Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра математики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: «**МАШИННЕ НАВЧАННЯ ДЛЯ ІДЕНТИФІКАЦІЇ ВПЛИВОВИХ ОСІБ І ВИМІРЮВАННЯ ЇХНЬОГО ВПЛИВУ НА ГРОМАДСЬКУ ДУМКУ**»

Виконала: студентка 4-го року навчання,
Спеціальності
113 Прикладна математика
Шютева Вікторія Мігалівна

Керівник Афонін А.О.
кандидат фізико-математичних наук,
доцент
Рецензент _____
Кваліфікаційна робота захищена
з оцінкою _____

Київ – 2024

План виконання роботи:

	Назва етапу	Дата	Примітки
1.	Отримання теми кваліфікаційної роботи та ознайомлення з нею.	Жовтень	
2.	Розробка плану та структури роботи.	Листопад	
3.	Ознайомлення з науковою літературою.	Грудень	
4.	Робота над теоретичною частиною.	Лютий	
5.	Робота над практичною частиною.	Березень-квітень	
6.	Робота над текстовим оформленням результатів та створення презентації	Початок травня	
7.	Попередній аналіз, передзахист та виправлення помилок.	Травень	
8.	Захист кваліфікаційної роботи.	Червень	

ЗМІСТ

ВСТУП.....4
РОЗДІЛ 1. ТЕОРЕТИЧНА ІНФОРМАЦІЯ..... 6
1.1 Методи машинного навчання..... 6
1.2. Машинне навчання з підкріпленням..... 8
1.3. Загальні алгоритми машинного навчання..... 9
1.4. Процес до створення машинного навчання..... 15
РОЗДІЛ 2. ПРАКТИЧНА ЧАСТИНА.....17
2.1. Мета, методи дослідження та збір даних..... 17
2.2. Реалізація методів та результати..... 17
2.2.1. Аналіз датасетів.....17
2.2.2. Коефіцієнт Пірсона та графіки кореляцій.....19
2.2.3. Очищення датасетів від неактуального шуму.....21
2.2.4. Найвживаніші слова — NLP Wordclouds..... 22
2.2.5. Коефіцієнт впливовості (influence score)..... 27
2.2.6. Аналіз сентименту.....29
2.3. Проблеми, які виникали.....30
2.4. Подальша робота..... 31
ВИСНОВКИ.....32
ДЖЕРЕЛА.....33
ДОДАТКИ.....34

ВСТУП

Машинне навчання сьогодні є однією з найбільш актуальних і перспективних галузей технологій. Застосування алгоритмів машинного навчання дозволяє аналізувати великі обсяги даних, автоматизувати процеси, прогнозувати, створювати системи, які самостійно навчаються та адаптуються до нових умов. Це відкриває багато нових можливостей у різних сферах, а особливо актуальним буде і для ідентифікації впливових осіб і вимірювання їхнього впливу на громадську думку.

Поява соціальних мереж змінила спосіб поширення інформації, формування думок і прийняття рішень. У цю еру інформаційного перевантаження роль впливових осіб виходить за межі простої популярності, поширюючись на вплив на людські рішення, думки, суспільні настрої та погляди, формування політичного дискурсу. У світі, який все більше формується під впливом цифровізації, здатність ідентифікувати та розуміти впливових осіб має безпрецедентне значення.

У цій дипломній роботі ми маємо справу з темою «Машинне навчання для ідентифікації впливових осіб і вимірювання їхнього впливу на громадську думку». Оскільки зараз ми живемо в епоху, коли люди мають владу впливати на громадську думку одним твітом чи дописом, розуміння динаміки впливу стає ключовим.

Використання можливостей машинного навчання і аналізу даних для ідентифікації впливових осіб відкриває двері для глибшого розуміння того, як люди можуть формувати наративи.

Тому у наступних розділах ми розглянемо алгоритми машинного навчання та розробимо власні функції з використанням МН для дослідження публічних осіб та їхньої впливовості на громадську думку у величезному просторі нескінченної інформації, NLP-аналіз дописів, а також проаналізуємо датасети. У цій дипломній роботі ми глибше зрозуміємо машинне навчання та соціальні мережі, які формують наші думки, рішення та колективну свідомість.

РОЗДІЛ 1. ТЕОРЕТИЧНА ІНФОРМАЦІЯ

Машинне навчання (МН) – це галузь штучного інтелекту (ШІ) та комп'ютерних наук, яка зосереджується на використанні даних і алгоритмів для імітації способу навчання людей, поступово покращуючи точність. [1]

1.1 Методи машинного навчання

Моделі машинного навчання діляться на три основні категорії.

1) Контрольоване машинне навчання (Supervised machine learning)

Контрольоване навчання, також відоме як кероване машинне навчання, визначається використанням позначених (labeled data) наборів даних для навчання алгоритмів для класифікації даних або точного прогнозування результатів. Коли вхідні дані надходять у модель, модель коригує свої значення, поки їх не буде встановлено належним чином. Це відбувається як частина процесу перехресної перевірки, щоб переконатися, що модель не перенавчена або не недонавчена. Перенавчання або недонавчання виникає, коли даних для аналізу є забагато або замало відповідно, і через це машина може надати неадекватні результати [2]. Контрольоване навчання допомагає організаціям вирішувати різноманітні реальні проблеми в масштабі, наприклад, класифікувати спам в окремій папці з папки "Вхідні". Деякі методи, які використовуються в керованому навчанні, включають нейронні мережі, наївну баєсівську регресію, лінійну регресію, логістичну регресію, випадковий ліс і метод опорних векторів (SVM).

Як працюють алгоритми контрольованого МН:

Нехай дано набір із N кількості тренованих прикладів типу $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, де x_1 — вектор i-го прикладу та y_i — це його позначення; алгоритм шукає функцію $g: X \to Y$, де X — вхідне поле, а Y — вихідне поле. Функція g — це один елемент із певного поля функцій G. Інколи дану функцію ще позначають через так звану рахункову функцію $f: X \times Y \to \mathbb{R}$. У даному випадку значення y повинно повертати максимальний результат $g(x) = arg\,max\, f(x, y)$. Нехай F позначить поле рахункових функцій.

Хоча G та F можуть бути будь-якими полями функцій, багато алгоритмів є ймовірнісними моделями, де g набирає форми умовної ймовірності $g(x) = arg\,max\, P(y|x)$.

Для того, щоб оцінити, наскільки добре функція підходить для даних, що тренують алгоритм, використовується функція втрат:

$$L: Y \times Y \to \mathbb{R}^{\geq 0}$$

Для прикладу (x_i, y_i) функція втрат для передбачення значення $\hat{y}$ є $L(y_i, \hat{y})$.

Ризик $R(g)$ функції g визначено як очікувана втрата функції g. Вираз для оцінки ризику є наступним:

$$R(g) = \frac{1}{N}\sum_i L(y_i, g(x_i)).$$ [3]

2) Неконтрольоване машинне навчання (unsupervised machine learning)

Неконтрольоване навчання, також відоме як некероване машинне навчання, використовує алгоритми машинного навчання для аналізу та кластеризації непозначених наборів даних (non-labeled data). Ці алгоритми виявляють приховані паттерни або групи даних без втручання людини. Здатність цього методу виявляти подібності та відмінності в інформації робить його

ідеальним для дослідницького аналізу даних, стратегій перехресних продажів, сегментації клієнтів, а також для розпізнавання зображень і шаблонів. Він також використовується для зменшення кількості функцій у моделі через процес зменшення розмірності. Метод головних компонентів (PCA) і сингулярний розклад матриці (SVD) є двома поширеними підходами для цього. Інші алгоритми, що використовуються в неконтрольованому навчанні, включають нейронні мережі, метод k-середніх і ймовірнісні методи кластеризації. [1]

3) Напівконтрольоване навчання (Semi-supervised learning)

Напівконтрольоване навчання є посередником між контрольованим і неконтрольованим навчанням. Під час навчання він використовує менший набір позначених даних, щоб керувати класифікацією та виділяти ознаки із більшого набору даних без позначень. Напівконтрольоване навчання може вирішити проблему відсутності достатньої кількості позначених даних для алгоритму контрольованого навчання. Це також допоможе, якщо позначати достатню кількість даних занадто дорого.

1.2. Машинне навчання з підкріпленням

Машинне навчання з підкріпленням (reinforcement learning) — це модель машинного навчання, схожа на контрольоване навчання, але алгоритм не навчається за допомогою вибіркових даних. Ця модель навчається по ходу за допомогою проб і помилок. Послідовність успішних результатів буде посилена, щоб розробити найкращу рекомендацію чи політику щодо певної проблеми.

Найважливіша відмінність навчання з підкріпленням від інших видів полягає в тому, що відсутнє представлення пар введення/виведення. Натомість після вибору дії агент отримує негайну винагороду і подальший

стан, але не повідомляється, яка дія була б у його найкращих довгострокових інтересах. Щоб діяти оптимально, агенту необхідно активно накопичувати корисний досвід щодо можливих станів системи, дій, переходів і винагород. Ще одна відмінність від навчання під наглядом полягає в тому, що виконання роботи у реальному часі є важливою: оцінка системи часто відбувається одночасно з навчанням.

Деякі аспекти навчання з підкріпленням тісно пов’язані з проблемами пошуку та планування в штучному інтелекті. Алгоритми пошуку ШІ генерують задовільну траєкторію через граф станів. Планування працює подібним чином, але зазвичай у складнішій конструкції, ніж граф, у якому стани представлені композиціями логічних виразів замість атомарних символів. Ці алгоритми штучного інтелекту є менш загальними, ніж методи навчання з підкріпленням, оскільки вони вимагають попередньо визначеної моделі переходів станів. З іншого боку, навчання з підкріпленням, принаймні в тих окремих випадках, для яких була розроблена теорія, припускає, що весь простір станів може бути перерахований і збережений у пам’яті, припущення, до якого не прив’язані звичайні алгоритми пошуку.

1.3. Загальні алгоритми машинного навчання

1) Лінійна регресія

Лінійна регресія – це контрольований метод машинного навчання, яка використовується для передбачення та прогнозування значень, які потрапляють у безперервний діапазон, наприклад кількості продажів або цін на житло. Це метод, отриманий зі статистики, який зазвичай використовується для встановлення зв’язку між вхідною змінною (X) і вихідною змінною (Y), яку можна представити прямою лінією.

Простими словами, лінійна регресія бере набір точок даних із відомими вхідними та вихідними значеннями та знаходить лінію, яка найкраще відповідає цим точкам. Ця лінія, відома як «лінія регресії», служить прогностичною моделлю. Використовуючи цей рядок, ми можемо оцінити або передбачити вихідне значення Y для даного вхідного значення X.

Лінійна регресія в основному використовується для прогнозного моделювання, а не для категоризації. Це корисно, коли ми хочемо зрозуміти, як зміни у вхідній змінній впливають на вихідну змінну. Аналізуючи нахил і відрізок лінії регресії, ми можемо отримати уявлення про зв'язок між змінними та робити прогнози на основі цього розуміння. [4]

2) Логістична регресія

Логістична регресія – це контрольований метод машинного навчання, який в основному використовується для завдань бінарної класифікації. Він зазвичай використовується, коли ми хочемо визначити, чи належать вхідні дані до того чи іншого класу, наприклад, вирішити, чи є зображення собакою чи ні.

Логістична регресія передбачає ймовірність того, що вхідні дані можна класифікувати в один первинний клас. Однак на практиці його зазвичай використовують для групування вихідних даних у дві категорії: основний клас, а не основний клас. Щоб досягти цього, логістична регресія створює порогове значення або межу для двійкової класифікації. Наприклад, будь-яке вихідне значення від 0.00 до 0.49 може бути класифіковано як одна група, а значення від 0.50 до 1.00 – як інша група.

Отже, логістична регресія зазвичай використовується для бінарної категоризації, а не для прогнозного моделювання. Це дозволяє нам призначити вхідні дані одному з двох класів на основі оцінки ймовірності та визначеного порогу. Це робить логістичну регресію потужним інструментом

для таких завдань, як розпізнавання зображень, виявлення спаму електронної пошти або медична діагностика, коли нам потрібно класифікувати дані за різними класами. [4]

Основою логістичної регресії є сигмоїдна функція:

$$y = \frac{1}{1+e^{-z}}, \text{ де } y \in (0, 1)$$

Сигмоїдну функцію також називають логістичною.

3) Наївний Байєсівський алгоритм

Naive Bayes — це контрольований метод машинного навчання, який використовується для створення прогнозованих моделей для бінарних або мультикласифікаційних завдань. Він базується на теоремі Байєса та оперує умовними ймовірностями, які оцінюють ймовірність класифікації на основі комбінованих факторів, припускаючи незалежність між ними.

Теорема Байєса:

$$P(A|B) = \frac{P(A|B)\cdot P(A)}{P(B)}$$

$A, B - events$

$P(A|B) - probability\ of\ A\ given\ B\ is\ true$

$P(A|B) - probability\ of\ B\ given\ A\ is\ true$

$P(A), P(B) - the\ independent\ probabilities\ of\ A\ and\ B$

Можна розглянути програму, яка ідентифікує тварин за допомогою алгоритму Наївного Байєса. Алгоритм враховує конкретні фактори, такі як колір та розміри тілобудови, що приймаються, для класифікації зображень тварин. Хоча кожен із цих факторів розглядається незалежно, алгоритм об’єднує їх, щоб оцінити ймовірність того, що об’єкт є певною твариною.

Наївний Байєс використовує припущення про незалежність факторів, що спрощує обчислення та дозволяє алгоритму ефективно працювати з великими наборами даних. Він особливо добре підходить для таких завдань, як класифікація документів, фільтрація спаму в електронній пошті, аналіз настроїв і багато інших програм, де фактори можна розглядати окремо, але все одно вносять свій внесок у загальну класифікацію. [4]

4) Дерево рішень

Дерево рішень — це контрольований алгоритм навчання, який використовується для завдань класифікації та прогнозного моделювання. Він нагадує блок-схему, починаючи з кореневого вузла, який задає конкретне запитання щодо даних. На основі відповіді дані спрямовуються різними гілками до наступних внутрішніх вузлів, які задають додаткові запитання та спрямовують дані до наступних гілок. Цей процес триває, доки дані не досягнуть кінцевого вузла, також відомого як листовий вузол, де більше не відбувається розгалуження.

Алгоритми дерева рішень популярні в машинному навчанні, оскільки вони можуть обробляти складні набори даних з легкістю та простотою. Структура алгоритму дозволяє легко зрозуміти та інтерпретувати процес прийняття рішень. Ставлячи послідовність запитань і дотримуючись відповідних гілок, дерева рішень дозволяють класифікувати або прогнозувати результати на основі характеристик даних.

Ця простота та можливість інтерпретації роблять дерева рішень цінними для різних застосувань у машинному навчанні, особливо при роботі зі складними наборами даних.

5) Випадковий ліс

Алгоритм випадкового лісу — це сукупність дерев рішень, які використовуються для класифікації та прогнозного моделювання. Замість

того, щоб покладатися на одне дерево рішень, випадковий ліс об’єднує прогнози з кількох дерев рішень, щоб зробити точніші прогнози.

У випадковому лісі численні алгоритми дерева рішень (іноді сотні або навіть тисячі) індивідуально навчаються з використанням різних випадкових вибірок із навчального набору даних. Такий метод відбору проб називається «пакетуванням». Кожне дерево рішень навчається незалежно на його відповідній випадковій вибірці.

Після навчання випадковий ліс приймає ті самі дані та передає їх у кожне дерево рішень. Кожне дерево створює прогноз, а випадковий ліс підраховує результати. Найпоширеніший прогноз серед усіх дерев рішень потім вибирається як остаточний прогноз для набору даних.

Випадкові ліси вирішують поширену проблему під назвою «переобладнання», яка може виникнути з окремими деревами рішень. Переобладнання відбувається, коли дерево рішень стає занадто тісно узгодженим із навчальними даними, що робить його менш точним при поданні нових даних.

6) K-найближчий сусід

K-найближчий сусід (KNN) — це контрольований алгоритм навчання, який зазвичай використовується для завдань класифікації та прогнозного моделювання. Назва «K-найближчий сусід» відображає підхід алгоритму до класифікації результату на основі його близькості до інших точок даних на графіку.

Скажімо, у нас є набір даних із позначеними точками, одні з яких позначені зеленим, а інші — жовтим. Коли ми хочемо класифікувати нову точку даних, KNN дивиться на її найближчих сусідів на графіку. Буква «K» у KNN означає кількість найближчих сусідів. Наприклад, якщо K встановлено на 10, алгоритм розглядає 10 точок, найближчих до нової точки даних.

Базуючись на більшості міток серед K-найближчих сусідів, алгоритм призначає класифікацію новій точці даних. Наприклад, якщо більшість найближчих сусідів є зеленими точками, алгоритм класифікує нову точку як належну до зеленої групи. [4]

Крім того, KNN також можна використовувати для завдань прогнозування. Замість призначення мітки класу KNN може оцінити значення невідомої точки даних на основі середнього або медіани K-найближчих сусідів.

7) Метод k-середніх

K-means — це неконтрольований алгоритм, який зазвичай використовується для завдань кластеризації та розпізнавання образів. Він спрямований на групування точок даних на основі їх близькості одна до одної. Подібно до K-найближчого сусіда (KNN), метод k-середніх використовує концепцію близькості для виявлення патернів у даних.

Кожен з кластерів визначається центроїдом, реальною або уявною центральною точкою для кластера. Метод корисний для великих наборів даних. Може надати розуміння внутрішньої структури даних шляхом групування подібних точок. Він може застосовуватися в різних сферах, таких як сегментація клієнтів, стиснення зображень і виявлення аномалій.

8) Нейронна мережа

Може бути як і контрольованим, так і неконтрольованим алгоритмом навчання. Нейронні мережі імітують роботу людського мозку за допомогою величезної кількості зв’язаних вузлів обробки. Нейронні мережі добре розпізнають патерни та відіграють важливу роль у програмах, включаючи переклад природної мови, розпізнавання зображень, розпізнавання мовлення та створення зображень.

9) Апріорі

Апріорі — це алгоритм неконтрольованого навчання, який використовується для прогнозного моделювання, зокрема у сфері аналізу правил асоціації.

Алгоритм Apriori спочатку був запропонований у 1994 році як алгоритм для аналізу даних та у машинному навчанні щодо асоціативних правил. Він зазвичай використовується в задачах розпізнавання образів і прогнозування, наприклад, для розуміння ймовірності придбання споживачем одного продукту після покупки іншого.

Алгоритм Апріорі працює шляхом вивчення транзакційних даних, що зберігаються в реляційній базі даних. Він визначає часті набори елементів, які є комбінаціями елементів, які часто зустрічаються разом у транзакціях. Потім ці набори елементів використовуються для створення правил асоціації. Наприклад, якщо клієнти часто купують певний продукт A і певний продукт B разом, можна створити правило асоціації, яке припускає, що покупка A збільшує ймовірність покупки B.

Застосовуючи алгоритм Апріорі, аналітики можуть виявляти цінну інформацію з транзакційних даних, дозволяючи їм робити прогнози або рекомендації на основі спостережуваних моделей асоціацій набору елементів.

1.4. Процес до створення машинного навчання

Для того, щоб створити модель машинного навчання з нуля, ми повинні пройти певні етапи процесу побудови моделі [7]. Процес включає в собі такі етапи:

1) Визначення проблеми та мети
2) Збір даних

Дані можуть бути зібрані з різних джерел, таких як файли, база даних, Інтернет або мобільні пристрої. Їх можна викачувати самостійно чи

використовувати вже готові датасети. Кількість і якість зібраних даних визначає ефективність результату. Чим більше даних, тим точніший прогноз.

3) Підготовка та обробка даних

Щоб отримувати якісний результат, дані необхідно очистити та перетворити у належний формат. Потрібно прибирати шум, дубльовані дані, використовувати методи фільтрації та заповнювати відсутні значення.

4) Аналіз даних

Використовуючи бібліотеки Python, до прикладу Pandas, Plotly, Numpy, ми можемо починати аналіз даних та знаходити патерни, які будемо використовувати для тренування моделі.

5) Тренування моделі

Це зазвичай ітеративний процес обрання потрібної моделі, а потім побудова моделі на навчальних датасетах. Мета тренінгу: якомога частіше правильно відповідати на питання або робити прогноз.

6) Тестування моделі

Коли модель вже навчена на датасеті, ми тестуємо модель. Надаючи їй тестовий набір даних, ми перевіряємо точність нашої моделі. Тестування моделі визначає відсоток точності моделі.

7) Деплоймент моделі (розгортання)

Останнім кроком життєвого циклу машинного навчання є деплоймент, коли ми розгортаємо модель у системі реального світу. У нашому випадку етапом розгортання буде створення остаточного звіту про проект.

РОЗДІЛ 2. ПРАКТИЧНА ЧАСТИНА

2.1. Мета, методи дослідження та збір даних

Мета роботи: побудувати алгоритми, які використовують машинне навчання й нейронну мережу, для соціальних досліджень, а саме для аналізу впливовості публічних осіб у соціальній мережі Twitter (X); зробити NLP-аналіз дописів та зробити висновок, яким чином вони можуть впливати на громадську думку; а також зробити аналіз датасетів.

Дані зібрані за 2020 рік із відкритого доступу з мережі Інтернет. Датасети містять таку необхідну нам інформацію: тексти дописів, áвтора, кількість вподобань, кількість поширень допису, кількість коментарів.

У дипломній роботі ми використовували такі засоби й інструменти для роботи: мову програмування Python, середовище Jupyter Notebook, бібліотеки Pandas, Numpy, Plotly Express (візуалізації), re, NLTK, Matplotlib, Wordcloud (візуалізації), Math, Scikit-Learn (машинне навчання), TensorFlow (машинне навчання).

2.2. Реалізація методів та результати

У цьому підрозділі описана вся виконана робота та результати (2.2.1-2.2.6).

2.2.1. Аналіз датасетів.

У нас є 5 датасетів публічних осіб із соціальної мережі Twitter. Перший датасет стосується кандидата у президенти США 2020 року Джо Байдена. За допомогою бібліотеки Pandas визначаємо, що він налічує 200 рядків (твітів)

даних. Другий датасет стосується заступниці кандидата Джо Байдена — Камали Гарріс. Він також налічує 200 (твітів) рядків даних.

Третій датасет стосується другого кандидата у президенти США 2020 року Дональда Трампа, він налічує 200 рядків (твітів) даних. Четвертий датасет стосується його заступника — Майка Пенса. Він налічує 199 (твітів) рядків даних.

За допомогою мови програмування Python, робочого середовища Jupyter Notebook, бібліотек Pandas та Plotly Express вдалось визначити та зобразити загальну кількість вподобань та поширень дописів кожного з них. На рисунку 1 зображена дана інформація.

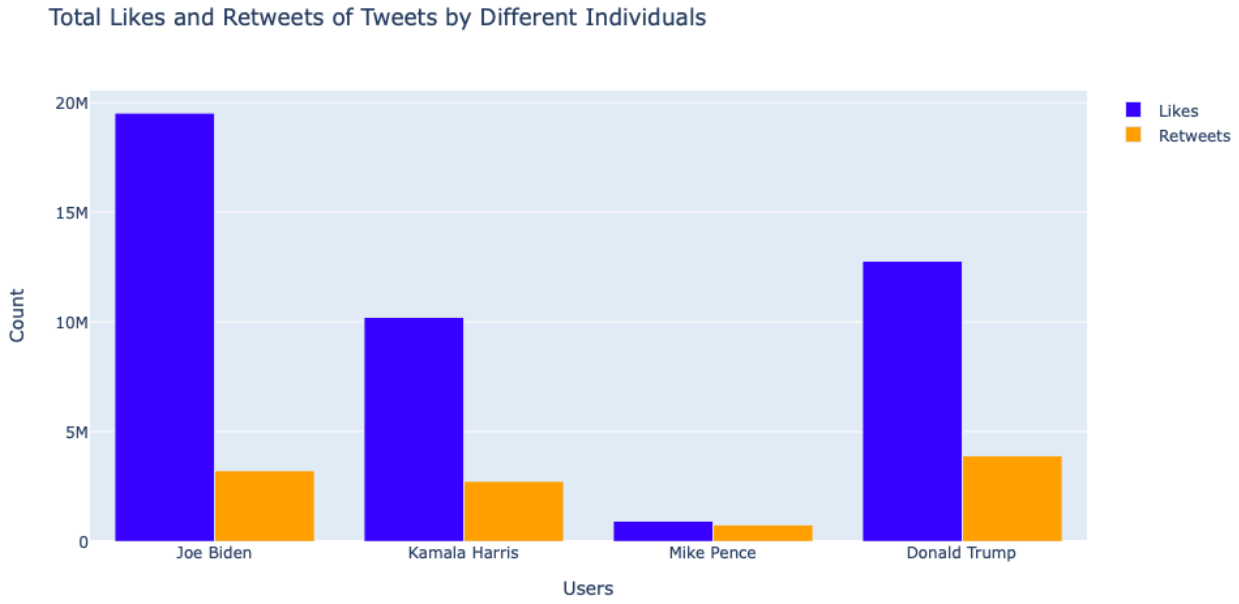

Рис. 1. Загальна кількість вподобань та поширень у кожного з осіб

З графіку ми можемо зробити висновок, що Джо Байдена можна назвати найбільш впливовим у цьому аспекті, оскільки він має найбільшу кількість вподобань, що сягає майже двадцяти мільйонів. На другому місці — Дональд Трамп, у якого майже тринадцять мільйонів вподобань. На третьому місці опинилася заступниця Джо Байдена – Камала Гарріс – з кількістю в десять мільйонів, а найменш впливовим у цьому плані можна вважати заступника

Трампа — Майка Пенса, у якого кількість вподобань не досягає навіть до одного мільйона.

П’ятий датасет стосується політичного журналіста Рогана О’Гендлі. Він налічує 300 рядків (твітів) даних. Містить наступну необхідну нам інформацію: тексти, кількість вподобань, кількість коментарів.

За допомогою бібліотеки Pandas та Plotly Express вдалось визначити та зобразити загальну кількість вподобань та коментарів під дописами журналіста Рогана О’Гендлі із нашої вибірки. На рисунку 2 зображена дана інформація.

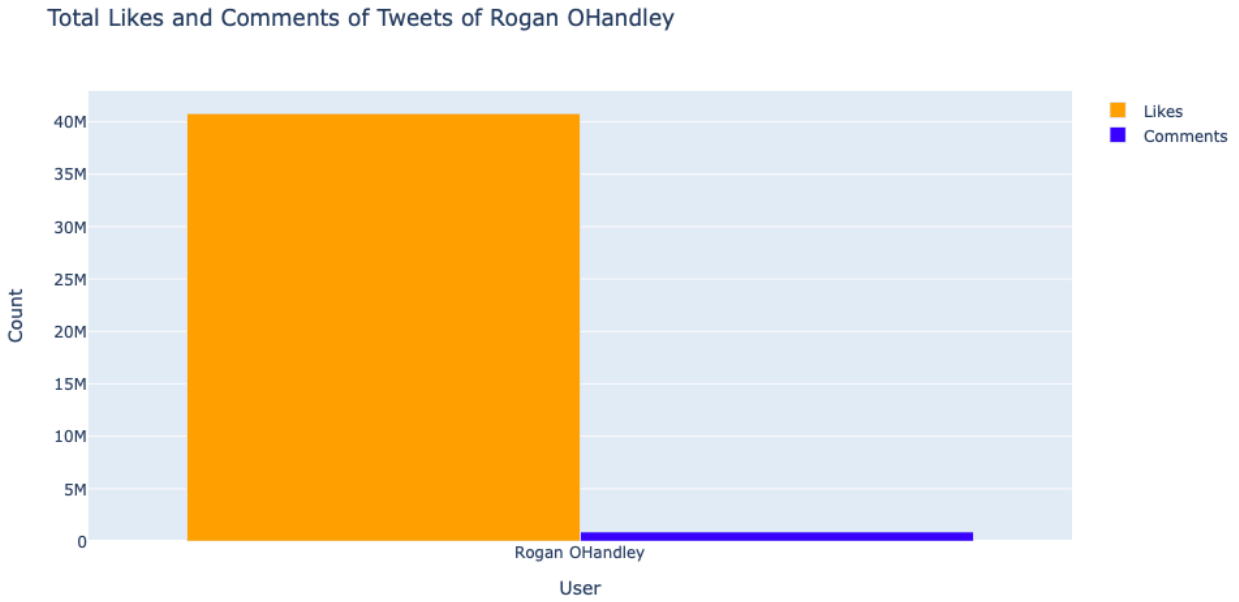

Рис. 2. Загальна кількість вподобань та коментарів у Рогана О’Гендлі

Код до даного пункту (2.2.1) представлений в додатку А.

2.2.2. Коефіцієнт Пірсона та графіки кореляцій

Коефіцієнт лінійної кореляції Пірсона — це статистичний зв’язок між двома наборами змінних. Визначаємо за наступною формулою:

$$r = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma(x_i - \bar{x})^2 (y_i - \bar{y})^2}}, \text{ де}$$

r - коефіцієнт кореляції,

x_i - значення змінних x у вибірці,

$\bar{x}$ - середнє значення змінної x,

y_i - значення змінних y у вибірці,

$\bar{y}$ - середнє значення змінної y.

Ми визначили коефіцієнти лінійної кореляції між кількістю лайків та поширень дописів кандидатів у президенти США 2020 року Байдена і Трампа та зобразили графіки кореляцій. За результатами Python-обчислень коефіцієнт кореляції Пірсона у випадку першого дорівнює 0.95, а у випадку другого відповідно – 0.85. У обох він вказує на існування сильного позитивного зв'язку. На графіках (побудованих за допомогою matplotlib) ми також можемо побачити, що зв'язок є сильним, точки здебільшого розташовані близько одна біля одної і йдуть вгору (рисунок 3).

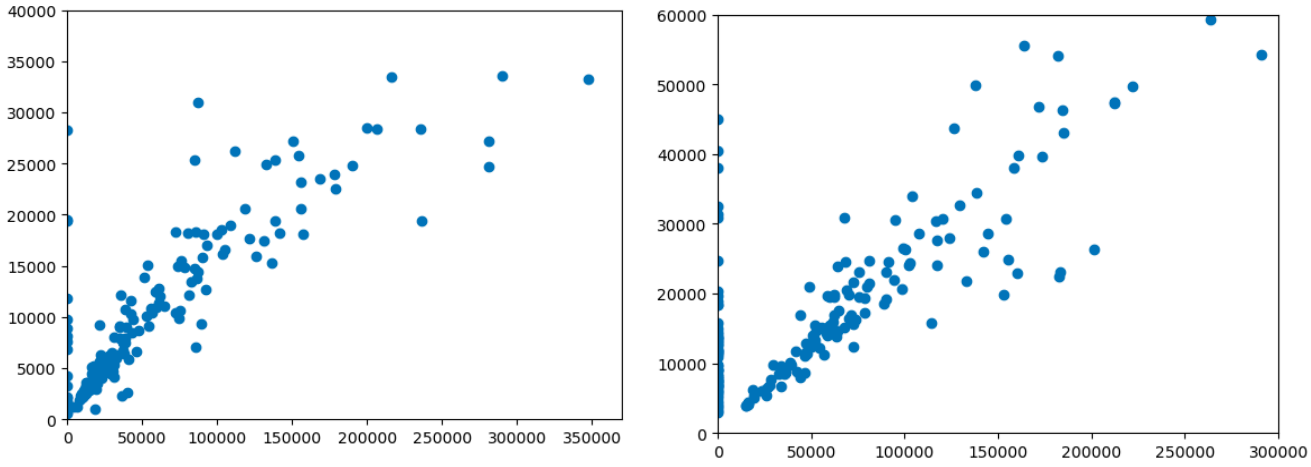

Рис. 3. Графіки кореляцій між кількістю вподобань і поширень дописів Байдена (1) та Трампа (2) відповідно

Також ми визначили коефіцієнт кореляції між кількістю лайків та коментарів у американського журналіста Рогана О'Гендлі та зобразили графік. За результатами обчислень коефіцієнт кореляції Пірсона в даному випадку дорівнює 0.32. Він вказує на існування деякого зв'язку, проте цей зв'язок є помірно слабким. На графіку ми також можемо побачити, що зв'язок є слабким (рисунок 4).

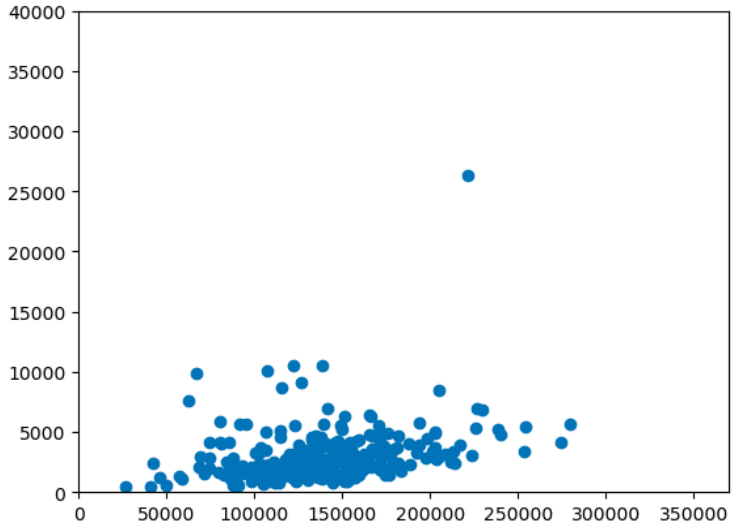

Рис. 4. Графік кореляції між кількістю вподобань та коментарів.

Код до даного пункту (2.2.2) представлений в Додатку Б.

2.2.3. Очищення датасетів від неактуального шуму.

Шум в даних — це неактуальна інформація, яка спотворює точний аналіз. Для подальшої якісної роботи з даними нам необхідна функція, яка прибирає ретвіти (поширення). Оскільки ретвіти є неактуальним шумом та не є цінними одиницями інформації, бо це тільки поширення, а не власноруч написана думка особи, котру ми аналізуємо, вони підлягають очищенню з датасету. Отже, ми отримали очищені датасети без ретвітів і надалі також використовуватимемо їх.

Тепер очищений датасет Байдена містить 183 рядки даних, Гарріс – 153 рядки, Трампа – 132 рядки, Пенса – 94 рядки, а датасет журналіста О'Гендлі не змінився – у нього ретвіти відсутні, залишається так само 300 рядків.
Код до даного пункту представлений в Додатку В.

2.2.4. Найвживаніші слова — NLP Wordclouds

NLP-аналіз — це галузь штучного інтелекту, яка обробляє та аналізує природню мову.

Ми виконали NLP-аналіз текстів твітів кожного датасету та зробили візуалізацію словесної хмари (wordcloud) із 20 найбільш вживаних слів. Для цього використовувались бібліотеки NLTK та Wordcloud. У даній Python-функції ми також визначили додаткові стоп-слова, які з'являлися у статистиці і з якими виникали проблеми, а саме вони були неактуальними та не несли жодного семантичного значення, до прикладу: 'one', 'https', 'let', 'could'. Також в коді виконуємо токенізацію текстів, щоб розбити його на менші одиниці, оскільки це спрощує аналіз та підвищує ефективність алгоритмів.

У ході NLP-аналізу ми порівняли оригінальні початкові датасети із очищеними від ретвітів. Найбільш виразно різницю ми можемо спостерігати у датасеті Трампа, де очищення від ретвітів допомогло зробити хмару слів більш семантично якісною та репрезентативною (з'явилося більше слів політичного та географічного забарвлення, до прикладу, назви штатів – рисунок 6, 7).

Код функції представлений в Додатку Г.

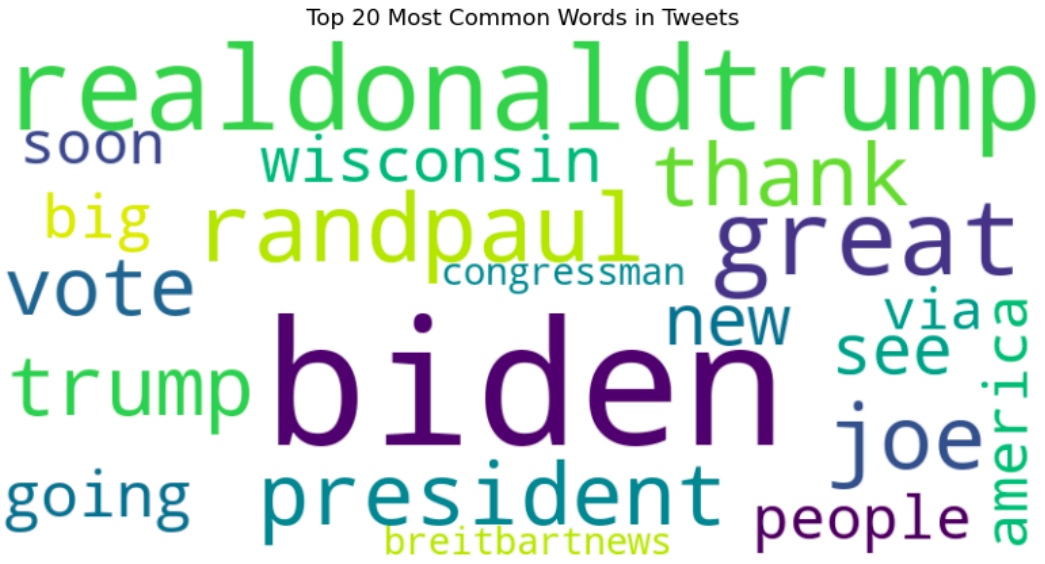

Рис. 5. Хмара найуживаніших слів Трампа до очищення датасету.

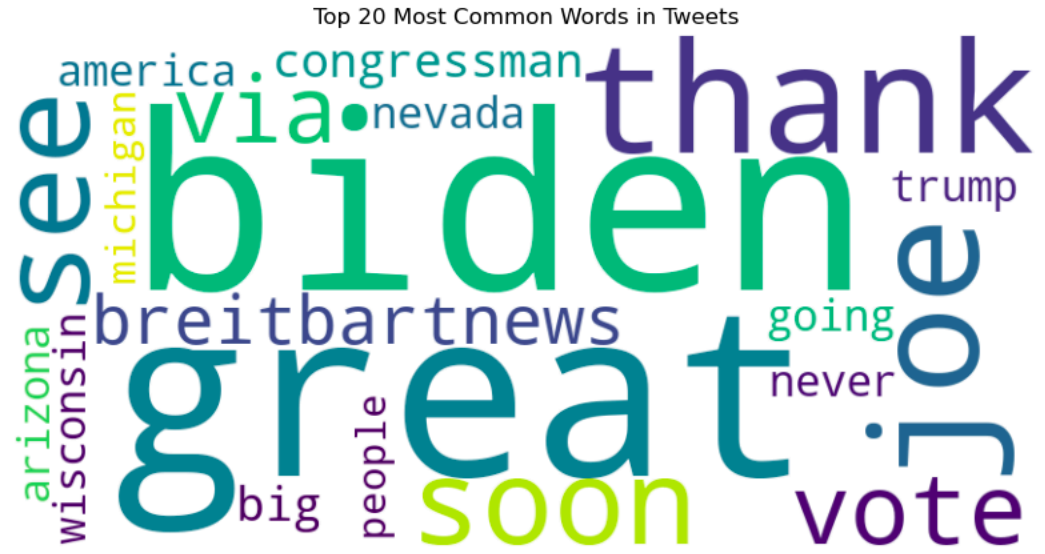

Рис. 6. Хмара найуживаніших слів Трампа після очищення датасету.

На наступних рисунках зображено хмари найуживаніших слів Джо Байдена, Камали Гарріс та Майка Пенса до та після очищення датасетів.

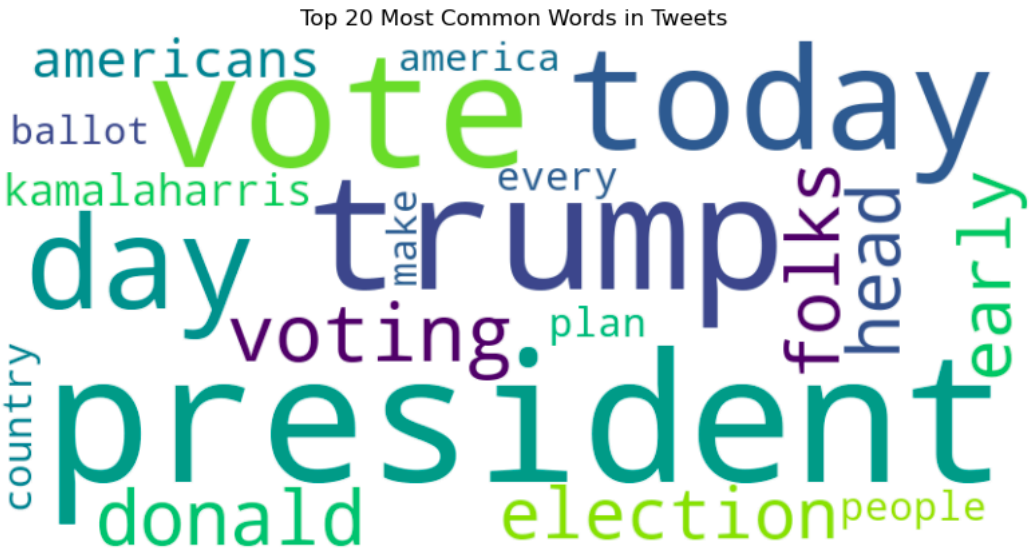

Рис. 7. Хмара найуживаніших слів Байдена до очищення датасету.

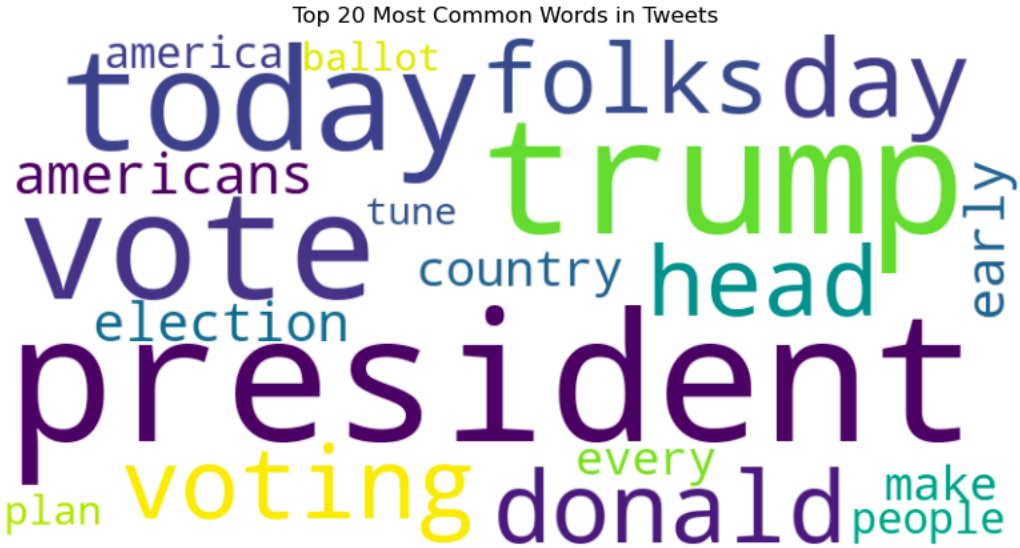

Рис. 8. Хмара найуживаніших слів Байдена після очищення датасету.

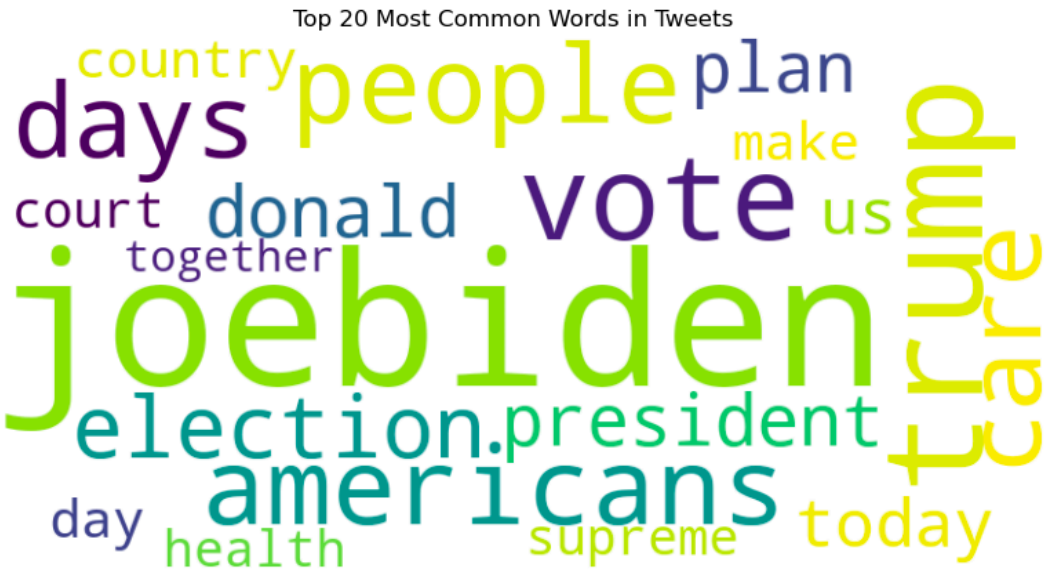

Рис. 9. Хмара найуживаніших слів Гарріс до очищення датасету.

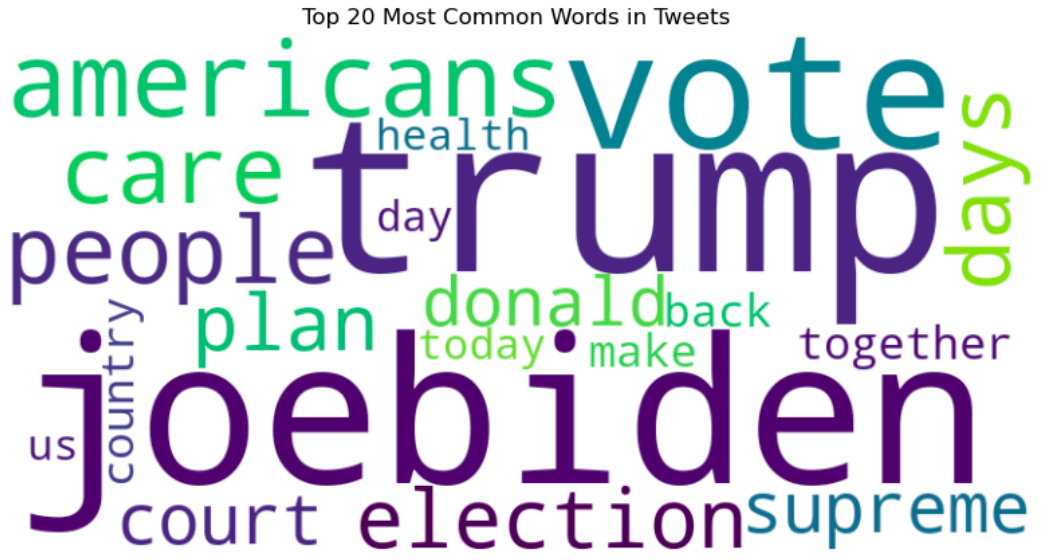

Рис. 10. Хмара найуживаніших слів Гарріс після очищення датасету.

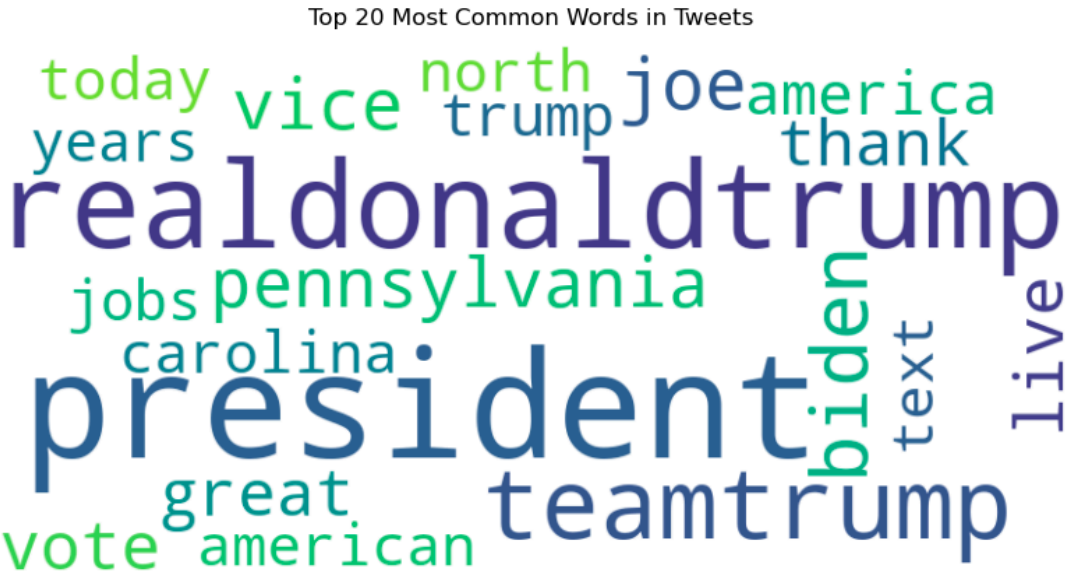

Рис. 11. Хмара найуживаніших слів Майка Пенса до очищення датасету.

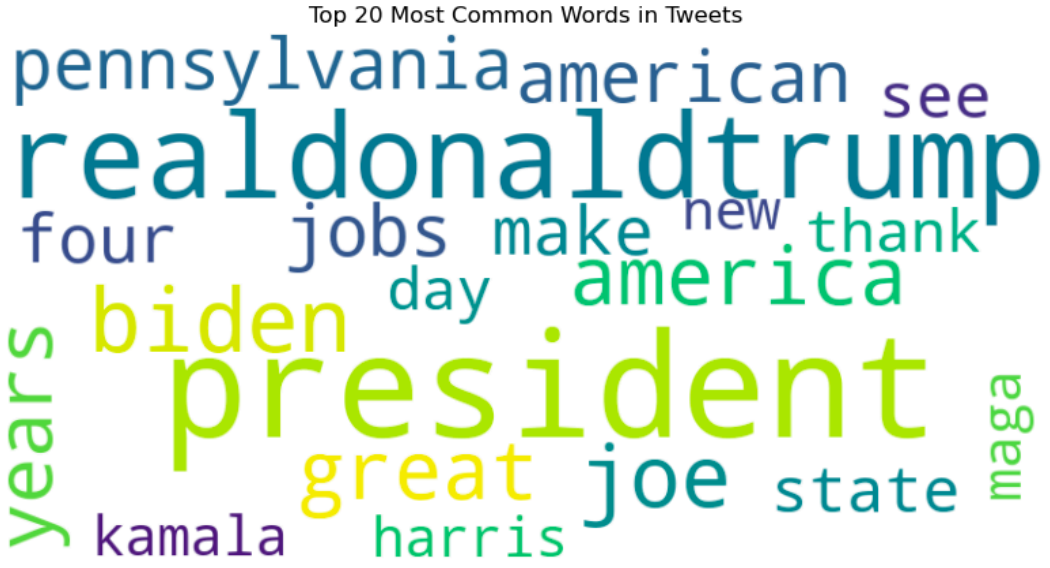

Рис. 12. Хмара найуживаніших слів Майка Пенса після очищення датасету.

Як ми можемо побачити, семантичний сенс до і після очищення у даних трьох осіб не дуже змінився. У кожного з них є політичний зміст слів у текстах. Отже, можна зробити висновок, що на громадськість вони впливають

із закликом до певного політичного впливу, голосування на виборах США, згадуючи назви штатів та закликаючи до дій.

Останній датасет у нас стосується журналіста О'Гендлі і на наступному рисунку ми можемо побачити NLP-аналіз його текстів. Він відрізняється від попередніх, проте жанр так само політично-публіцистичний.

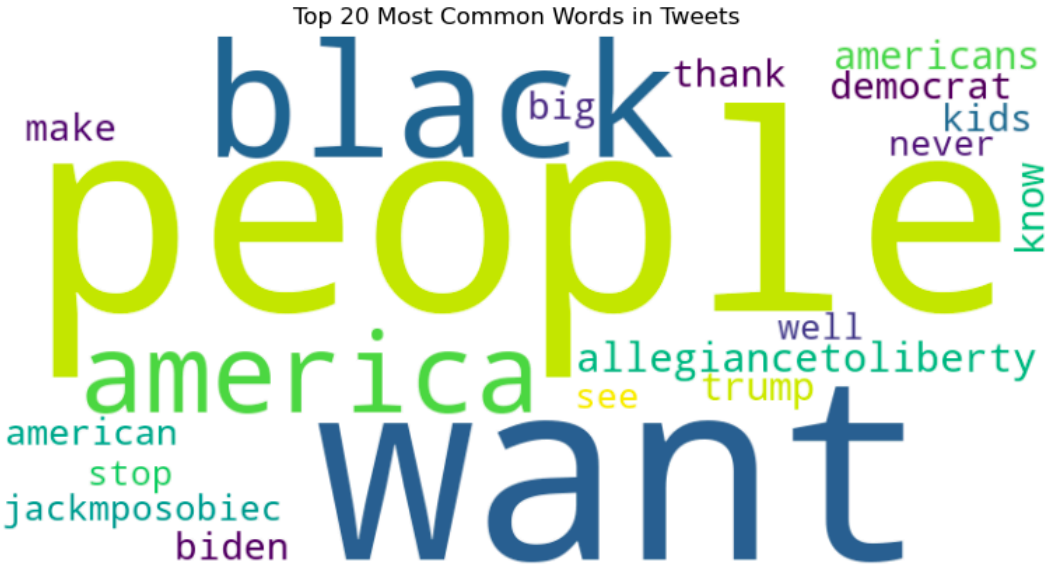

Рис. 13. Хмара найуживаніших слів журналіста Рогана О'Гендлі.

2.2.5. Коефіцієнт впливовості (influence score)

Коефіцієнт впливовості — це значення, якого набуває функція машинного навчання після аналізу датасету особи, базуючись на трьох параметрах: кількості вподобань, поширень та підписників у даної людини.

У цьому підрозділі описую створення та використання функції машинного навчання із застосуванням бібліотек TensorFlow та Scikit-learn. Python-функція допомагає визначити коефіцієнт впливовості особи, покладаючись на формулу:

$$influence = likes \times 0.5 + retweets \times 0.2 + followers \times 0.3$$

Тобто я надала наступну вагу для: кількості вподобань – 50%, кількості поширень – 20% та кількості підписників – 30%. Формула проходить через модель нейронної мережі із трьома шарами. Вхідний шар має три нейрони (input_shape=(3,)), і два прихованих шари мають 64 та 32 нейрони відповідно; обидва використовують функцію активації ReLU для надання їй нелінійності, масштабності й складності. Модель має оптимізатор Адама, який потребує небагато пам’яті і використовує поєднання алгоритму градієнтного спуску із методом середньоквадратичної прогонки. У якості функції втрат використовуємо середньоквадратичну похибку (mean squared error, MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \text{ де}$$

N - кількість вхідних даних,

y_i - спостережувальні значення,

$\hat{y}_i$ - передбачувані значення.

Отже, за результатами алгоритму машинного навчання ми отримали (з першого по четверте місце з коефіцієнтами):

1. Джо Байден – 152.3.
2. Камала Гарріс – 137.9.
3. Дональд Трамп – 129.6.
4. Майк Пенс – 114.9.

Можемо зробити висновок, що нам вдалося розробити алгоритм машинного навчання для визначення коефіцієнту впливовості людей. Як ми бачимо, серед нашої вибірки Джо Байдена знову можна назвати найвпливовішим, як і в одному з попередніх методів дослідження.

Код функції знаходження коефіцієнту впливу представлений в Додатку Д.

2.2.6. Аналіз сентименту

Наступною функцією з машинним навчанням є аналіз сентименту. Аналіз сентименту – це процес визначення емоційної тональності тексту. Для побудови функції ми використовували бібліотеку TensorFlow. У нашому випадку груп, на які можна було розподілити текст, було шість: «sarcasm», «irony», «satire», «understatement», «overstatement» та «rhetorical question». Це у нас був тренувальний датасет з 4805 рядків даних, з якого навчалась чотиришарова модель нейронної мережі. Перший шар – Embedding – перетворює цілочисельні індекси слів у їхні векторні представлення. Максимальна кількість унікальних слів, які можна закодувати (тобто розмір словника) – 10000, а розмірність векторів, які використовуються для представлення кожного слова – 16. Другий шар – GlobalAveragePooling1D – обчислює середнє значення для кожного вектора по всій послідовності. Третій шар – Dense – має 64 нейрони та використовує функцію активації ReLU, яка надає нелінійності моделі, дозволяючи навчатися більш складним залежностям у даних. Останній, четвертий шар – Dense – має 6 нейронів, оскільки у нас 6 класів, на які ми розподіляємо сентимент, як вже згадувалось вище. Шар використовує функцію сигмоїди, яка перетворює вихідні значення у діапазон (0,1), таким чином дозволяючи класифікувати результати на 6 класів.

Сигмоїдна функція:

$$y = \frac{1}{1+e^{-z}}\text{, де } y \in (0,1)$$

При тренуванні моделі за функцію втрат застосовуємо перехресну ентропію:

$$H(p,q) = -\sum_i p_i log_2(q_i)\text{, де}$$

p – істинна ймовірність,

q – передбачена моделлю ймовірність.

Результати класифікації на 6 груп виводяться у вигляді бінарної матриці з нулями та одиницями (рисунок 14). Далі одиниці кожної колонки сумуються і виводяться загальним порахованим результатом.

Код функції аналізу сентименту представлений в додатку Е.

```
Test Accuracy: 0.6742976307868958

Train Predictions:
[[0 0 0 0 0 1]
 [0 0 0 0 0 0]
 [0 0 0 0 0 0]
 ...
 [0 0 0 0 0 0]
 [0 0 0 0 0 0]
 [0 0 0 0 0 0]]

Label Counts:
sarcasm: 4
irony: 3
satire: 0
understatement: 0
overstatement: 1
rhetorical_question: 4
```

Рис. 14. Один із результатів аналізу сентименту.

2.3. Проблеми, які виникали

1) Неможливість самостійно зібрати бажані свіжі дані.

Після зміни власника у компанії Twitter, там було введено нові правила приватності, і тому викачування даних, а саме збір публікацій та суміжної до нього інформації тепер є платним. Тому зібрати дані за 2024 рік коштувало б мінімум 100 доларів і більше, залежно від того, скільки даних користувач хоче викачати. Хоча раніше це було безкоштовно.

2) Недонавчання під час тренування моделі у алгоритмі аналізу сентименту.

Під час аналізу сентименту у мене були різні датасети для тренування моделі, і перший з них спочатку не підходив. Там було замало даних, тому виходило недонавчання, і отже доводилось підбирати зовсім інший датасет для тренування.

2.4. Подальша робота

1) Покращення функції аналізу сентименту

Для одного з датасетів функція аналізу сентименту на матриці нових передбачених настроїв відображає «1» у колонці «rhetorical question», проте не вдалось зарахувати в загальну кількість count (рисунок 15 (1)).

Для іншого з датасетів певні пораховані результати з count відображаються (рисунок 15 (2)).

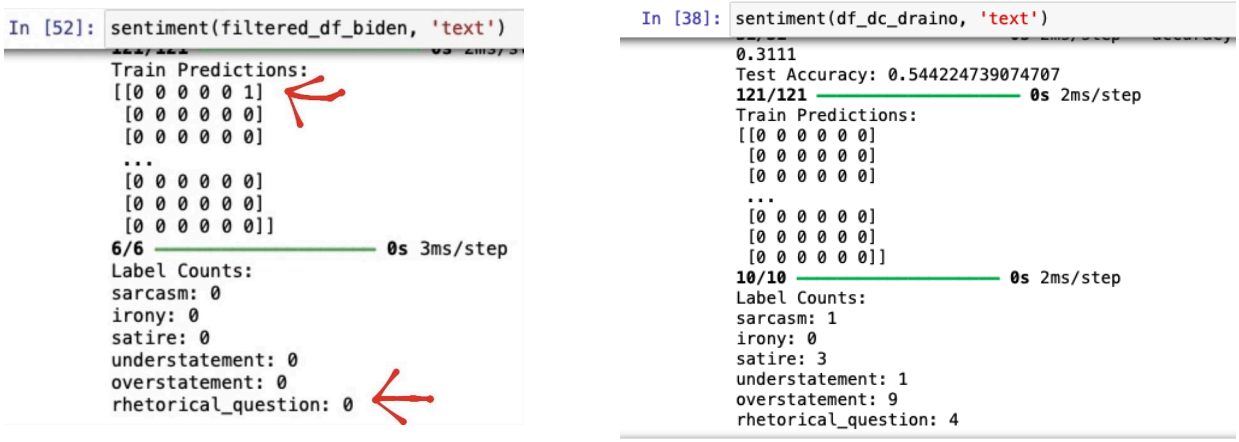

Рис. 15. Результати виконання функції аналізу сентименту (1) та (2).

ВИСНОВКИ

У ході роботи ми розглядали машинне навчання для ідентифікації впливових осіб і вимірювання їхнього впливу. Нам вдалося розробити функцію для соціального дослідження з використанням машинного навчання й моделі нейронної мережі, а саме функцію з обчисленням коефіцієнту впливу особи, базуючись на кількості вподобань та поширень її дописів.

Нам вдалося розробити функцію з визначення сентименту текстів, яка використовує машинне навчання й нейронну мережу, розподіляючи настрої тексту на шість різних груп.

Ми також проаналізували датасети; провели NLP-аналіз текстів кількох політичних осіб та визначили, що їхні дописи мають політично-публіцистичний характер.

Машинне навчання сьогодні є найбільш актуальним та перспективним напрямком розвитку технологій, який відкриває нам дуже багато нових можливостей у різних сферах.

ДЖЕРЕЛА:

1. «What is machine learning (ML)?» IBM, https://www.ibm.com/topics/machine-learning
2. «Underfitting and Overfitting», 11 March 2024, https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/
3. «Artificial Intelligence and Machine Learning for EDGE Computing», Ashish Tiwari, 29 April 2022, https://www.sciencedirect.com/science/article
4. «10 Top Machine Learning Algorithms», Datacamp Staff, February 2024, https://www.datacamp.com
5. «Introduction to Logistic Regression - Sigmoid Function, Code Explanation», Dinesh Kumawat, 21 Aug 2019, https://www.analyticssteps.com/blogs
6. «Reinforcement Learning: A Survey», L. P. Kaelbling M. L. Littman A. W. Moore, 1 May 1996, https://www.jair.org/index.php/jair
7. «The Machine Learning Process», Jeff Saltz, 10 April 2024, https://www.datascience-pm.com/machine-learning-process/

ДОДАТКИ

Додаток А

```
import pandas as pd
import numpy as np
import plotly.graph_objects as go
import matplotlib.pyplot as plt

df_biden = pd.read_csv('/Users/vikavika/Desktop/dyplomnarobota/csv_files/biden.csv')
followers_biden = 21000000 # у подальшому для МЛ нам знадобиться кількість фоловерів кожного з них
df_biden.shape[0]

df_harris = pd.read_csv('/Users/vikavika/Desktop/dyplomnarobota/csv_files/harris.csv')
followers_harris = 6000000
df_harris.shape[0]

df_pence = pd.read_csv('/Users/vikavika/Desktop/dyplomnarobota/csv_files/pence.csv')
followers_pence = 5000000
df_pence.shape[0]

df_trump = pd.read_csv('/Users/vikavika/Desktop/dyplomnarobota/csv_files/trump.csv')
followers_trump = 88000000
df_trump.shape[0]

df_dc_draino = pd.read_csv('/Users/vikavika/Desktop/dyplomnarobota/csv_files/dc_draino.csv')
followers_draino = 1100000
df_dc_draino.shape[0]

sum_likes_biden = df_biden['favorite_count'].sum()
sum_retweets_biden = df_biden['retweet_count'].sum()
print("Sum of likes of Joe Biden tweets:", sum_likes_biden)
print("Sum of retweets of Joe Biden tweets:", sum_retweets_biden)

sum_likes_harris = df_harris['favorite_count'].sum()
sum_retweets_harris = df_harris['retweet_count'].sum()
print("Sum of likes of Kamala Harris tweets:", sum_likes_harris)
print("Sum of retweets of Kamala Harris tweets:", sum_retweets_harris)
```

```
sum_likes_pence = df_pence['favorite_count'].sum()
sum_retweets_pence = df_pence['retweet_count'].sum()
print("Sum of likes of Mike Pence tweets:", sum_likes_pence)
print("Sum of retweets of Mike Pence tweets:", sum_retweets_pence)

sum_likes_trump = df_trump['favorite_count'].sum()
sum_retweets_trump = df_trump['retweet_count'].sum()
print("Sum of likes of Donald Trump tweets:", sum_likes_trump)
print("Sum of retweets of Donald Trump tweets:", sum_retweets_trump)

sum_likes_draino = df_dc_draino['likes'].sum()
sum_coments_draino = df_dc_draino['comments'].sum()
print("Sum of likes of Draino tweets:", sum_likes_draino)
print("Sum of comments of Draino tweets:", sum_coments_draino)

users = ['Joe Biden', 'Kamala Harris', 'Mike Pence', 'Donald Trump']
likes = [sum_likes_biden, sum_likes_harris, sum_likes_pence, sum_likes_trump]
retweets = [sum_retweets_biden, sum_retweets_harris, sum_retweets_pence,
sum_retweets_trump]
fig = go.Figure()
fig.add_trace(go.Bar(
    x=users,
    y=likes,
    name='Likes',
    marker_color='blue'
))
fig.add_trace(go.Bar(
    x=users,
    y=retweets,
    name='Retweets',
    marker_color='orange'
))
fig.update_layout(
    title='Total Likes and Retweets of Tweets by Different Individuals',
    xaxis_title='Users',
    yaxis_title='Count',
    barmode='group'
)
fig.show()
```

```
users = ['Rogan OHandley']
likes = [sum_likes_draino]
comments = [sum_coments_draino]
fig = go.Figure()
fig.add_trace(go.Bar(
    x=users,
    y=likes,
    name='Likes',
    marker_color='orange'
))
fig.add_trace(go.Bar(
    x=users,
    y=comments,
    name='Comments',
    marker_color='blue'
))
fig.update_layout(
    title='Total Likes and Comments of Tweets of Rogan OHandley',
    xaxis_title='User',
    yaxis_title='Count',
    barmode='group'
)
fig.show()
```

Додаток Б

```
import math

x = df_biden['favorite_count']
y = df_biden['retweet_count']
plt.xlim(0,370000)
plt.ylim(0,40000)
plt.scatter(x,y)
plt.show()

x_v = sum(x)/len(x)
y_v = sum(y)/len(y)
sd_x = math.sqrt(sum([(x-x_v)**2 for x in x]))
sd_y = math.sqrt(sum([(y-y_v)**2 for y in y]))
cov = sum([(x-x_v)*(y-y_v) for x,y in zip(x,y)])
r_xy = cov/(sd_x*sd_y)
print('X_v:', x_v)
print('Y_v:', y_v)
print('sd_x:', sd_x)
print('sd_y:', sd_y)
print('r_xy:', r_xy)
```

Додаток В

```
def remove_retweets(df, column):
    df = df[~df[column].str.match(r'^RT\s?@|.*\bRT\b', case=False)]
    return df

filtered_df_pence = remove_retweets(df_pence, 'text')
filtered_df_pence.shape[0]

filtered_df_biden = remove_retweets(df_biden, 'text')
filtered_df_biden.shape[0]

filtered_df_harris = remove_retweets(df_harris, 'text')
filtered_df_harris.shape[0]

filtered_df_trump = remove_retweets(df_trump, 'text')
filtered_df_trump.shape[0]

filtered_df_draino = remove_retweets(df_dc_draino, 'text')
Filtered_df_draino.shape[0]
```

Додаток Г

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from nltk.corpus import stopwords
from wordcloud import WordCloud
import string
nltk.download('punkt')
nltk.download('stopwords')

def common_words_cloud(df):
    stop_words = set(stopwords.words('english'))
    # додаткові стоп-слова з якими виникли проблеми(неактуальні та не несуть семантичного значення)
    additional_stopwords = {'us', 'one', 'rt', 'https', 'last', 'across', 'let', 'get', 'could', 'pc', '88022', 'like', 'dc',  'got', 'even'}
    stop_words.update(additional_stopwords)

    all_words = []
    for tweet in df['text']:
        words = word_tokenize(tweet.lower())
        words = [word for word in words if word.isalnum() and word not in stop_words]
        all_words.extend(words)

    freq_dist = FreqDist(all_words)
    most_common = freq_dist.most_common(20)
    print("The most repetitive words in the dataset (excluding stop words) are:")
    for word, freq in most_common:
        print(f"{word}: {freq} times")

    # хмара найпоширеніших слів
    wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(dict(most_common))
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title('Top 20 Most Common Words in Tweets')
    plt.show()
```

Додаток Д

```
import tensorflow as tf
from sklearn.model_selection import train_test_split

def influential_score(df, column_likes, column_retweets, followers_of):
    # беремо масив всіх даних (лайки та ретвіти)
    likes = df[column_likes].values
    retweets = df[column_retweets].values
    # constant number of followers
    followers = followers_of

    # формула of influence score
    influence = likes * 0.5 + retweets * 0.2 + followers * 0.3

    # нормалізуємо features дані
    likes_norm = likes / np.max(likes)
    retweets_norm = retweets / np.max(retweets)
    followers_norm = np.full_like(likes, followers) / followers

    # features у один масив
    features = np.column_stack((likes_norm, retweets_norm, followers_norm))

    # тренувальні та тестувальні сети
    X_train, X_test, y_train, y_test = train_test_split(features, influence, test_size=0.2,
random_state=0)

    # створення моделі з трьома шарами
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(64, activation='relu', input_shape=(3,)),
        tf.keras.layers.Dense(32, activation='relu'),
        tf.keras.layers.Dense(1)
    ])

    # компіляція та тренування моделі. у ролі функції втрат використовуємо
середньоквадратичне відхилення.
    # оптимізатор Адам для градієнтного спуску, який потребує не багато пам'яті
    model.compile(optimizer='adam', loss='mean_squared_error')
    model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test))
```

```
# оцінка моделі
loss = model.evaluate(X_test, y_test)
print("Test Loss:", loss)

predictions = model.predict(X_test)

# приклад використання
sample_index = 0
sample_input = X_test[sample_index]
sample_output = y_test[sample_index]
predicted_output = model.predict(np.expand_dims(sample_input, axis=0))[0][0]

print("Sample Input (Likes, Retweets, Followers):", sample_input)
print("Sample Output (True Influence Score):", sample_output)
print("Predicted Output (Predicted Influence Score):", predicted_output)
```

Додаток Е

```
def sentiment(df, column):
    traindata = pd.read_csv('/Users/vikavika/Desktop/dyplomnarobota/csv_files/trainnew.csv')
    tweets = traindata["tweet"].values
    labels =
traindata[['sarcasm','irony','satire','understatement','overstatement','rhetorical_question']].values
    labels = labels.astype(float)

    print(labels)
    # розділення даних на навчальні та тестові сети
    X_train, X_test, y_train, y_test = train_test_split(tweets, labels, test_size=0.2,
random_state=42)

    # перетворення tweets -> strings
    X_train = X_train.astype(str)
    X_test = X_test.astype(str)

    # токенізація
    tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=10000, oov_token='<OOV>')
    tokenizer.fit_on_texts(X_train)
    X_train_seq = tokenizer.texts_to_sequences(X_train)
    X_test_seq = tokenizer.texts_to_sequences(X_test)
    X_train_pad = tf.keras.preprocessing.sequence.pad_sequences(X_train_seq, maxlen=100,
padding='post')
    X_test_pad = tf.keras.preprocessing.sequence.pad_sequences(X_test_seq, maxlen=100,
padding='post')

    # побудова моделі
    model = tf.keras.Sequential([
        tf.keras.layers.Embedding(10000, 16, input_length=100),
        tf.keras.layers.GlobalAveragePooling1D(),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dense(6, activation='sigmoid') # 6, бо шість колонок labels
    ])

    # компіляція та тренування моделі, за функцію втрат застосовуємо перехресну ентропію
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    model.fit(X_train_pad, y_train, epochs=10, batch_size=32, validation_data=(X_test_pad,
y_test))
```

```
    # оцінка моделі
    loss, accuracy = model.evaluate(X_test_pad, y_test)
    print("Test Accuracy:", accuracy)

    # оцінка прогнозів моделі на даних навчання
    train_predictions = model.predict(X_train_pad)
    train_predictions_binary = np.where(train_predictions >= 0.5, 1, 0)
    print("Train Predictions:")
    print(train_predictions_binary)

    # тут далі вже йде тестування та передбачення:
    new_tweets = df[column].values
    # токенізація
    new_tweets_seq = tokenizer.texts_to_sequences(new_tweets)
    new_tweets_pad = tf.keras.preprocessing.sequence.pad_sequences(new_tweets_seq,
maxlen=100, padding='post')

    predicted_labels = model.predict(new_tweets_pad)

    # додаємо передбачені labels як нові колонки до датасету
    for i, target in
enumerate(['sarcasm','irony','satire','understatement','overstatement','rhetorical_question']):
        df[f"predicted_{target}"] = (predicted_labels[:, i] > 0.5).astype(int)


    # рахуємо counts до кожного типу label
    label_counts = {}
    for target in ['sarcasm','irony','satire','understatement','overstatement','rhetorical_question']:
        label_counts[target] = df[f"predicted_{target}"].sum()

    # повертаємо результат
    print("Label Counts:")
    for target, count in label_counts.items():
        print(f"{target}: {count}")

    return df
```