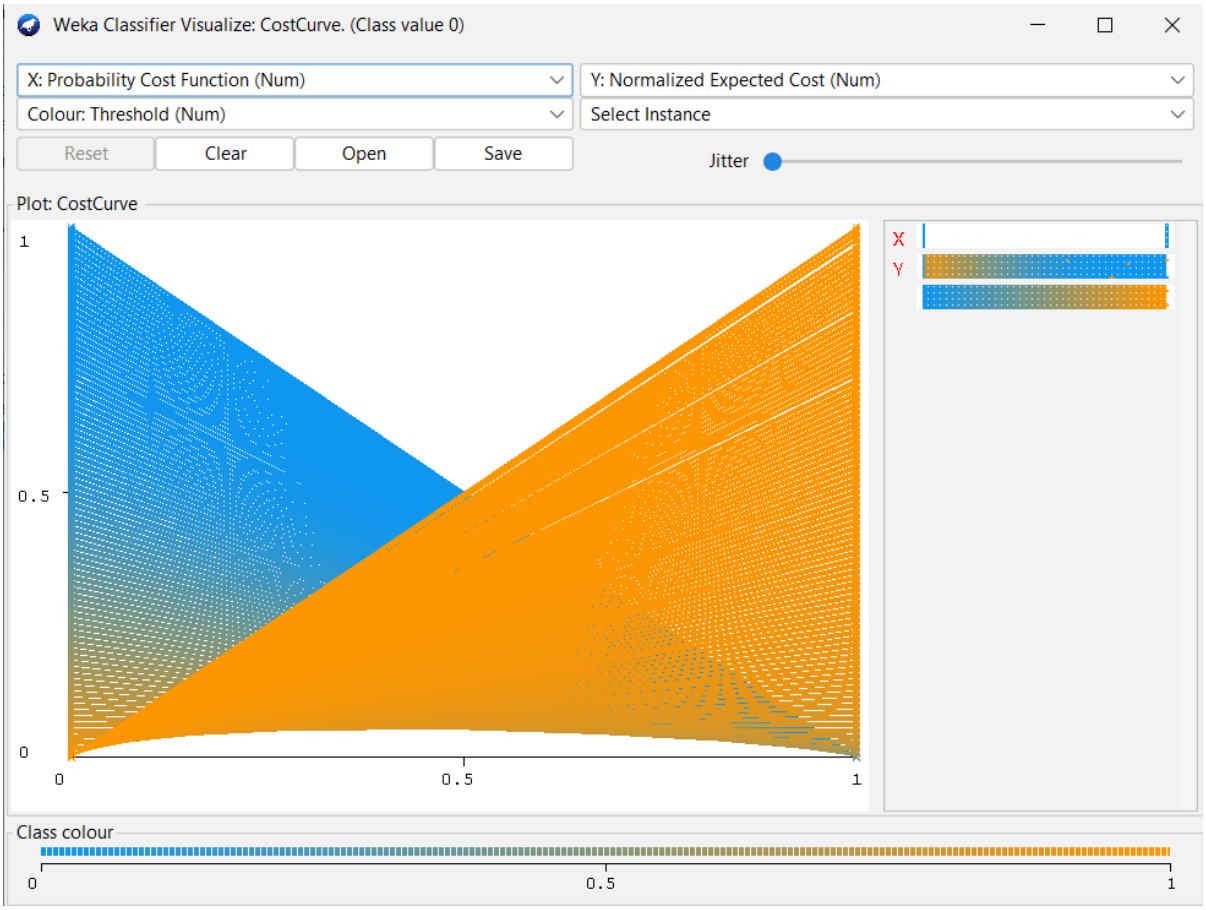

Рисунок 3.35 – Графік CostCurve для класу 0.

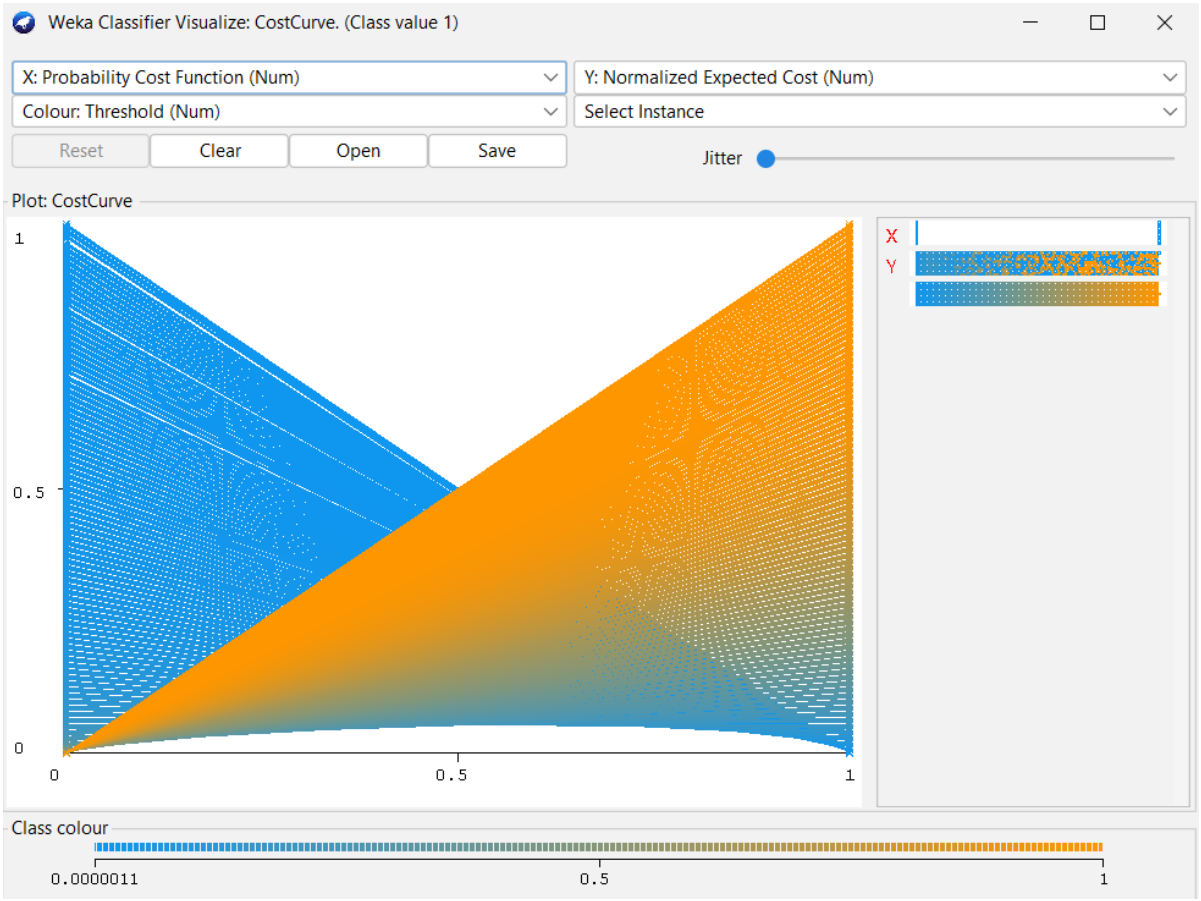

Рисунок 3.36 – Графік CostCurve для класу 1.

3.7 Кластеризація даних та аналіз результатів

Кластеризація даних була виконана на основі карт Коханана. Для їх використання у ПЗ Weka був встановлений пакет під назвою SelfOrganizingMap.

Кількість кластерів дорівнює 4 для забезпечення інформативності кластеризації.

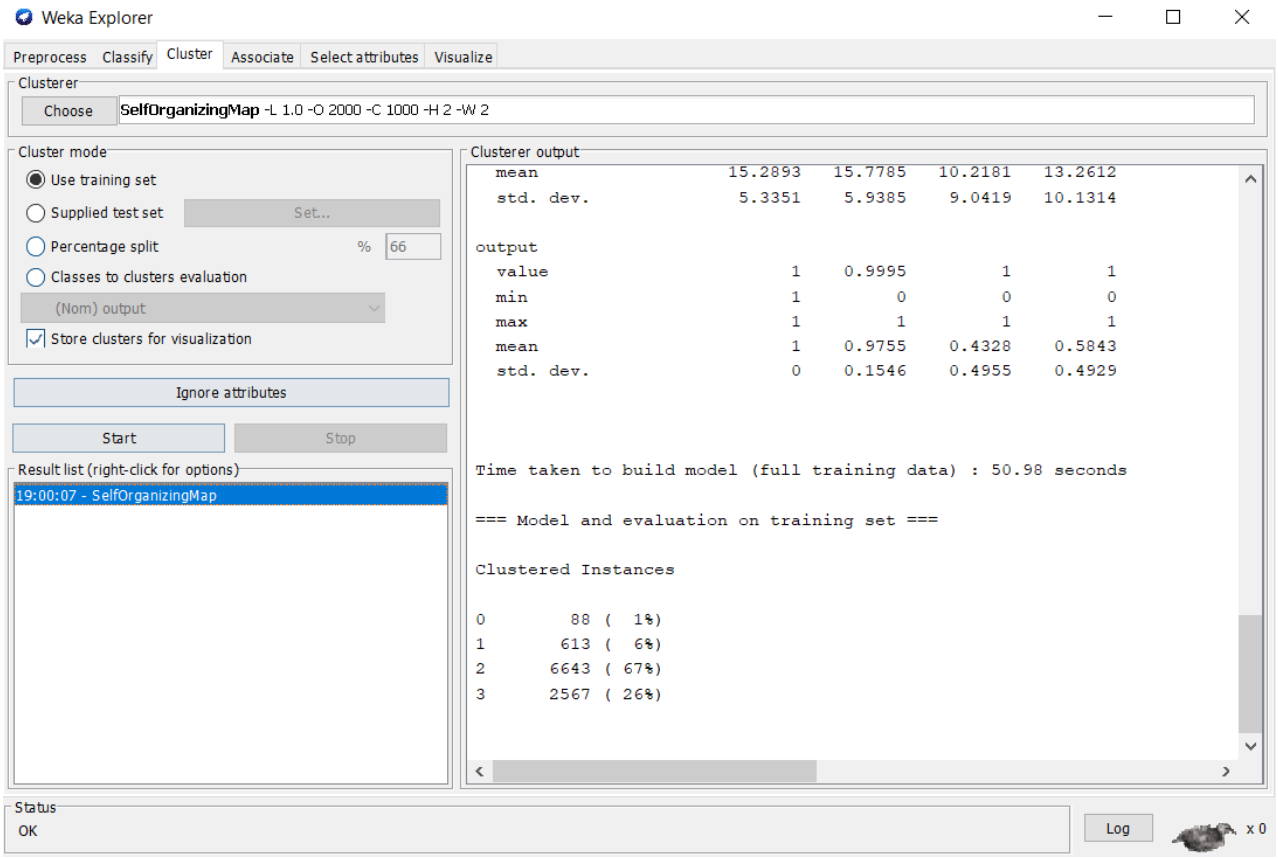

Рисунок 3.37 – Налаштування кластеризації за допомогою SelfOrganizingMap.

На рисунках 3.38, 3.39 наведені співвідношення характеристик тексту до output (характеристики, що показує, чи є текст маніпулятивним).

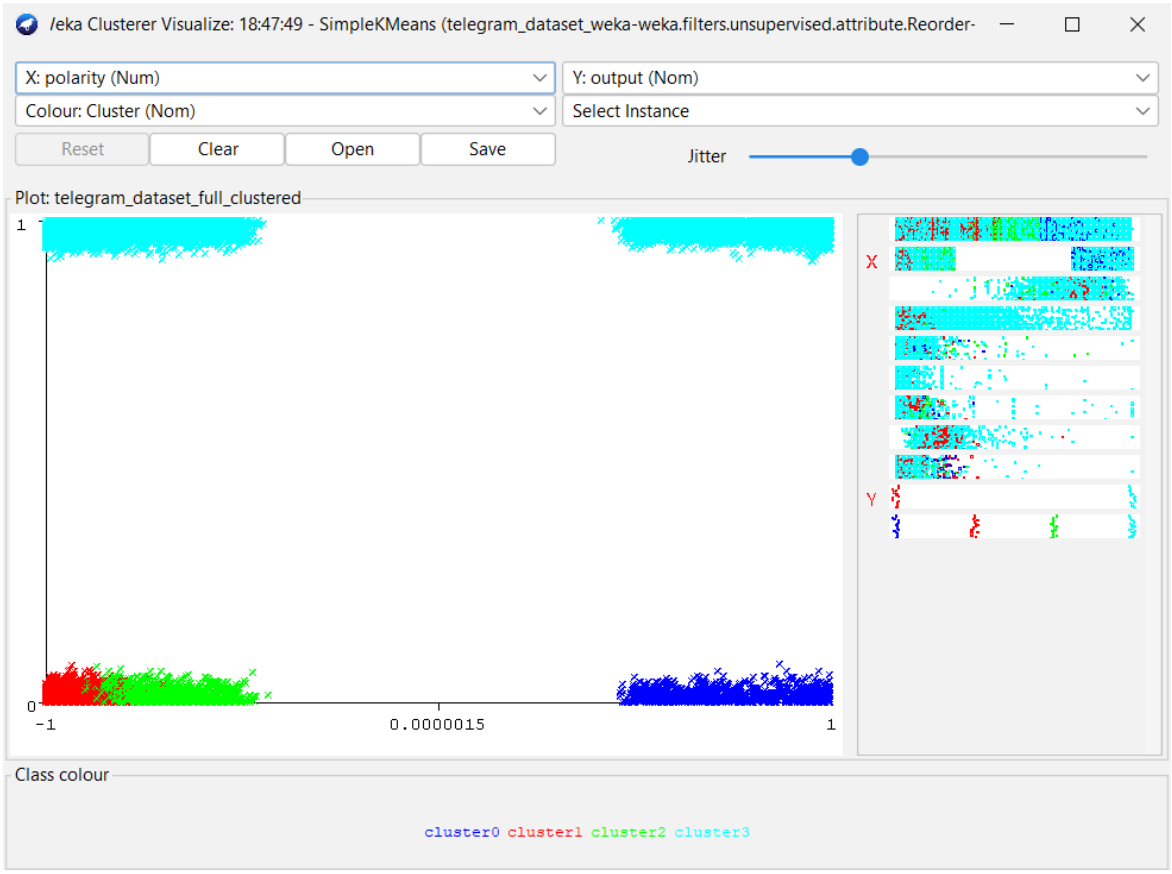

Рисунок 3.38 – Співвідношення polarity та output.

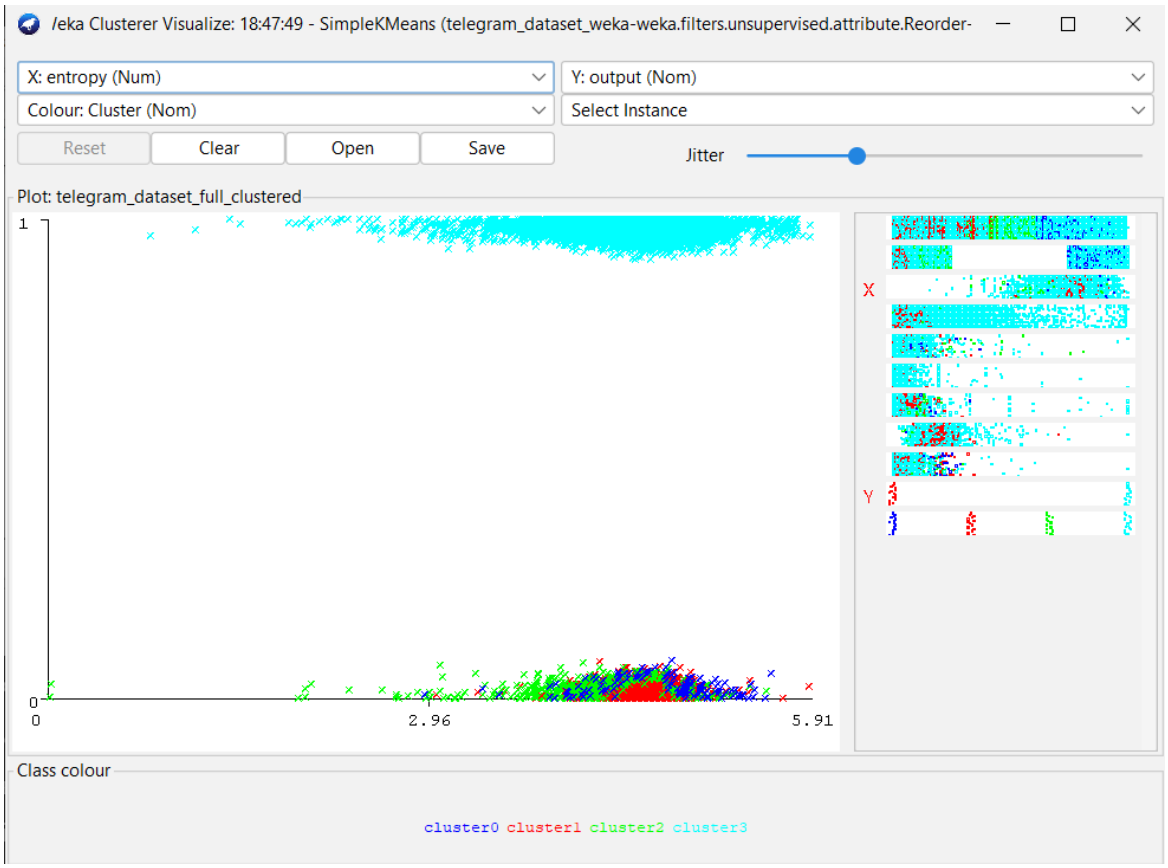

Рисунок 3.39 – Співвідношення entropy та output.

З візуалізації можна дійти висновку, що ймовірність того, що текст є маніпулятивним або не маніпулятивним при певному значенні полярності або ентропії є приблизно рівними.

Натомість, рисунок 3.40 демонструє, що при збільшенні довжини статті збільшується ймовірність того, що вона маніпулятивна.

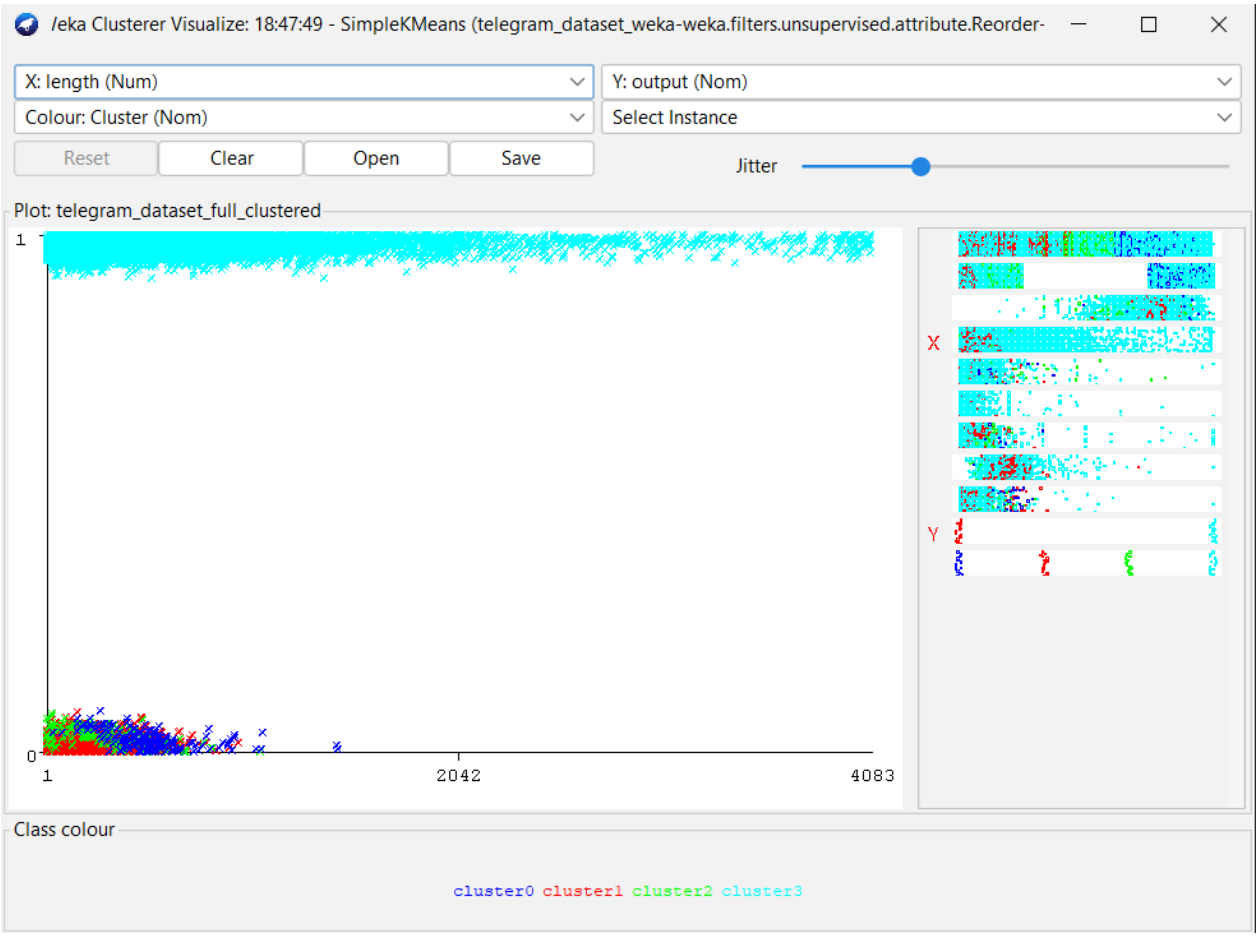

Рисунок 3.40 – Співвідношення length та output.

Рисунок 3.41 – Співвідношення keywords_frequency та output.

Рисунок 3.42 – Співвідношення excessive_punc та output.

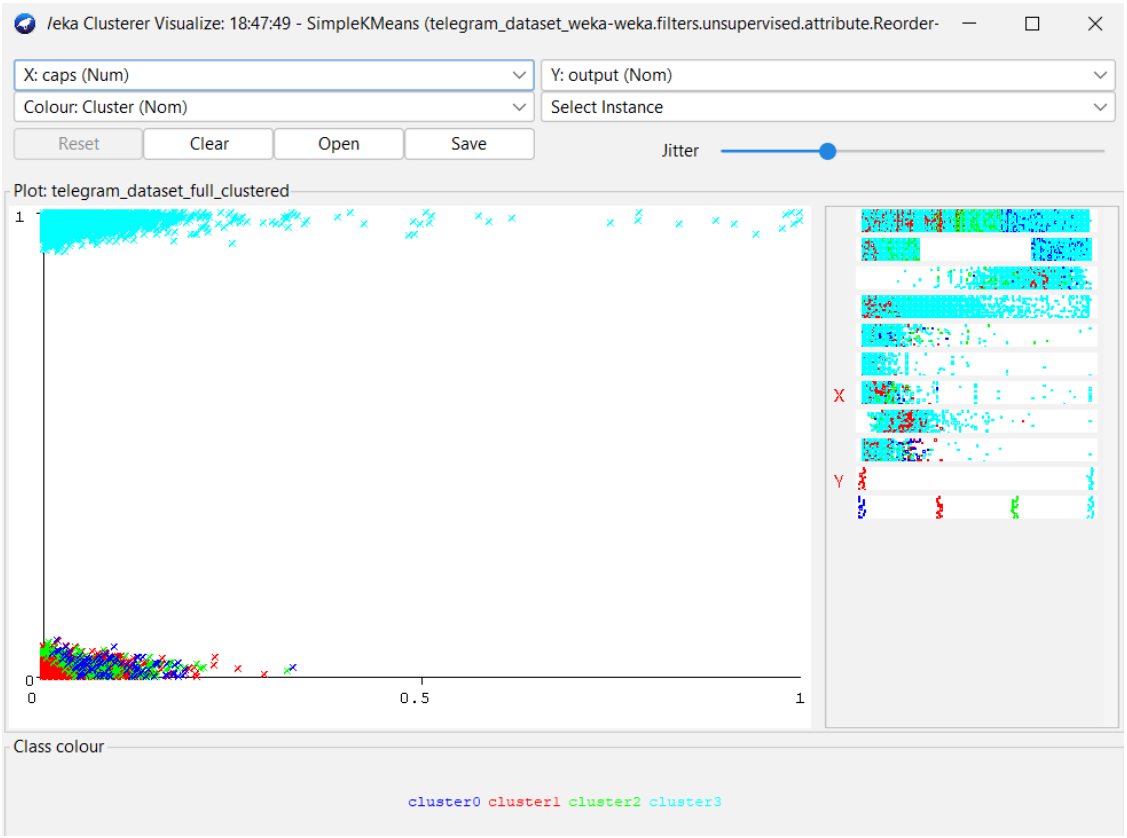

Рисунок 3.43 – Співвідношення caps та output.

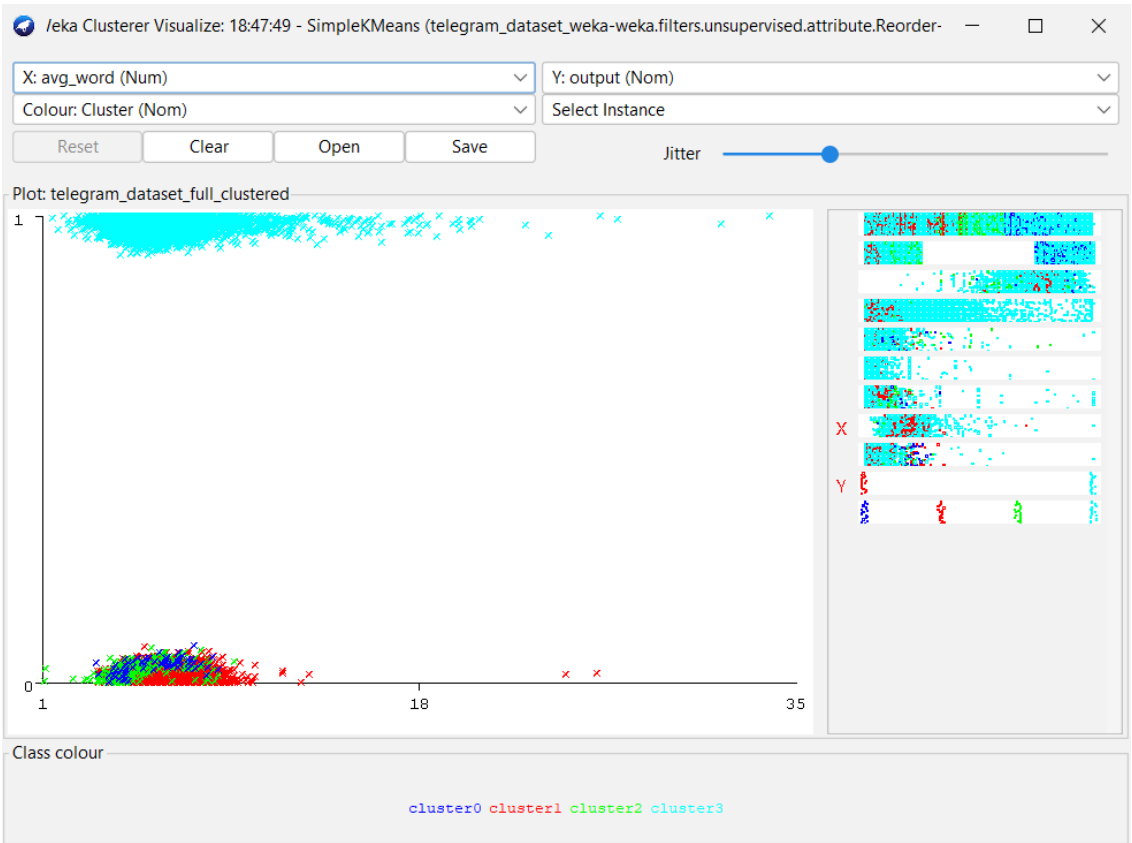

Рисунок 3.44 – Співвідношення avg_word та output.

Рисунок 3.45 – Співвідношення avg_sentence та output.

3.8 Перевірка роботи нейронної мережі

Створеній нейронній мережі був наданий випадковий текст для аналізу та визначення класу (маніпуляцятивний, не маніпулятивний). У якості такого прикладу був використаний запис з Телеграм-каналу “Новинач” за 30 травня, 20:55.

Рисунок 3.46 – Оригінал посту.

Для початку аналізу тексту створена .csv-файл, який буде містити відповідні характеристики тексту.

	A	B	C	D	E	F	G	H
1	message							
2	Кабмін ухвалив рішення щодо підвищення з 1 червня тарифів на електроенергію до 2,64 грн за кВт-год. За словами міністра енергетики Галущенка, підвищення тарифів спричинили руйнування і пошкодження енергетичної інфраструктури							
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								

Рисунок 3.47 – Створений .csv-файл.

Після завантаження тексту у Python-скрипт був отриманий новий файл, з повністю обрахованими характеристиками.

	A	B	C	D	E	F	G	H	I	J
1	message	polarity	entropy	length	keywords_frequenc	excessive_punc	caps	avg_word	avg_sentence	output
2	Кабмін ухвалив рішення щодо підвищення з	-0.942688923	4.611709059	224	0	0	0	6.75862069	14.5	0

Рисунок 3.48 – csv-файл з обчисленими характеристиками тексту.

Ці характеристики були занесені у .arff-файл – розширення, з яким працює програмне середовище WeKa.

NovinachTestEvaluated.arff - Notepad

File Edit Format View Help

```
@relation NovinachTestEvaluated

@attribute polarity numeric
@attribute entropy numeric
@attribute length numeric
@attribute keywords_frequency numeric
@attribute excessive_punc numeric
@attribute caps numeric
@attribute avg_word numeric
@attribute avg_sentence numeric
@attribute output {0,1}

@data
-0.9426889233980209,4.611709059100829,224,0.0,0.0,0.0,6.758620689655173,14.5,0
```

Рисунок 3.49 – .arff-файл з обчисленими характеристиками тексту.

Даний “датасет”, що складався лише з характеристик одного тексту, був завантажений до створеної нейронної мережі у якості тестового. В такому випадку, натренована мережа перевіряла передбачений клас даного тексту з фактичним.

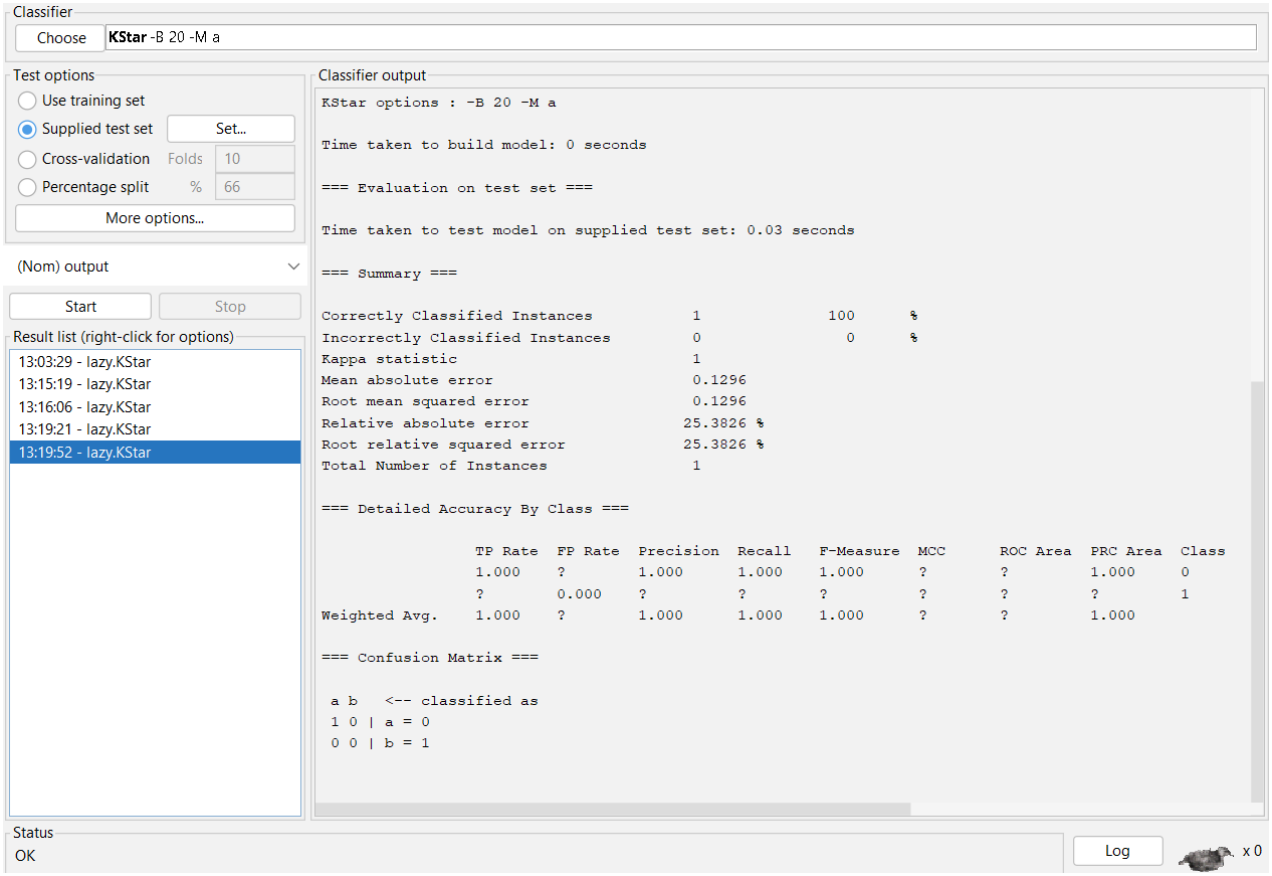

Рисунок 3.50. Результат перевірки передбачення класу тексту.

В результаті перевірки роботи мережі відсоток коректної класифікації дорівнював 100% – тобто, клас тексту з Телеграм-каналу “Новинач” був передбачений правильно.

Висновки до розділу 3

Даний розділ описував практичну реалізацію нейронної мережі. Першим етапом створення нейронної мережі була підготовка датасетів – текстів та їх обчислених характеристик. В якості текстів були обрані записи з політичних Телеграм-каналів, їх список наведений у додатку В. У середовищі PyCharm був розроблений скрипт з автоматичного копіювання від 1000 записів з кожного каналу та створення окремого .csv-файлу з ними. Під час створення цього датасету з текстів також вилучались символи переносу, табуляції та інші, що могли призвести до неправильного обчислення їх характеристик.

У середовищі PyCharm на окремих словниках, що містили позитивно та негативно забарвлені слова української та російської мов, був натренований власний класифікатор. Його призначення було обчислення характеристики polarity з огляду на слова, що знаходяться у кожному реченні тексту. Інші формули, що були наведені у розділі 2, також були реалізовані за допомогою модуля Python math, pandas (для створення .csv-файлів), langdetect (для автоматизованого визначення мови) та re (регулярних виразів).

В результаті були створені два датасети. Один для навчання моделі – 9911 записів (з приблизно рівною кількістю маніпулятивних та не-маніпулятивних статей), один для тестування моделі – 22335 записів. При завантаженні даних у WeKa було помічено, що розподіл характеристик між маніпулятивними та неманіпулятивними статтями порівняно однаковий.

Були наведені характеристики класифікатора K-Star, що використовувались при тренуванні моделі. Перевірка роботи моделі показала точність у 81,5752 %. Після створення, характеристики мережі були проаналізовані та виконано кластеризацію вихідних даних. За результатами кластеризації було зроблено висновок, що довжина тексту є найбільш показовою характеристикою з усіх використаних, інші характеристики порівняно не відрізняються між собою у маніпулятивних і неманіпулятивних статей.

Роботу мережі було перевірено на абсолютно незнайомій їй записі, що в результаті був коректно прокласифікований.

ВИСНОВОК

Результати кваліфікаційної роботи є актуальними та сприятливими для вирішення питання підвищення рівня інформаційної безпеки країни та автоматизації процесу класифікації текстового контенту.

Для створення моделі класифікації тексту були розглянуті теоретичні засади машинного навчання та нейронних мереж. Було виконано аналіз та порівняння парадигм машинного навчання, обрана найбільш доцільна парадигма для досягнення необхідного функціоналу нейронної мережі. Окремо були названі та проаналізовані числові характеристики тексту, що можуть бути використані в їх класифікації. Вони відрізняються в залежності від сфери використання тексту. Під час порівняння кожної з характеристик були оцінені їх інформативність та доцільність у обчисленні.

Модель нейронної мережі була формально визначена, перед етапом її практичного втілення були розроблені підходи до обчислення обраних характеристик тексту. Після аналізу відкритих джерел в якості навчальних вибірок були обрані статті з переліку Телеграм-каналів, що поширювали дезінформацію, або навпаки, були джерелами якісної інформації. З огляду на майбутніх функціонал мережі, був обраний класифікатор K-Star, що найкраще описував кінцеві дані, відповідно, результати класифікації з його використанням були найбільш точними.

Для реалізації нейронної мережі був створений проєкт у PyCharm. Був розроблений скрипт для парсингу Телеграм-каналів, що дозволив швидко зібрати статті для навчання та тестування нейронної мережі. Для обчислення характеристик були власноруч створені словники абревіатур, ключових слів російської пропаганди, емоційно забарвленої російської лексики. Ці словники використовувались для навчання класифікаторів модулю TextBlob, що обчислювали характеристику polarity текстів. Інші характеристики також були обчислені під час обробки текстів у PyCharm.

В результаті були створені два датасети. Один для навчання моделі – 9911 записів (з приблизно рівною кількістю маніпулятивних та не-маніпулятивних статей), один для тестування моделі – 22335 записів. Після тренування точність мережі склала

81,5752 %. Перевірка роботи мережі на повністю нових даних підтвердила працездатність мережі.

Процес кластеризації виявив, що більшість числових параметрів є малоінформативними для аналізу наміру тексту. Проте, була помічене співвідношення між довжиною і результатами класифікації: зі зростанням довжини тексту ймовірність його маніпулятивності збільшується.

При подальшому дослідженні бажано розширити набір використовуваних характеристик, можливо, використовувати мета-дані, такі як оцінки користувачів тощо. Слід попередньо виконати більш детальний аналіз інформативності характеристик. Також можливе використання двоетапної класифікації, або іншого способу підвищити її комплексність.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Проєкт Закону України від 28.05.2014 № 4949 (Одержаний ВР України) “Про засади інформаційної безпеки України” [Електронний ресурс] // ips.ligazakon.net. – 2014. – Режим доступу до ресурсу: https://ips.ligazakon.net/document/JG3TH00A?an=3.

2. PSYCHOLOGICAL ASPECTS OF MANIPULATION WITHIN AN INTERPERSONAL INTERACTION: MANIPULATIONS AND MANIPULATORS [Електронний ресурс] // researchgate.net. – 2020. – Режим доступу до ресурсу: https://www.researchgate.net/publication/344540018_PSYCHOLOGICAL_ASPECTS_OF_MANIPULATION_WITHIN_AN_INTERPERSONAL_INTERACTION_MANIPULATIONS_AND_MANIPULATORS.

3. Рябчик А. В. Методи психологічного впливу в рекламі. Ефективна економіка. 2018. № 11. – URL: http://www.economy.nayka.com.ua/?op=1&z=6695 (дата звернення: 05.03.2023). DOI: 10.32702/2307-2105-2018.11.92

4. Morgan S. Fake news, disinformation, manipulation and online tactics to undermine democracy [Електронний ресурс] / Susan Morgan // Journal of Cyber Policy. – 2018. – Режим доступу до ресурсу: https://www.tandfonline.com/doi/full/10.1080/23738871.2018.1462395.

5. Нестеряк Ю. Державна підтримка ЗМІ: європейські традиції та українська практика // Вісник Київського національного університету. Журналістика. – 2002. – № 10. – С. 50-52.

6. Khanday A. Detecting Textual Propaganda Using Machine Learning Techniques [Електронний ресурс] / A. Khanday, Q. Rayees Khan, S. Tanzeel Rabani // researchgate.net. – 2020. – Режим доступу до ресурсу: https://www.researchgate.net/publication/347296550_Detecting_Textual_Propaganda_Using_Machine_Learning_Techniques.

7. Evans D. J. Parallel Distributed Neural Networks for Classification. / D. J. Evans L. P. Tay // Parallel Algorithms and Applications – 1995. – №3. – С. 293-305.

8. Text classification in memristor-based spiking neural networks [Електронний ресурс] / J.Huang, A. Serb, S. Stathopoulos, T. Prodromakis // researchgate.net. – 2023. – Режим доступу до ресурсу: https://www.researchgate.net/publication/367118466_Text_classification_in_memristor-based_spiking_neural_networks.

9. Research on Text Classification based on Deep Learning [Електронний ресурс] / L.Zhu, B. He, X. Wang, H. Zhang // researchgate.net. – 2022. – Режим доступу до ресурсу: 362166798_Research_on_Text_Classification_based_on_Deep_Learning.

10. Achimescu V. Feeding the troll detection algorithm Informal flags used as labels in classification models to identify perceived computational propaganda [Електронний ресурс] / V. Achimescu, D. Sultanescu // researchgate.net. – 2020. – Режим доступу до ресурсу: https://www.researchgate.net/publication/344188577_Feeding_the_troll_detection_algorithm_Informal_flags_used_as_labels_in_classification_models_to_identify_perceived_computational_propaganda.

11. Кількість сайтів, які поширюють фейки про Україну, подвоїлася після вторгнення Росії – моніторинг [Електронний ресурс] // Радіо Свобода. – 2022. – Режим доступу до ресурсу: https://www.radiosvoboda.org/a/news-kilkist-saitiv-feiky-pro-ukrainu/31980447.html.

12. Maier S. R. Compassion Fatigue and the Elusive Quest for Journalistic Impact: A Content and Reader-Metrics Analysis Assessing Audience Response [Електронний ресурс] / Scott Maier // researchgate.net. – 2015. – Режим доступу до ресурсу: https://www.researchgate.net/publication/283935453_Compassion_Fatigue_and_the_Elusive_Quest_for_Journalistic_Impact_A_Content_and_Reader-Metrics_Analysis_Assessing_Audience_Response.

13. Satapathy R. Polarity and Subjectivity Detection with Multitask Learning and BERT Embedding [Електронний ресурс] / R. Satapathy, S. Pardeshi, E. Cambria // researchgate.net. – 2022. – Режим доступу до ресурсу: https://www.researchgate.net/publication/363008866_Polarity_and_Subjectivity_Detection_with_Multitask_Learning_and_BERT_Embedding.

14. Як відрізняти справжні новини від брехні, маніпуляцій і напівправди. Інструкція [Електронний ресурс] // texty.org.ua – 2021. – Режим доступу до ресурсу: https://texty.org.ua/articles/85998/Jak_vidriznaty_spravzhni_novyny_vid_brehni_manipulacij-85998/

15. Cialdini R. B. Influence: The Psychology of Persuasion, Revised Edition, 2006, p.354 / Robert B. Cialdini. – New York, 2006. – 279 c. – (HarperCollins Publishers Inc.).

16. O'Neil C. Doing data science: Straight talk from the frontline, 2013 / O'Neil C. Schutt R. – Sebastopol, 2013 – (O'Reilly Media, Inc.);

17. Mistake or Manipulation? Conceptualizing Perceived Mis- and Disinformation among News Consumers in 10 European Countries [Електронний ресурс] / M.Hameleers, A. Brosius, F. Marquart, A. Goldberg // researchgate.net. – 2021. – Режим доступу до ресурсу: https://www.researchgate.net/publication/350639361_Mistake_or_Manipulation_Conceptualizing_Perceived_Mis-_and_Disinformation_among_News_Consumers_in_10_European_Countries.

18. How One 'fact-Checking' Site Spreads Russian Propaganda [Електронний ресурс] // dw.com. – 2022. – Режим доступу до ресурсу: https://www.dw.com/en/ukraine-war-how-a-fact-checking-website-is-spreading-russian-propaganda/a-61062940.

19. Propaganda Techniques. / 2003. – (Holt Literature & Language Arts, Introductory Course). – С. 643-645.

20. Skupriienko. Ukrainian–Stopwords [Електронний ресурс] / Skuprienko // github.com. – 2021. – Режим доступу до ресурсу: https://github.com/skupriienko/Ukrainian-Stopwords.

21. Skupriienko, Ukrainian–Sentiment–Analysis [Електронний ресурс] / Skuprienko // github.com. – 2021. – Режим доступу до ресурсу: https://github.com/skupriienko/ Ukrainian–Sentiment-Analysis

22. Антонім Онлайн [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://slovnyk-antonimiv.com/.

23. Manjoo F. You Won't Finish This Article [Електронний ресурс] / Farhad Manjoo // Slate Magazine. – 2013. – Режим доступу до ресурсу:

https://slate.com/technology/2013/06/how-people-read-online-why-you-wont-finish-this-article.html.

24. Сотні тисяч. Яка аудиторія (про)російських медіа в Україні [Електронний ресурс] / Є.Дроздова, Н. Кельм, Ю. Дукач, Д. Судачек // texty.org.ua. – 2021. – Режим доступу до ресурсу: https://texty.org.ua/projects/103537/pro-audytoriyu-prorosijskyh-media-v-ukrayini/.

25. “IMI склав «білий список» загальнонаціональних сайтів з якісною інформацією.” [Електронний ресурс] // detector.media. – 2021 – Режим доступу до ресурсу: https://detector.media/infospace/article/185190/2021-02-24-imi-sklav-bilyy-spysok-zagalnonatsionalnykh-saytiv-z-yakisnoyu-informatsiieyu/.

26. «Кремлівська гідра»: 300 Телеграм-каналів, які отруюють український інфопростір [Електронний ресурс] // detector.media. – 2022 – Режим доступу до ресурсу: https://detector.media/monitorynh-internetu/article/205954/2022-12-14-kremlivska-gidra-300-telegram-kanaliv-yaki-otruyuyut-ukrainskyy-infoprostir/.

ДОДАТОК А

Список опублікованих праць за темою кваліфікаційної роботи

1. Бездольний В. Методи ідентифікації та протидії ворожим медіа-ресурсам у кіберпросторі / Юрій Ландовський, Тетяна Бабенко, Лариса Мирутенко / VI Міжнародна науково-практична конференція «Проблеми кібербезпеки інформаційно-телекомунікаційних систем» (PCSITS) 27 квітня 2023, Київ, Україна, стр. 130-131.

ДОДАТОК Б

Попередня обробка даних за допомогою Python

```
import pandas as pd
import ArticleProcessing
import UkrainianProcessing
import RussianProcessing
from langdetect import detect

ua_classifier = UkrainianProcessing.trainClassifier()
ru_classifier = RussianProcessing.trainClassifier()

df = pd.read_csv(test.csv', delimiter=";")
df2 =
pd.DataFrame(columns=['message','polarity','entropy','length','keyword
s_frequency','excessive_punc',
'caps','avg_word','avg_sentence','output'])

for i in range(len(df)):
    try:
        text = df['message'][i]
        length = len(text)
        excessive_punc = ArticleProcessing.excessivePunctuation(text)
        #uniqueness = ArticleProcessing.uniquenessCheck(text)
        entropy = ArticleProcessing.Entropy(text)
        text_for_keywords = text.lower()
        keywords_frequency =
ArticleProcessing.keywordFrequency(text_for_keywords)
        caps = ArticleProcessing.capsLock(text)
        avg_word = ArticleProcessing.averageWord(text)
        avg_sentence = ArticleProcessing.averageSentence(text)

        polarity = 0.0
        lang = detect(text)
        if lang == 'ua':
```

```
            print('Ukrainian!')
            polarity = UkrainianProcessing.classifyArticle(text,
ua_classifier)
        elif lang == 'ru':
            print('Russian!')
            polarity = RussianProcessing.classifyArticle(text,
ru_classifier)
        else:
            print('Ukrainian!')
            polarity = UkrainianProcessing.classifyArticle(text,
ua_classifier)
        # 1 = Маніпуляція
        # 0 = Не маніпуляція
        #list_row = [text, polarity, entropy, length,
keywords_frequency, excessive_punc, caps, avg_word, avg_sentence, 1]
        list_row = [text, polarity, entropy, length,
keywords_frequency, excessive_punc, caps, avg_word, avg_sentence, 0]
        print(list_row)
        df2.loc[len(df2)] = list_row
    except:
        continue

df2.to_csv('test_evaluated.csv', sep=';', encoding='utf-8-
sig',index=False)
```

ДОДАТОК В

Список використаних Телеграм-каналів

Маніпулятивні:

- https://t.me/ASupersharij
- https://t.me/legitimniy
- https://t.me/VasiletsDmitriy
- https://t.me/sheyhtamir1974
- https://t.me/montyan2
- https://t.me/olegtsarov
- https://t.me/OlgaSharij
- https://t.me/ukraina_ru
- https://t.me/ZeRada1
- https://t.me/KlymenkoTime
- https://t.me/Medvedeva_Olesya
- https://t.me/panchenkodi
- https://t.me/Varjag2007
- https://t.me/vadim_rabinovich
- https://t.me/Novoeizdanie
- https://t.me/stranaua
- https://t.me/spletnicca
- https://t.me/zp_live
- https://t.me/kharkov_trempel
- https://t.me/life_odessa
- https://t.me/chesno_movement
- https://t.me/mriya24
- https://t.me/kievadekvat

Не маніпулятивні:

- https://t.me/vmolnia
- https://t.me/novinach

ДОДАТОК Г

sentiment_ua_2.csv

;word;pos_neg
0;Будда;pos
1;Господь;pos
2;Магомет;pos
3;Немезида;neg
4;Судний день;neg
5;Христос;pos
6;аборт;neg
7;абразивний;neg
8;абсурд;neg
9;абсурдний;neg
10;абсурдність;neg
11;авангардний;pos
12;авантюрний;pos
13;аварійний;neg
14;аварія;neg
15;автентичний;pos
16;автомат;neg
17;автоматний;neg
18;автономний;pos
19;авторитарний;neg
20;авторитетний;pos
21;авіаудар;neg
22;ага;pos
23;агонізувати;neg
24;агонія;neg
25;агонії;neg

26;агресивний;neg

27;агресивність;neg

28;агресор;neg

29;агресія;neg

30;агітка;neg

31;адаптивний;pos

32;адаптований;pos

33;адвокат;pos

34;адвокати;pos

35;адекватний;pos

36;адекватно;pos

37;ажурний;pos

38;ажіотаж;neg

39;акламація;pos

40;акомодаційний;pos

41;активний;pos

42;акторський;pos

43;акула;neg

44;акуратний;pos

45;акуратно;pos

46;алергічний;neg

47;алергія;neg

48;алкаш;neg

49;алкоголізм;neg

50;алогічний;neg

51;альтруїстичний;pos

52;альтруїстично;pos

53;амбівалентний;neg

54;амбівалентність;neg

55;амбітний;pos

56;амбітно;pos

57;амбіційний;pos

58;амбіція;pos

59;американські гірки;neg

60;амністія;pos

61;аморальний;neg

62;аморально;neg

63;аморальність;neg

64;аморфний;neg

65;ампутація;neg

66;анархізм;neg

67;анархіст;neg

68;анархічний;neg

69;анархія;neg

70;ангел;pos

71;ангельський;pos

72;анемічний;neg

73;аномальний;neg

74;аномалія;neg

75;антагонізм;neg

76;антагонізувати;neg

77;антагоніст;neg

78;антагоністичний;neg

79;анти;neg

80;анти–;neg

81;анти–білий;neg

82;антиамериканський;neg

83;антигромадський;neg

84;антигромадські;neg

85;антигуманний;neg

86;антидержавний;neg

sentiment_ru.csv

word;pos_neg
аборт;neg
абракадабра;neg
абсурдный;neg
авантюрист;neg
авантюристский;neg
аварийность;neg
авиакатастрофа;neg
авиаудар;neg
авитаминозный;neg
автовор;neg
автомобильная пробка;neg
авторитарность;neg
авторитет снижается;neg
авторитет;pos
авторитетом не пользоваться;neg
автоугонщик;neg
агонизировать;neg
агрессивный;neg
агрессорский;neg
адаптация;pos
адаптироваться;pos
административные меры;neg
адовый;neg
ажиотажный;neg
аккуратистка;pos
актуальность;pos

алкашка;neg

алкоголик;neg

алкогольная зависимость;neg

алконавт;neg

аллергический;neg

алогический;neg

алчность;neg

аморальность;neg

аморфность;neg

аморфный;neg

ампутация;neg

анархизм;neg

анархистский;neg

ангажированность;neg

ангельский;pos

ангиозный;neg

анемичный;neg

аномальность;neg

анорексия;neg

антагонистический;neg

антагонистский;neg

антигуманный;neg

антиконституционный;neg

антиобщественный;neg

антипатия;neg

антисанитарный;neg

антисемитка;neg

антисоциальный;neg

античеловечный;neg

анус;neg

анфолловить;neg

апатический;neg

апатия;neg

аппетитность;pos

арест;neg

аристократ;pos

аристократичность;pos

аритмия;neg

ароматичность;pos

ароматный;pos

артеросклероз;neg

артистизм;pos

артистичный;pos

архаичность;neg

астма;neg

асфиксия;neg

атеросклероз;neg

аутсайдер;neg

аферист;neg

афигенно;pos

афтершок;neg

ахинея;neg

бабник;neg

базарить;neg

баиньки;pos

балаболка;neg

балда;neg

баловать;neg

банальность;neg

банда;neg

бандит;neg

бандитский;neg

бандподполье;neg

бандюган;neg

банкрот;neg

бардачный;neg

бархатистый;pos

бархатный;pos

басмаческий;neg

бахвал;neg

бацилловый;neg

башковитый;pos