Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему *Метод класифікації текстових даних на основі групування рішень моделей ансамблю*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент 2 курсу, група КНм-21-1* _____ *М.М. Стебелецький*
Курс, група виконавця | Підпис | Ініціали, прізвище

Керівник: *к.т.н., доцент кафедри КН* _____ *Е.А. Манзюк*
Науковий ступінь, посада | Підпис | Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КН* _____ *Р.О. Багрій*
Науковий ступінь, посада | Підпис | Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КН, д.т.н., професор _____ *О.В. Бармак*
Підпис | Ініціали, прізвище

7 грудня 2022 р.

Хмельницький 2022

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *магістр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*
_____
(підпис)
д.т.н., професор О.В. Бармак
«*01*» *вересня 2022* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

1. Тема кваліфікаційної роботи магістра: *«Метод класифікації текстових даних на основі групування рішень моделей ансамблю»*

2. Завдання видано студенту *Стебелецькому Мирославу Мироновичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *доцент кафедри КН Манзюк Едуард Андрійович*
(посада, прізвище, ім'я, по батькові)

4. Затверджено наказом університету від «*21*» *липня 2022* р. № *83*

5. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Мета роботи – розробка методу класифікації текстових даних на основі групування рішень моделей ансамблю.

Об'єктом дослідження є процес забезпечення високих якісних показників класифікації за оцінками повноти та точності.

Предметом дослідження є ансамблеві моделі та методи побудови класифікаційних мета-алгоритмів на основі групування рішень базових класифікаторів.

Виконавець:	*студент 2 курсу, група КНм-21-1*		*М.М. Стебелецький*
	Курс, група виконавця	Підпис	Ініціали, прізвище
Керівник:	*к.т.н., доцент кафедри КН*		*Е.А. Манзюк*
	Науковий ступінь, посада	Підпис	Ініціали, прізвище

Реферат

Кваліфікаційна робота магістра присвячена розробці методу класифікації текстових даних на основі групування рішень моделей ансамблю із використанням кластерного аналізу даних задля усереднення результатів базових класифікаторів ансамблю.

Актуальність теми. В магістерській роботі було розроблено та набув практичної реалізації метод створення мета-алгоритму із використанням агрегаційного механізму на основі неієрархічного методу кластеризації у завданнях бінарної класифікації текстового контенту.

Машинне навчання - це область дослідження, яка займається алгоритмами, що навчаються на основі прикладів.

Класифікація - це процес розподілу заданого набору даних на класи, який може бути виконаний як для структурованих, так і для неструктурованих даних. Процес починається з прогнозування класу заданих точок даних. Класи часто називають цільовими, мітками або категоріями. Класифікаційне прогнозне моделювання постає як завдання апроксимації функції відображення від вхідних змінних до дискретних вихідних змінних. Основна мета - визначити, до якого класу/категорії потраплять нові дані. Класифікаційні завдання, що мають лише дві класові мітки, називаються бінарною класифікацією. Зазвичай одна класова мітка позначає нормальний стан, в той час як інша відхилення від норми.

Дискретний розподіл ймовірностей, відомий як розподіл Бернуллі, розглядає ситуацію, коли певна річ має двійковий результат, зокрема, 0, або 1. З точки зору класифікації, це означає, що модель прогнозує ймовірність того, що зразок потрапить до класу 1, або до аномального стану. Модель, яка прогнозує імовірнісний розподіл Бернуллі для кожного випадку, часто використовується для представлення задачі бінарної класифікації. За допомогою використання ймовірнісних значень передбачень класифікаторів можна створити модель агрегації показників декількох базових алгоритмів-провісників у один цільовий мета-алгоритм, або ансамбль. Ансамблеве навчання - це процес, в якому кілька

моделей машинного навчання об'єднуються для отримання кращих результатів. Основна ідея полягає в тому, що результат, отриманий від комбінації моделей, може бути більш точним, ніж будь-яка окрема модель машинного навчання в групі. Задля об'єднання показників декількох алгоритмів класифікації використовують різні підходи, один з яких кластерний аналіз ймовірностей передбачень.

Основи кластерного аналізу спираються на один з найбільш фундаментальних способів аналізу і пізнання - групування «об'єктів» у «схожі» групи. Цей процес включає в себе ряд різних алгоритмів і методів для створення кластерів подібного роду. Він також є частиною управління даними в статистичному аналізі. Групуючи сукупність об'єктів, які мають схожі характеристики або ознаки, множини, які утворюються, називаються кластерами. Процес називається кластеризацією. Отже, використавши кластерний аналіз можна згрупувати рішення окремих моделей у кластери, потім усереднити центроїди кожного з кластерів, після чого отриманий результат вважається усередненим або оптимізованим. Слід зазначити, що групуючи моделі варто брати до уваги «сумісність» кожної з них у комбінації з іншими. Показником продуктивності агрегованого рішення базових алгоритмів класифікації може виступати коефіцієнт кореляції між оптимізованим рішенням ансамблю та істинним значенням.

Коефіцієнт кореляції - це число від -1 до 1, яке показує силу та напрямок зв'язку між змінними. Іншими словами, він відображає, наскільки подібними є вимірювання двох або більше змінних у наборі даних. Суть кореляційного дослідження полягає в тому, щоб визначити, чи пов'язані зміни однієї змінної зі змінами іншої. Таким чином, отримавши коефіцієнт кореляції між рішенням ансамблю та істинним значенням, встановлюється якість зв’язку моделей у комбінації та їхня наближеність до правильних передбачень. Знайшовши найбільш задовільну за признаком кореляційних зв’язків комбінацію моделей, отримується оптимальний ансамбль моделей.

Метою дослідження є розробка метода побудови ансамблів моделей для класифікації даних на основі кореляційних зв'язків рішень. Використовується агрегаційний алгоритм, який усереднює показники передбачень базових моделей у кожній унікальній ансамблевій комбінації. Додатковим завданням являється вирахування кореляційних показників між результатами алгоритмів класифікації та істинними значеннями, ці показники повинні вираховуватись для кожної унікальної комбінації моделей у ансамблі.

Задля наглядності результативності наукової роботи, слід реалізувати систему візуалізації показників основних метрик алгоритмів. Візуалізація показників – важлива частина роботи, оскільки на базі цього можна наочно оцінити результати моделей, провести аналіз задля подальшого дослідження.

Для досягнення зазначеної мети визначені наступні завдання:

– провести аналіз: сучасних підходів, що базуються на ансамблях, зіставлення результативності застосування відомих методів у досліджуваній предметній області, результативності запропонованих рішень шляхом здійснення відповідних експериментальних досліджень на відомих масивах даних;

– покращити результативність класифікаційної системи, зокрема, покращенню підлягають такі метрики системи, як точність та повнота;

– розробити підхід агрегатування базових алгоритмів класифікації та подальшого усереднення їхніх результатів;

– реалізувати інформативну систему візуалізації даних за допомогою двох та трьох-вимірного простору. За допомогою візуалізації проглядаються всі необхідні закономірності.

Об'єктом дослідження є процес забезпечення високих якісних показників класифікації за оцінками повноти та точності.

Предметом дослідження є ансамблеві моделі та методи побудови класифікаційних мета-алгоритмів на основі групування рішень базових класифікаторів.

Наукова новизна одержаних результатів. В результаті проведеної роботи були отримані наступні результати:

– запропоновано інноваційний метод групування моделей в унікальні набори та подальше «відсіювання» результатів моделей задля створення ансамблів та агрегування результату в середині кожного ансамблю;

– розроблено метод агрегації моделей у задачах бінарної класифікації на основі кластерного аналізу;

– розроблено метод оцінювання ансамблів з використанням кореляційних показників щодо передбачень базових класифікаторів та оригінальних значень;

– реалізовано систему бінарної класифікації, яка містить модуль візуалізації основних метрик у дво та тривимірному просторах для аналізу отриманих результатів та подальшого дослідження.

Існує багато різноманітних готових ансамблевих методів для завдань класифікації у машинному навчанні, проте, питанню як агрегувати прогнози окремих членів ансамблю приділялось мало уваги. У цій роботі представляється формальна структура ансамблевої бінарної класифікації, в якій виділяється основний підхід, а саме: «передбачити, а потім об’єднати». В цьому випадку члени ансамблю спочатку роблять прогнози, після чого дані прогнози об’єднуються з використанням оптимізаційних методів.

Даний підхід узагальнює методи «голосування», які зазвичай використовуються для багатоелементних ансамблів, він дозволяє явно вирахувати цільовий показник ефективності.

Практичне значення одержаних результатів. Наукова робота висвітлює проблему підвищення точності прогнозування бінарної класифікації з використанням алгоритмів машинного навчання.

Основою інформаційної системи бінарної класифікації виступає ансамблева модель. Ця модель, в свою чергу, містить набір унікальних комбінацій базових класифікаторів – свого роду алгоритмічні примітиви. Ансамблева модель може розглядатись як деякий мета-алгоритм, який складається із унікальних наборів алгоритмів класифікації машинного навчання (ML).

Завданням ансамблевої моделі являється знаходження такої комбінації базових алгоритмів класифікації, яка б давала найвищі показники

результативності. Результативність оцінюється згідно з основними метриками ML у завданнях класифікації.

Особливістю цього дослідження є знаходження коефіцієнтів кореляцій базових моделей у кожній комбінації. За допомогою величини кореляцій встановлюється залежність між передбаченням класифікатора (базова модель) та істинним значенням, в результаті чого відкривається простір для подальших досліджень щодо покращення ансамблевої моделі (мета-алгоритму).

Апробація кваліфікаційної роботи.

Основні положення і результати роботи опубліковані в збірнику наукових праць – Манзюк Е.А. Метод побудови ансамблів моделей для класифікації даних на основі кореляційних зв'язків рішень / Стебелецький М.М., Манзюк Е.А., Скрипник Т. К., Багрій Р.О. // Збірник наукових праць за матеріалами Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук - 2022» Хмельницький, 2022, – С. 10-30. Покращення результативності класифікації методом ансамблевої агрегації / Стебелецький М.М., Манзюк Е.А., Скрипник Т. К., Багрій Р.О. // Всеукраїнська конференція АПКН-2022, Хмельницький, 31 жовтня, – С. 22-27.

Структура та обсяг роботи. Кваліфікаційна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із [39] найменувань та 3 додатків. Загальний обсяг кваліфікаційної роботи магістра становить [119] сторінок, з них [80] сторінок основного тексту та [39] сторінок додатків. У роботі наведено [43] рисунків та [2] таблиць.

Ключові слова: підвищення точності, бінарна класифікація, ансамблева модель, інформаційна система, машинне навчання, коефіцієнт кореляції, унікальна комбінація, модель, алгоритм класифікації.

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
МН	Машинне навчання
ШІ	Штучний інтелект
SDK	Software development kit
КС	Класифікаційна система
NLP	Natural Language Processing
TF-IDF	Term Frequency – Inverse Document Frequency
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
SVM	Support Vector Machine
KNN	K-Nearest Neighbors

Вступ

Ансамблеве навчання - це процес, за допомогою якого кілька моделей, таких як класифікатори або експерти, стратегічно формуються і об'єднуються для вирішення певної проблеми обчислювального інтелекту. Ансамблеве навчання в першу чергу використовується для поліпшення (класифікації, прогнозування, апроксимації функцій і т.д.) продуктивності моделі або зменшення ймовірності невдалого прогнозу поганої моделі. Інші застосування ансамблевого навчання включають визначення достовірності рішення, прийнятого моделлю, вибір оптимальних (або близьких до оптимальних) ознак, злиття даних, інкрементне навчання, навчання в нестаціонарному режимі та виправлення помилок.

Ансамблеві системи, або системи множинних класифікаторів використовуються часто у повсякденному житті, варто лиш глянути на деякі процеси під особливим «кутом». Наприклад, люди запитують думки кількох експертів перш ніж прийняти рішення, зокрема: читають відгуки користувачів перед покупкою товару (особливо дорогого), оцінюють майбутніх працівників, перевіряючи їхні рекомендації, тощо. Фактично, навіть ця робота перед тим, як бути прийнятою до публікації, проходить рецензування кількома експертами. У кожному конкретному випадку остаточне рішення приймається шляхом об'єднання індивідуальних рішень кількох експертів. При цьому першочерговою метою є звести до мінімуму похибку.

Процес об’єднання базових експертів (моделей або класифікаторів) є одним із найважливіших під час створення мета-алгоритму. Результати всіх окремих моделей об'єднуються в єдиний прогноз в рамках остаточної моделі. З точки зору регресії, результат є просто середнім значенням прогнозованих значень результатів. З точки зору класифікації, обирається категорія з найвищою частотою вихідних даних. Прогнози моделі піддаються агрегуванню, щоб об'єднати їх для остаточного розрахунку. Агрегування може здійснюватися на основі загальної суми отриманих прогнозів або на основі ймовірності прогнозів.

Розділ 1 Аналіз методів класифікації та побудови ансамблів моделей

1.1 Опис предметної області

Компанії мають обробляти більше даних, ніж будь-коли раніше. У той час як певні процедури, такі як юридичні та бухгалтерські дані, потребують кваліфікованих фахівців з багаторічним досвідом роботи, інші вимагають базових видів групування, фільтрації та аналізу.

Обробка природної мови (NLP) та машинне навчання (МН) - обидві підгрупи штучного інтелекту (ШІ) - є двома найбільш перспективними технологіями, що з'явилися в останні роки [1, 2, 3]. Ці технології можуть виконувати класифікацію тексту - інтелектуальну категоризацію тексту на основі його змісту.

Класифікація тексту - це техніка машинного навчання, яка присвоює набір заздалегідь визначених категорій відкритому тексту [4]. Приклад класифікації даних зображений на рисунку 1.1. Іншими словами, це явище маркування неструктурованих текстів відповідними тегами, які прогнозуються з набору заздалегідь визначених категорій. Наприклад, класифікація текстів використовується для фільтрації спаму та неспамових листів. Це використовується постачальниками послуг електронної пошти, такими як Alphabet та Microsoft, для своїх поштових сервісів [5,6]. Можна помітити, що у поштовій скриньці немає беззмістовних електронних листів, оскільки постачальник послуг фільтрує деякі небажані електронні листи у папку "Спам". Це робиться шляхом класифікації тексту в електронному листі; текст в електронному листі перевіряється постачальником послуг електронної пошти за допомогою моделей на основі NLP для розділення вхідних повідомлень.

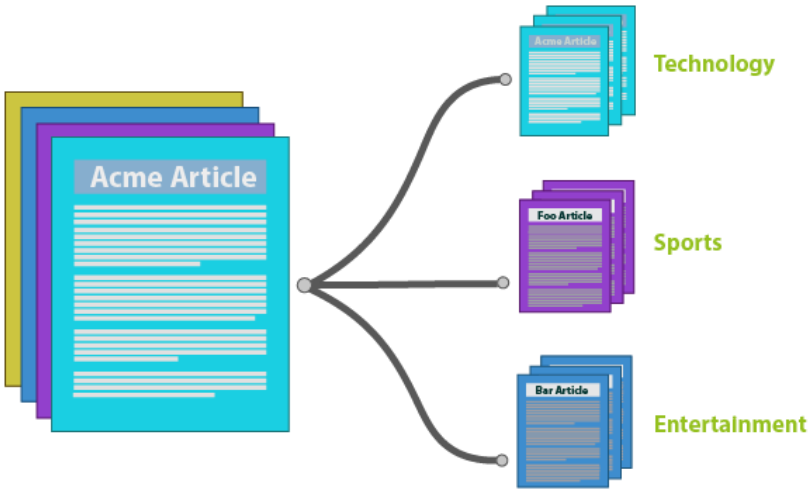


Рисунок 1.1 – Класифікація текстів на відповідні категорії

Класифікація тексту може здійснюватися двома способами: вручну або автоматично. При ручній класифікації тексту людина-спостерігач оцінює зміст тексту і класифікує його належним чином. Цей метод може забезпечити відмінні результати, але він дуже трудомісткий і дорогий.

Автоматична класифікація тексту поєднує в собі машинне навчання, обробку природної мови та інші підходи на основі штучного інтелекту, щоб класифікувати текст швидше, ефективніше і точніше.

Існує багато підходів до автоматичної класифікації текстів, але всі вони відносяться до трьох типів систем:

– системи на основі правил;

– системи на основі МН;

– гібридні системи;

Підходи, засновані на правилах, класифікують текст в організовані групи за допомогою набору створених вручну лінгвістичних правил [7]. Ці правила вказують системі використовувати семантично релевантні елементи тексту для визначення відповідних категорій на основі його змісту. Наприклад, припустимо, потрібно класифікувати статті на категорії "Галузь" і "Продукт". Перше що потрібно зробити, так це визначити список ключових слів, які відносяться до кожної з цих категорій. Тепер можна встановити правила для категоризації, такі як певна кількість ключових слів з однієї із двох категорій або певне

співвідношення термінів у двох категоріях. Потім статті можуть бути віднесені до відповідної категорії з використанням цих критеріїв.

Системи, засновані на правилах, є зрозумілими для людини і можуть бути вдосконалені з часом. Але такий підхід має певні недоліки. По-перше, ці системи вимагають глибоких знань предметної області. Вони також є трудомісткими, оскільки створення правил для складної системи може бути досить складним завданням і, як правило, вимагає багато аналізу та тестування. Ці системи також складні в обслуговуванні і погано масштабуються, оскільки додавання нових правил може вплинути на роботу вже існуючих.

Класифікація тексту за допомогою машинного навчання базується на попередніх спостереженнях, а не на правилах, створених людиною [8]. Алгоритми машинного навчання розуміють різноманітні кореляції між бітами тексту, а також розуміють, що на основі навчальних даних (тобто тексту) очікується конкретний вихід (тобто теги) для конкретного входу шляхом використання навчальних даних (тобто тексту). Коли використовується термін "тег" у цьому матеріалі, мається на увазі певну групу або категорію, до якої можна віднести будь-який текст. Щоб навчити класифікатор машинного навчання, спочатку потрібно виконати вилучення ознак. Вилучення ознак зображене на рисунку 1.2 Це означає перетворення фрагментів тексту в математичну структуру у вигляді вектора. Одним з найпоширеніших способів є «мішок» слів або bag of words, в якому вектор відображає частоту слова в заздалегідь визначеному лексиконі слів. Наприклад, якщо визначено, що словник містить наступні слова {This, is, the, not, awesome, bad, basketball}, і потрібно векторизувати текст "This is awesome", то отримається наступне векторне представлення цього тексту: (1, 1, 0, 0, 0, 1, 0, 0).

Модель може почати робити точні прогнози після того, як її навчать на достатній кількості прикладів. Візуалізація процесу передбачення зображена на рисунку 1.3. Класифікація тексту на основі машинного навчання є більш точною і швидкою, ніж методи, засновані на людських правилах. Нові зразки завжди можуть бути позначені для вивчення нових класифікацій тексту, оскільки класифікаторами легко керувати.

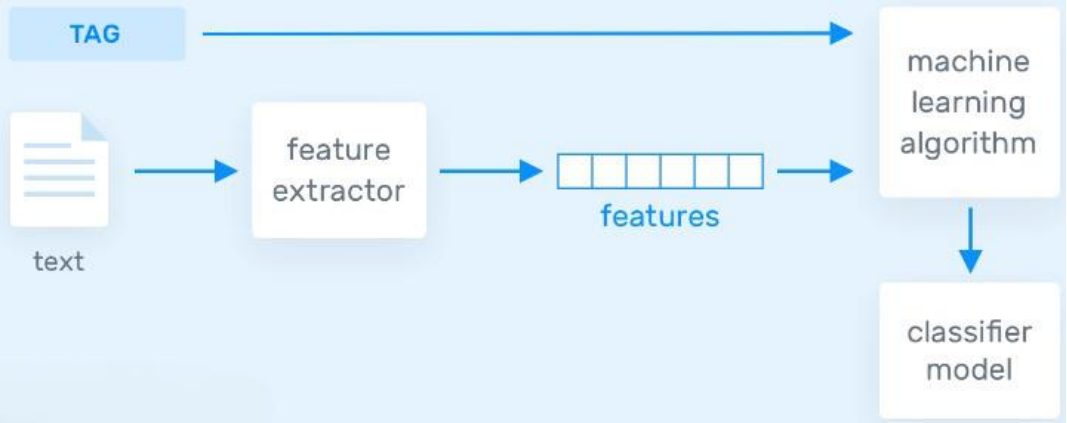


Рисунок 1.2 – Процес вилучення ознак та створення моделі

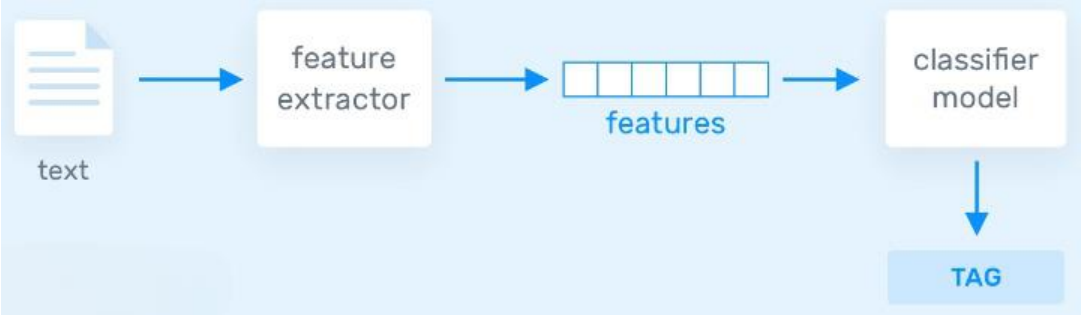


Рисунок 1.3 – Схематичне зображення процесу передбачення класифікатора

Гібридні системи представляють собою гібрид між системами класифікації тексту на основі правил і на основі машинного навчання, вони дають ще більш точні результати. Гібридні системи можна вдосконалити, просто додавши спеціальні правила для протиставлення тегів, які не були належним чином описані в базовому класифікаторі. Така комбінація також значно скорочує роботу, необхідну для маркування даних.

Можна зробити висновок, що аналіз текстів є новою темою досліджень в цілому. Класифікація текстів є чудовим інструментом для створення передових робочих процесів та управління даними.

1.2 Методи класифікації текстових даних

Машинне навчання - це галузь знань, яка присвячена алгоритмам, що навчаються на прикладах. Класифікація - це процес, що використовує алгоритми машинного навчання, які вчаться присвоювати мітку класу прикладам з проблемної області. Існує багато різних типів підходів до класифікації, з якими стикаються в МН і спеціалізовані підходи до моделювання, які можуть бути використані для кожного з них. У машинному навчанні класифікація відноситься до завдання прогнозного моделювання, де мітка класу прогнозується для даного прикладу вхідних даних. До завдань прогнозування можна віднести наступні:

– класифікація матеріалу на предмет спаму;

– виявлення літер алфавіту в рукописному тексті;

– аналіз поведінки користувача інформаційної системи та класифікація його дій.

З точки зору моделювання, класифікація вимагає навчального набору даних з багатьма прикладами вхідних і вихідних даних, на яких можна вчитися. Модель використовує навчальний набір даних і обраховує, як найкраще зіставити приклади вхідних даних з конкретними мітками класів. Таким чином, навчальний набір даних повинен бути достатньо репрезентативним для проблеми і містити багато прикладів кожної мітки класу.

Існує чотири основні методи у завданнях класифікації контенту:

– бінарний (віднесення даних до одного з двох класів);

– мультикласовий (віднесення даних до одного з багатьох класів);

– мультимітковий (дані можуть мати більше ніж 1 клас, віднесення до множини);

– імбалансний (дані нерівномірно розподілені між класами).

Бінарна класифікація відноситься до тих класифікаційних завдань, які мають дві класифікаційні ознаки [9]. У випадку задачі бінарної класифікації метою є класифікація вхідних даних на дві взаємовиключні категорії. Навчальні дані в такій ситуації маркуються в двійковому форматі: істина і брехня;

позитивний і негативний; 0 і 1; спам і не спам і т.д. в залежності від задачі, що розглядається. Зазвичай модель оцінки бінарної класифікації передбачає розподіл ймовірностей Бернуллі для кожного прикладу.

Розподіл Бернуллі - це дискретний розподіл ймовірностей, який охоплює випадки, коли результат події буде двійковим - 0 або 1 [10]. Для класифікації це означає, що модель прогнозує ймовірність того, що приклад належить до істинного класу або до аномального. На рисунку 1.4 зображений приклад віднесення об'єкта до одного із двох заданих класів (міток)

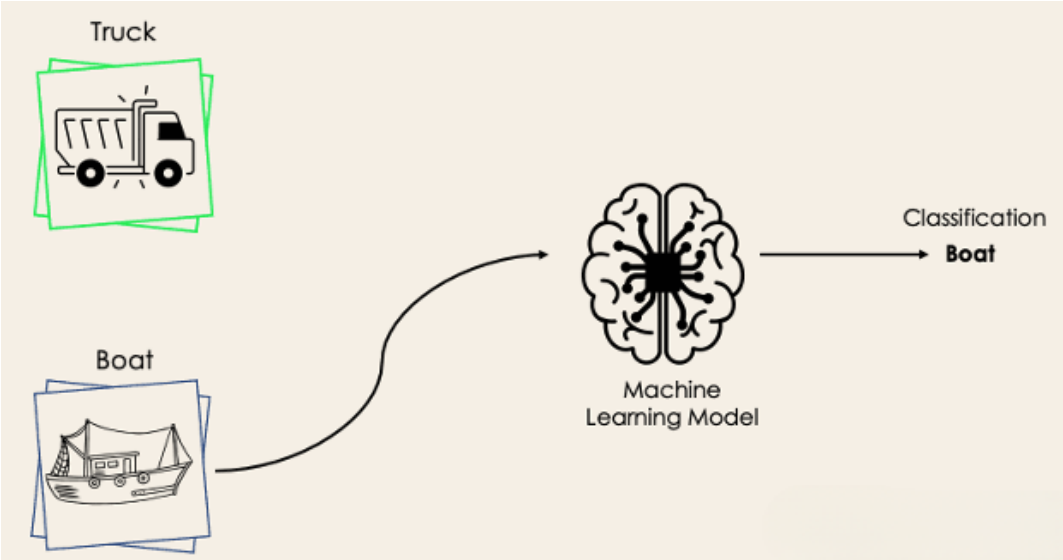


Рисунок 1.4 – Бінарна класифікація зображення

Алгоритми логістичної регресії та машин опорних векторів призначені для бінарних класифікацій [11,12]. Однак інші алгоритми, такі як K-найближчих сусідів та дерева рішень, також можуть бути використані для бінарної класифікації [13,14].

Мультикласова класифікація відноситься до тих класифікаційних завдань, які мають більше ніж дві класові позначки [15]. Прикладом може служити:

– розпізнавання обличчя;
– віднесення екземпляру рослини до виду;
– оптичне розпізнавання символів.

На відміну від бінарної класифікації, багатокласова класифікація не містить поняття нормального та відхиленого від норми результату. Замість цього приклади класифікуються як такі, що належать до одного з ряду відомих класів. Наприклад, використання моделі для ідентифікації видів тварин на зображеннях з енциклопедії є прикладом багатокласової класифікації, оскільки існує багато різних видів тварин, до яких можна віднести кожне зображення. Багатокласова класифікація також вимагає, щоб зразок мав тільки один клас (тобто слон - це тільки слон; він не є також лемуром). Приклад даної класифікації наведений на рисунку 1.5.

За винятком регресії, мультикласифікація є, мабуть, найпоширенішою задачею машинного навчання. При класифікації надається ряд навчальних прикладів, розділених на K окремих класів і будується модель машинного навчання, щоб передбачити, до якого з цих класів належать деякі раніше не помічені дані (наприклад, типи тварин з попереднього прикладу). Спостерігаючи за навчальним набором даних, модель вивчає закономірності, характерні для кожного класу, і використовує ці закономірності для прогнозування приналежності майбутніх даних.

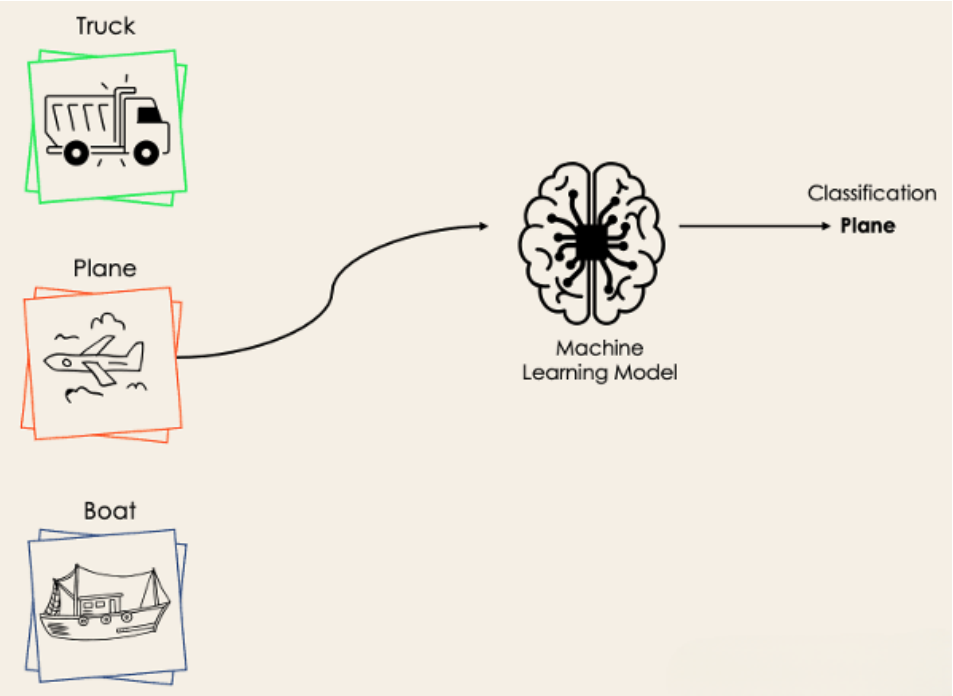


Рисунок 1.5 – Мультикласифікація об’єкта

Зазвичай завдання даної класифікації вирішується за допомогою моделі, яка прогнозує категоріальний розподіл ймовірностей для кожного конкретного прикладу. Категоріальний розподіл - це дискретний розподіл ймовірностей, який охоплює такий випадок, коли подія матиме категоріальний результат, наприклад, K в {1, 2, 3, ..., n}. Для класифікації це означає, що модель прогнозує ймовірність належності зразка до кожної мітки класу.

Алгоритми, розроблені для бінарної класифікації, можуть бути адаптовані для використання в мультикласифікаційних задачах.

Це передбачає використання стратегії підбору декількох моделей бінарної класифікації для кожного класу порівняно з усіма іншими класами (так звана стратегія «один проти решти») або однієї моделі для кожної пари класів (так звана стратегія «один проти одного»).

Стратегія «один проти одного» передбачає навчання стількох класифікаторів, скільки є пар міток. Якщо потрібно робити три-класову класифікацію, то необхідно три пари позначок, тобто три класифікатори, як показано на рисунку 1.6.

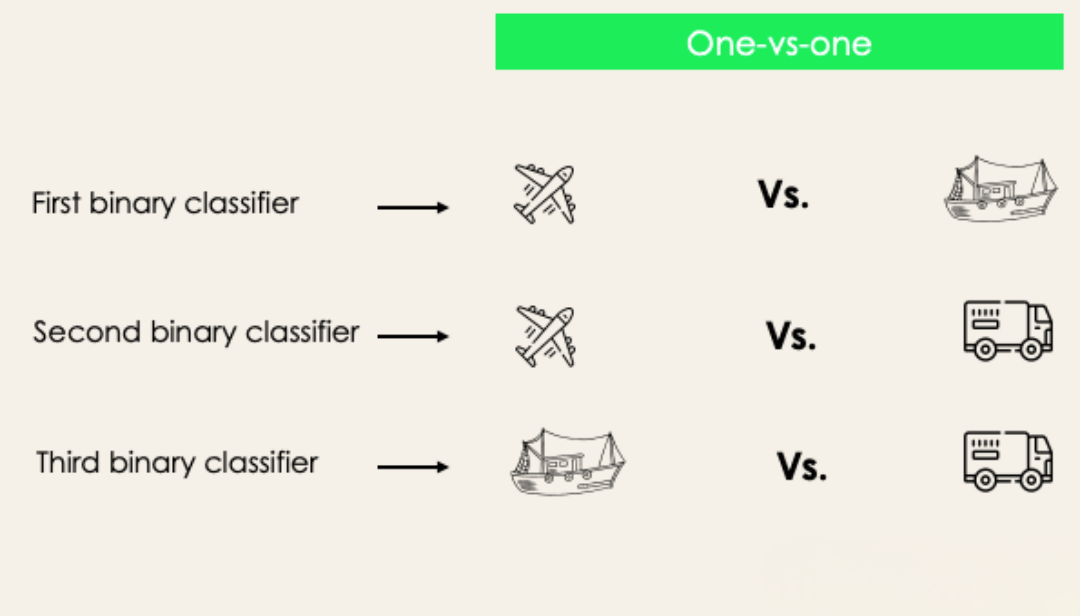


Рисунок 1.6 – Класифікація стратегією «один проти одного»

У загальному випадку для N міток потрібно $N_x(N-1)/2$ класифікаторів. Кожен класифікатор навчається на одному двійковому наборі даних, і остаточний клас прогнозується більшістю голосів між усіма класифікаторами. Підхід «один проти одного» найкраще працює для SVM та інших алгоритмів на основі ядер [16].

При використанні стратегії «один проти решти» потрібно для кожної пари класів підібрати одну бінарну модель класифікації. У загальному випадку для N міток необхідно мати N двійкових класифікаторів. Приклад класифікації за даною стратегією наведений на рисунку 1.7.

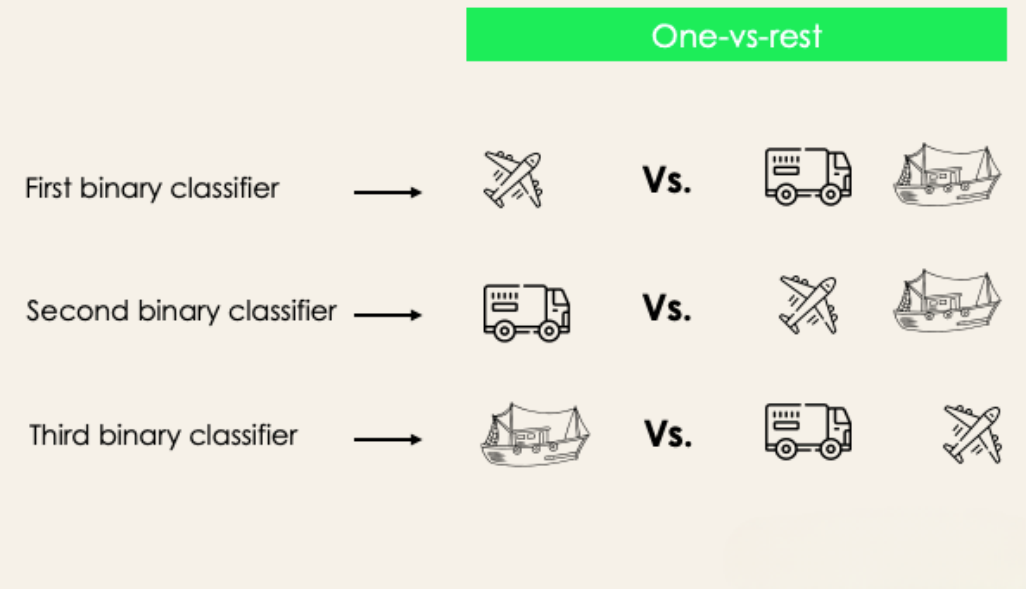


Рисунок 1.7 – Класифікація стратегією «один проти решти»

Мультиміткова класифікація відноситься до тих завдань класифікації, які мають дві або більше класифікаційних ознак, де одна або більше класифікаційних ознак можуть бути передбачені для кожного прикладу [17]. У задачах класифікації з декількома мітками потрібно передбачити 0 або більше класів для кожного вхідного зразку. У цьому випадку немає взаємного виключення, оскільки вхідний зразок може мати більше однієї ознаки. Це відрізняється від бінарної класифікації та мультикласифікації, де для кожного входу прогнозується одна позначка класу.

Такий механізм можна спостерігати в різних сферах, наприклад, при автоматичному тегуванні в NLP, де заданий текст може містити кілька тем.

Аналогічно в завданнях комп'ютерного зору, зображення може містити декілька об'єктів, як показано на рисунку 1.8, модель передбачила, що зображення містить: літак, човен, вантажівку і собаку.

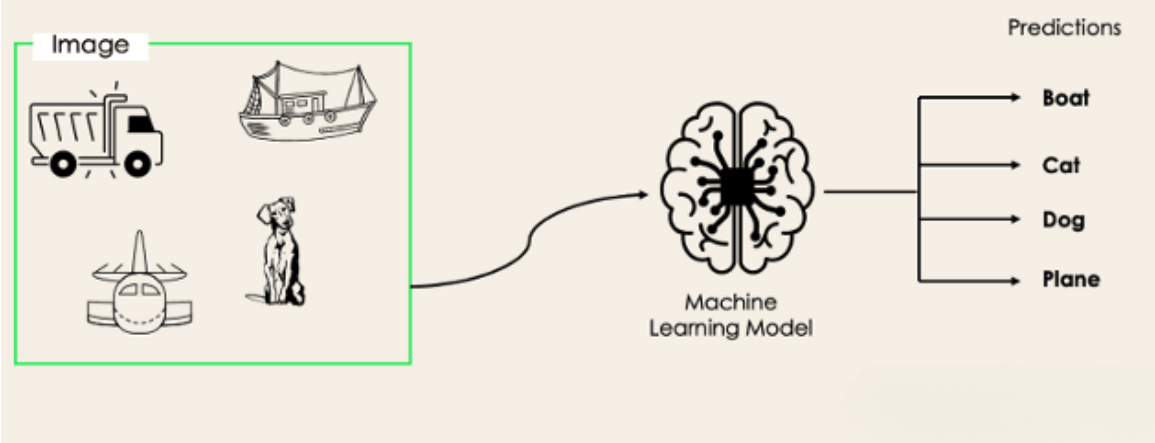


Рисунок 1.8 – Приклад мультиознакової класифікації

Зазвичай задачі класифікації з декількома мітками використовують модель, яка прогнозує декілька результатів, причому кожен результат прогнозується як розподіл ймовірностей Бернуллі. Це, по суті, модель, яка робить кілька прогнозів двійкової класифікації для кожного зразка. Інший підхід полягає у використанні окремого алгоритму класифікації для прогнозування міток для кожного окремого класу.

Незбалансована (імбалансна) класифікація відноситься до завдань класифікації, в яких кількість зразків у кожному класі розподілена нерівномірно. Як правило, задачі незбалансованої класифікації - це задачі бінарної класифікації, в яких більшість прикладів у навчальному наборі даних належать до нормального класу, а меншість прикладів - до ненормального. Для незбалансованої класифікації кількість зразків нерівномірно розподілена в кожному класі, що означає, що в навчальних даних може бути більше одного класу, ніж інших. На рисунку 1.9 розглянуто наступний сценарій три-класової класифікації, де навчальні дані містять 60% вантажівок, 25% літаків та 15% катерів.

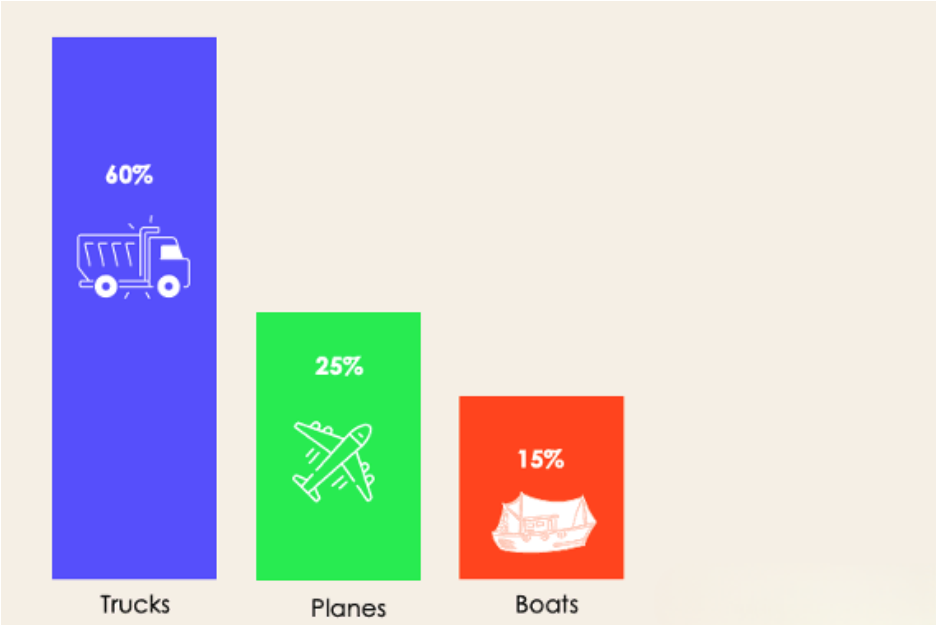


Рисунок 1.9 – Приклад незбалансованого розподілення даних

Ці проблеми моделюються як задачі бінарної класифікації, хоча можуть вимагати застосування спеціалізованих методів. Спеціалізовані методи можуть бути використані для зміни складу вибірок у навчальному наборі даних шляхом зменшення вибірки класу більшості або збільшення вибірки класу меншості.

1.3 Аналіз існуючих підходів ансамблевого моделювання

Коли приймаються важливі рішення, наприклад, вступ до університету, підписання трудового договору, необхідно отримати пораду. Потрібно зібрати якомога більше інформації та звернутися до різних експертів, оскільки такі рішення можуть вплинути на подальший хід дій. Прогнозування у машинному навчанні проводиться подібним чином. Моделі обробляють задані вхідні дані і видають результат. Результатом є прогноз, заснований на тому, яку закономірність моделі побачили під час навчання. У багатьох випадках однієї моделі недостатньо для отримання стабільних результатів.

У прогнозному моделюванні та інших видах аналізу даних одна модель, що базується на одній вибірці даних, може мати високу варіабельність або відверті неточності, які впливають на надійність її аналітичних висновків. Використання

конкретних методів моделювання може мати подібні недоліки. Поєднуючи різні моделі або аналізуючи кілька вибірок, фахівці з аналізу даних та інші аналітики даних можуть зменшити вплив цих обмежень і надати більш якісну інформацію особам, які приймають бізнес-рішення.

Алгоритми машинного навчання мають свої обмеження, і створення моделі з високою точністю є складним завданням. Якщо побудувати і об'єднати кілька моделей, то загальна точність може бути підвищена. Поєднання може бути реалізовано шляхом агрегування вихідних даних кожної моделі з двома цілями: зменшення похибки моделі та збереження її узагальненості. Поверхнева схема об'єднання базових алгоритмів передбачень зображена на рисунку 1.10. Реалізувати таке агрегування можна за допомогою певних методів. У деяких підручниках таку архітектуру називають мета-алгоритмами.

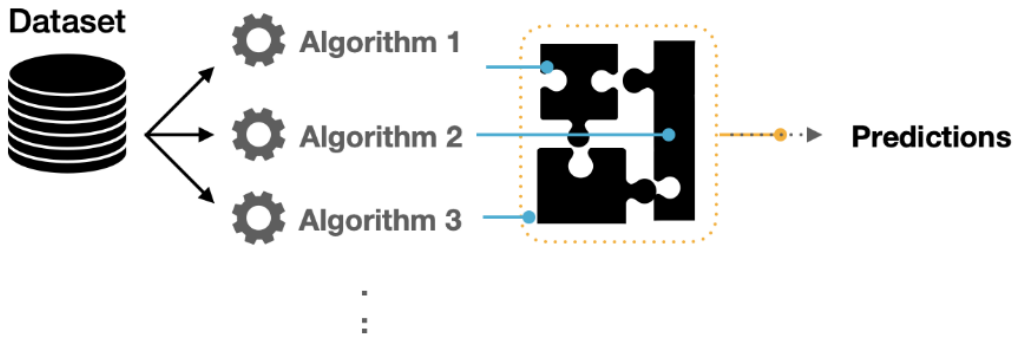


Рисунок 1.10 – Диверсифікація прогнозів моделі з використанням декількох алгоритмів

Ансамблеве моделювання - це процес об’єднання двох або більше пов'язаних, але різних аналітичних моделей з подальшим синтезом результатів в єдиний показник або спред з метою підвищення точності прогнозної аналітики [18].

Побудова ансамблевих моделей зосереджена не тільки на використанні якогомога меншого числа однакових алгоритмів. Можна взяти декілька ідентичних моделей, де кожна навчається певному шаблону, що спеціалізується на прогнозуванні одного аспекту. Ці моделі називаються слабкими учнями, які

можуть бути використані для отримання мета-моделі. У цій архітектурі ансамблевого навчання вхідні дані передаються кожному слабкому учню під час збору їхніх прогнозів. Об'єднані прогнози можуть бути використані для побудови остаточної ансамблевої моделі.

Під час побудови ансамблеввого алгоритму, або мета-алгоритму в задачах МН, зазвичай використовують такі популярні техніки об’єднання базових алгоритмів, зокрема: бегінг (Bagging), стекінг (Stacking), бустинг (Boosting).

Bagging - це тип ансамблевого методу, в якому один алгоритм навчання використовується на різних підмножинах навчальних даних, де вибірка підмножин виконується із заміною (бутстреп) [19]. На рисунку 1.11 зображений процес створення ансамблю даним методом. Бутстреп створює декілька наборів вихідних навчальних даних із заміною. Заміна дозволяє дублювати екземпляри зразків у наборі. Кожна підмножина має однаковий розмір і може використовуватися для паралельного навчання моделей.

Після того, як алгоритм навчений на всіх підмножинах, відбувається прогнозування шляхом агрегування всіх прогнозів, зроблених алгоритмом на різних підмножинах.

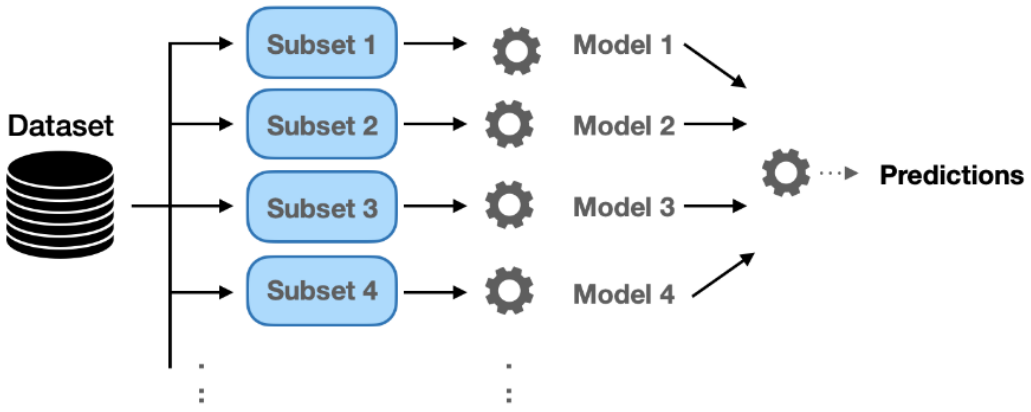


Рисунок 1.11 – Метод розподілення вхідних даних для отримання остаточних результатів шляхом об'єднання прогнозів від декількох моделей

Переваги bagging моделі:

– значно зменшує дисперсію, не збільшуючи зміщення передбачень в сторону конкретної вибірки даних;

– працює добре через різноманітність навчальних даних, оскільки вибірка робиться шляхом бутстрапінгу;

– якщо навчальний набір дуже великий, можна заощадити обчислювальний час, навчаючи модель на відносно меншому наборі даних, та все одно підвищити точність моделі;

– добре працює із невеликими наборами даних.

Недоліки:

– основним недоліком вважається те, що під час створення роздільних навчальних вибірок даних, деякі дані з корпусу даних можуть не потрапити у процес навчання моделі, що спричиняє деяку нестабільність.

Boosting - це ансамблевий метод покращення прогнозів моделі будь-якого алгоритму навчання [20]. Ідея бустінгу полягає в послідовному навчанні слабких учнів, кожен з яких намагається виправити свого попередника. Процес навчання зображений на рисунку 1.12. Слабкі учні послідовно виправляються своїми попередниками і в процесі навчання перетворюються на сильних учнів.

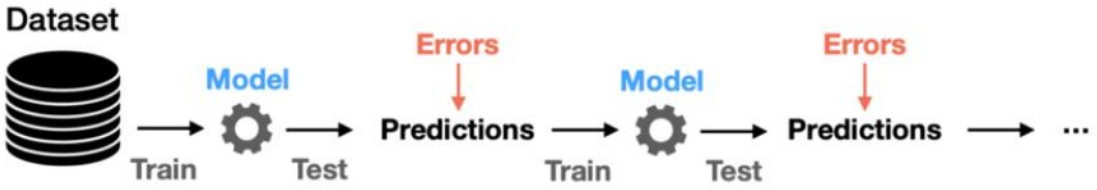


Рисунок 1.12 – Послідовне навчання моделі задля створення кращого провісника

Перевати boosting моделі:

– це гнучкий метод, який легко контролює надмірне навчання;

– доказово ефективний;

– універсальний - може бути застосований до широкого спектру проблем.

Недоліки:

– недоліком бустінгу є його чутливість до хибних передбачень, оскільки кожен класифікатор зобов'язаний виправляти помилки попередників. Таким чином, метод є надто залежним від помилок-викидів;

– іншим недоліком є те, що метод майже неможливо масштабувати. Це пов'язано з тим, що кожен оцінювач базує свою правильність на попередніх предикторах, що ускладнює оптимізацію процедури.

Stacking - це метод ансамблевого навчання, який поєднує декілька алгоритмів машинного навчання через мета-навчання, в якому алгоритми базового рівня навчаються на основі повного набору навчальних даних, а мета-модель навчається на кінцевих результатах роботи всієї моделі базового рівня в якості ознаки [21]. На рисунку 1.13 показується, що для навчання класифікаторів не береться окрема вибірка даних. Замість цього обирається весь набір даних на навчання для кожного окремого класифікатора. При цьому кожен класифікатор працює незалежно, що дозволяє отримати класифікатори з різними передбаченнями та результатами в цілому.

Рисунок 1.13 – Техніка стекінгу для отримання остаточного прогнозу в ансамблевій архітектурі.

Переваги методу stacking:

– перевага стекінгу полягає в тому, що він може використовувати можливості ряду добре працюючих моделей на задачі класифікації або регресії і робити прогнози, які мають кращу продуктивність, ніж будь-яка окрема модель в ансамблі;

– стекінг покращує точність прогнозування моделі.

Недоліки:

– оскільки для навчання береться весь набір даних для кожного окремого класифікатора, у випадку великих наборів даних час обчислень буде більшим, оскільки кожен класифікатор працює незалежно на великому наборі даних.

Виходячи із трьох вище зазначених методів ансамблювання базових моделей, можна дійти висновку, що стекінг у завданнях створення мета-алгоритму із використанням алгоритму виведення колективного рішення буде оптимальним варіантом, адже цей підхід дає можливість застосування власного методу агрегування передбачень базових класифікаторів. Важливим аспектом застосування стекінгу при ансамблюванні є те, що цей метод збільшує точність передбачень у мета-алгоритмі.

Коли об'єднується декілька алгоритмів щоб оптимізувати процес прогнозування для поєднання декількох моделей, потрібен метод агрегування. Можна використовувати три основні методи:

– максимальна кількість голосів: остаточний прогноз в цьому методі робиться на основі голосування більшості для задач класифікації;

– усереднення: зазвичай використовується для задач регресії, де прогнози усереднюються. Ймовірність передбачень також може бути використана, наприклад, для усереднення остаточної класифікації;

– середньозважене: іноді потрібно надати вагу деяким моделям/алгоритмам при створенні остаточних прогнозів.

1.4 Постановка завдання

Метою роботи є створення метода побудови мета-алгоритму, який складається з наборів ансамблів, які, в свою чергу, складаються з унікальних комбінацій базових класифікаторів. В кожному ансамблі використовується агрегаційний механізм на основі кластерного аналізу, який усереднює передбачення базових класифікаторів, після чого ансамблева модель має змогу видати оптимізоване рішення для порції даних. Результуючі показники кожного ансамблю аналізуються на предмет кореляційних зв’язків рішень, за допомогою даних показників мета-алгоритм визначає найкращу ансамблеву модель, в якій базові класифікатори показали найкращі результати спільної продуктивності. У висновку система визначає найкращий ансамбль у завданнях класифікації текстових даних на основі групування рішень моделей.

Для досягнення поставленої мети визначені наступні завдання дослідження:

– провести аналіз методів з класифікації текстових даних;

– розробити модель формування ансамблю прийняття рішень з класифікації даних;

– розробити метод класифікації даних на основі групування рішень моделей ансамблю;

– реалізувати інформаційну технологію класифікації текстових даних на основі ансамблю моделей;

– провести валідацію розробленого метода.

Висновки до розділу 1

У даному розділі проведено детальний огляд щодо методів та підходів у завданнях класифікації даних. Проаналізовано та наведено схематичні приклади застосування існуючих підходів ансамблевого моделювання. Визначено переваги та недоліки наявних методів об’єднання базових класифікаторів у єдину модель.

Розділ 2 Розробка методу обробки природної мови

2.1 Аналіз та загальні положення щодо методу опрацювання текстових даних

Обробка природної мови (NLP) - це підгалузь лінгвістики, комп'ютерних наук та штучного інтелекту, яка використовує алгоритми для інтерпретації та маніпулювання людською мовою. Ця технологія є однією з найбільш широко застосовуваних напрямів машинного навчання і має вирішальне значення для ефективного аналізу величезних обсягів неструктурованих, насичених текстом даних.

Людська мова наповнена неоднозначностями, які неймовірно ускладнюють написання програмного забезпечення, що точно визначає передбачуване значення текстових або голосових даних. Омоніми, омофони, сарказм, ідіоми, метафори, граматика і винятки, варіації в структурі речень - це лише деякі з особливостей людської мови, на вивчення яких люди витрачають роки, але які інформаційним системам необхідно навчитись розпізнавати і розуміти з самого початку, якщо ці системи повинні нести користь для користувачів.

Попередня обробка даних передбачає підготовку та очищення текстових даних для того, щоб машини могли їх аналізувати. Попередня обробка приводить дані в зручну для роботи форму і виділяє в тексті особливості, з якими може працювати алгоритм. Це можна зробити в кілька кроків, зокрема:

1. Токенізація. Текст розбивається на менші одиниці для роботи з ним.
2. Видалення стоп-слів. З тексту видаляються загальні слова, щоб залишилися унікальні слова, які несуть найбільше інформації про текст.
3. Маркування за частинами мови. Слова позначаються на основі частини мови, до яких вони відносяться, наприклад, іменники, дієслова та прикметники.
4. Лематизація та стеммінг. Слова для обробки зводяться до їх кореневих форм.
5. Векторизація текстових даних. Приведення у форму, яку може прийняти алгоритм.

Після того, як дані попередньо оброблені, розробляється алгоритм для їх обробки. Існує багато різних підходів обробки природної мови, але зазвичай використовуються два основних типи:

– базована на правилах система. Ця система використовує ретельно розроблені лінгвістичні правила. Даний підхід використовувався на ранніх етапах розвитку обробки природної мови, і використовується досі;

– система на основі МН. Алгоритми МН використовують статистичні методи. Вони вчаться виконувати поставлені завдання на основі навчальних даних, які їм подаються, і коригують свої методи в міру того, як опрацьовується більша кількість даних. Використовуючи поєднання машинного навчання, глибокого навчання та нейронних мереж, алгоритми обробки природної мови відточують свої власні правила шляхом повторної обробки та навчання.

Використовуючи обробку природної мови, алгоритми класифікації тексту можуть автоматично аналізувати текст, а потім призначати набір заздалегідь визначених тегів або категорій на основі його змісту. Схема робочого процесу та подальшого подання даних на алгоритм зображена на рисунку 2.1.

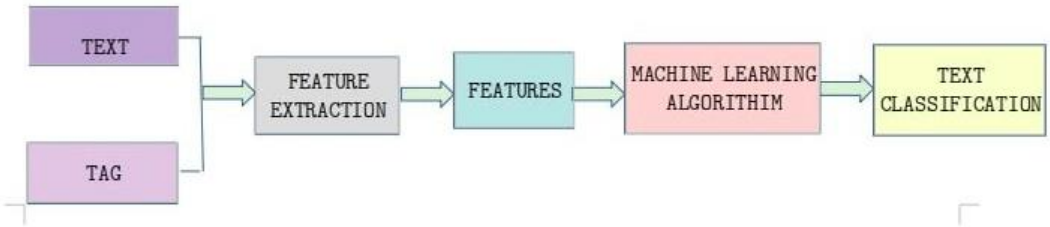


Рисунок 2.1 – Загальна схема NLP при класифікації

Варто зазначити, що більшість алгоритмів класифікації у МН оперують даними, які представлені у цифрових наборах значень, тому потрібен спосіб перетворення послідовностей слів в числові характеристики, цим займається векторизація. Найпростішою технікою векторизації тексту є метод Bag Of Words (BOW) [22]. Він складається зі списку слів, який називається словниковим запасом (часто це всі слова, які зустрічаються в навчальних даних). Потім,

враховуючи вхідний текст, виводиться числовий вектор, який є просто вектором значень кількості слів для кожного слова в словнику

2.2 Сегментація текстової інформації

Сегментація тексту або токенизація - це завдання розбиття тексту на осмислені сегменти. Ці сегменти можуть складатися зі слів, речень або розділів [23]. Токенизація - це перший крок у будь-якому конвеєрі НЛП. Вона має важливий вплив на решту роботи конвеєра. Токенизатор розбиває неструктуровані дані і текст природної мови на фрагменти інформації, які можна розглядати як дискретні елементи. Вміст токенів у документі можна використовувати безпосередньо як вектор, що представляє цей документ. Це дозволяє відразу перетворити неструктурований потік (текстовий документ) на числову структуру даних, придатну для машинного навчання. Ці дані також можуть бути використані безпосередньо машиною для того, щоб ініціювати потрібні рішення і поведінку. Або вони можуть бути використані в конвеєрі машинного навчання як ознаки, що призводять до більш складних рішень або моделей поведінки. Токенизація дозволяє відокремлювати речення, слова, символи або підслова. Коли розбивається текст на речення, це називається токенизацією речень. Для слів відповідно вживається термін токенизація слів.

Фундаментальним підходом до токенизації є розбиття тексту на слова. Однак при такому підході слова, які не входять до словника, розглядаються як невідомі. Сучасні моделі NLP вирішують цю проблему шляхом токенизації тексту на підсловники, які часто зберігають лінгвістичне значення (наприклад, морфеми). Таким чином, навіть якщо слово може бути невідомим для моделі, окремі підслова можуть зберігати достатньо інформації для того, щоб модель могла певною мірою вивести їх значення. Одна з таких технік токенизації підслів, що широко використовується і може бути застосована до багатьох інших моделей NLP, називається WordPiece [24]. Отримавши текст, WordPiece спочатку попередньо токенізує текст на слова (шляхом розбиття на розділові знаки та

пробіли), а потім токенизує кожне слово на підслова, які називаються фрагментами слів. Процес токенизації речення зображений на рисунку 2.2.

Рисунок 2.2 – Токенизація методом WordPiece на прикладі речення

Метод WordPiece використовує стратегію пошуку найдовшого співпадіння для токенизації окремого слова - тобто він ітеративно вибирає найдовший префікс з тексту, що залишився, який відповідає слову в словнику моделі. Цей підхід відомий як максимальна відповідність або MaxMatch [25]. Загальноприйняті обчислення підходу MaxMatch є квадратичними по відношенню до довжини вхідного слова (n). Це пов'язано з тим, що для сканування вхідних даних потрібні два вказівники: один для позначення початкової позиції, а інший для пошуку найдовшого підрядка, що відповідає лексемі словника в цій позиції.

Альтернативою алгоритму MaxMatch для токенизації WordPiece являється алгоритм LinMaxMatch, він має строго лінійний час токенизації [26]. Спочатку лексеми словника організовуються в дерево (так зване дерево префіксів), де кожне ребро дерева позначається символом, а шлях по дереву від кореня до деякого вузла являє собою префікс деякої лексеми в словнику. На рисунку 2.3 вузли зображено у вигляді кружечків, а ребра дерева - чорними суцільними стрілками. За допомогою дерева можна знайти лексему словника, яка відповідає вхідному тексту, рухаючись від кореня до вершин дерева, щоб символ за символом співпав з вхідним текстом. Цей процес називається зіставленням символів у дереві.

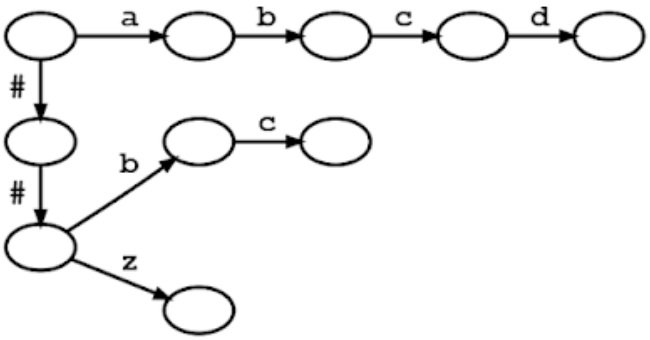


Рисунок 2.3 – Деревоподібна діаграма словника. Кола та стрілки позначають вузли та ребра вздовж дерева відповідно.

Алгоритм виконання WordPiece наступний:

1. Взяти достатньо великий корпус.
2. Визначити бажаний розмір словника підслів.
3. Розбити слова на послідовність символів.
4. Ініціалізувати словник всіма символами тексту.
5. Побудувати мовну модель на основі словника.
6. Повторювати крок 5 до тих пір, поки не буде досягнутий розмір словника підслів (визначений на кроці 2).

2.3 Очищення даних від шумів

Видалення стоп-слів є одним з найбільш часто використовуваних кроків попередньої обробки в різних процесах NLP. Ідея полягає в тому, щоб просто видалити слова, які часто зустрічаються у всіх документах в корпусі. Як правило, артиклі, прийменники, займенники, сполучники класифікуються як стоп-слова. Ці слова не мають значення в таких завданнях як пошук і класифікація інформації, що означає, що ці слова не є дуже важливими для аналізу.

Стоп-слова у великій кількості є в будь-якій людській мові. Видаляючи ці слова, відбувається видалення низькорівневої інформації з тексту для того, щоб приділити більше уваги важливій інформації. Щодо службових слів, то можна

сказати, що видалення таких слів не показує ніяких негативних наслідків на моделі, яку навчають для виконання поставленого завдання.

Алгоритм видалення шуму з текстових документів можна представити наступними кроками:

1. Текст цільового документа токенізується, а окремі слова зберігаються в масиві.
2. Зі списку стоп-слів зчитується одне стоп-слово.
3. Стоп-слово порівнюється з цільовим текстом у формі масиву з використанням методу послідовного пошуку.
4. Якщо є співпадіння, то слово з масиву видаляється, а порівняння продовжується до останнього елемента масиву.
5. Після повного видалення стоп-слова, зчитується інше стоп-слово зі списку стоп-слів і знову алгоритм виконує крок 2. Алгоритм працює безперервно до тих пір, поки не будуть порівняні всі стоп-слова.

2.4 Нормалізація порцій текстових даних

Нормалізація - це процес перетворення лексеми в її базову форму. У процесі нормалізації відмінювальна форма слова видаляється, щоб отримати базову форму. Нормалізація допомагає зменшити кількість унікальних лексем, присутніх у тексті, усунути варіації в тексті, а також очистити текст, видаливши надлишкову інформацію. Два популярних методи, що використовуються для нормалізації, - це стеммінг і лематизація.

Стеммінг - одна з найпоширеніших операцій попередньої обробки даних, яка виконується майже у всіх задачах з обробки природної мови [27]. Простіше кажучи, стеммінг - це процес видалення частини слова або скорочення слова до його основи чи кореня. Це не обов'язково означає, що слово скорочується до його словникового кореня. Використовується кілька алгоритмів, щоб вирішити, як відсікти слово. Цим, здебільшого, стеммінг відрізняється від лематизації, яка зводить слово до його словникового кореня, що є більш складним і потребує дуже

високого рівня знання мови. На рисунку 2.4 зображено результат стеммінгу для слів української мови та порівняння із основою слова. Варто зазначити, що використовуваний корпус даних у роботі складається із документів на англійській мові.

Слово	Основа	Результат стемінгу
ти	ти	ти
весна	весн	весн
міський	міськ	міськ
підводний	підводн	підводн
швидкий	швидк	швидк
бігати	бігати	бігати
безпритульними	безпритул	безпритульн
ортогональний	ортогонал	ортогональн
повільне	повіл	повільн
цивільним	цивіл	цивільн
дивними	дивн	дивн
критикувати	критикувати	критикувати
блиск	блиск	блиск
восени	восени	восен
бюро	бюро	бюро
перемагати	перемагати	перемагати
танцюючи	танцюючи	танцюючи
швидко	швидко	швидко
тут	тут	тут
авіадиспетчер	авіадиспетчер	авіадиспетчер

Рисунок 2.4 – Порівняння між ідеальним варіантом основи слова та результатом роботи алгоритму стемінгу

За своєю обчислювальною природою метод стеммінгу має притаманні обмеження. Процедура працює з окремими словами: вона не має доступу до інформації про їх граматичні та семантичні зв'язки.

При виконанні стеммінгу для слова, використовуються алгоритми різної варіації, найпопулярнішим є алгоритм «Porter Stemmer»[28]. «Porter Stemmer» - це алгоритм видалення суфіксів. Коротше кажучи, він використовує заздалегідь визначені правила для розбиття слів на їх базові форми. Зазвичай даний алгоритм працює з англомовними корпусами даних. На рисунку 2.5 зображено діаграму виконання даного алгоритму.

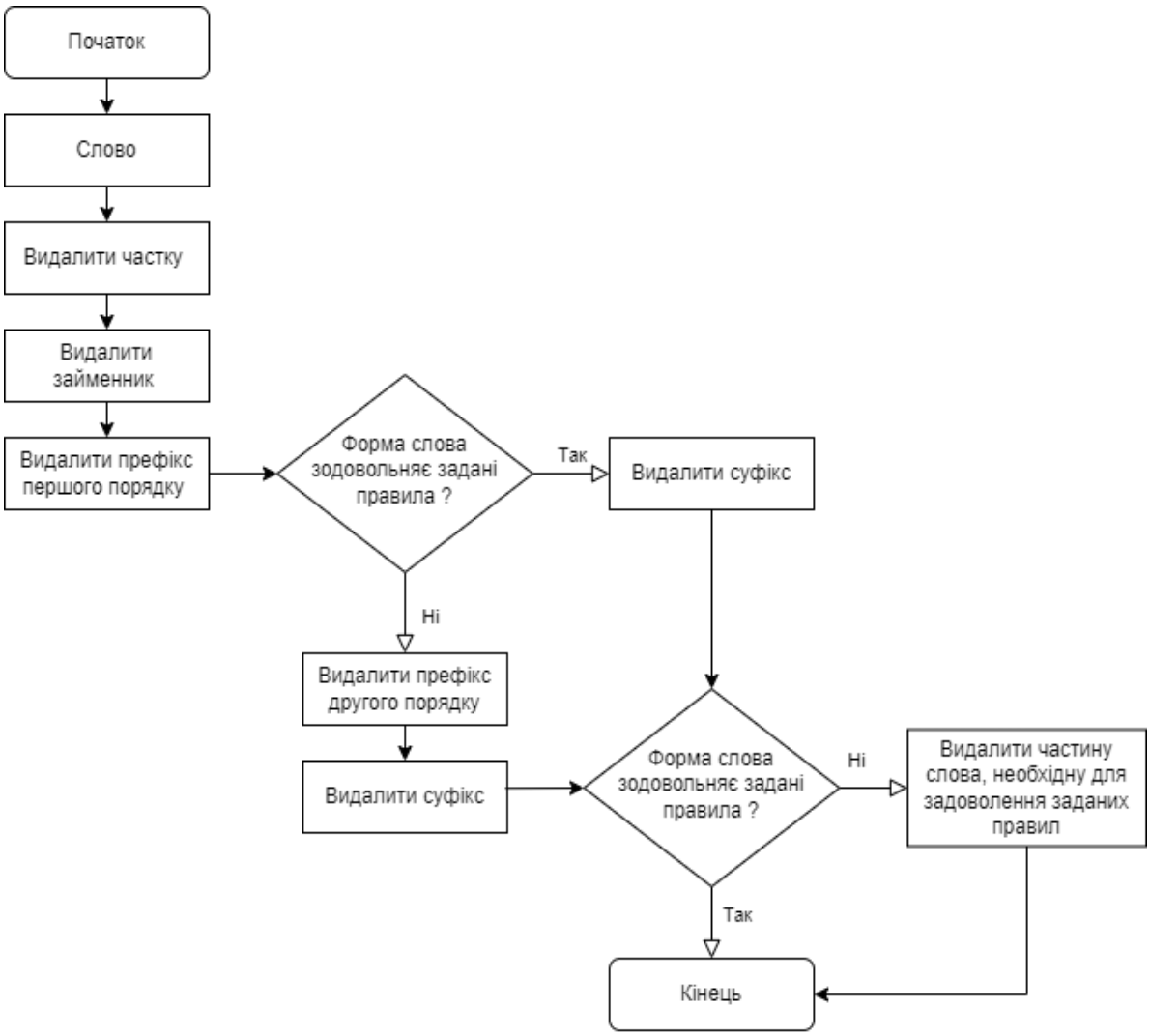


Рисунок 2.5 – Аналіз процесу стеммінгу за допомогою блок-схеми алгоритму Портера

У висновку можна сказати, що «Porter Stemmer» складається з декількох різних фаз. Ці фази застосовуються послідовно. В рамках кожної фази існують певні конвенції для вибору правил. Весь алгоритм Портера є невеликим, а отже, швидким і простим. Недоліком цього стеммера є те, що він підтримує лише англійську мову, і отриманий стем може бути як лінгвістично правильним, так і ні.

Підхід стеммінгу має суттєвий недолік, а саме, частина слова просто відсікається в кінці, щоб отримати основу слова. Безумовно, існують різні алгоритми, які використовуються для визначення того, скільки символів потрібно відрізати, але алгоритми фактично не знають значення слова в мові, до якої воно

належить. Справитись з даним недоліком під час нормалізації даних допоможе лематизація.

Лематизація - це техніка нормалізації тексту, що використовується в обробці природної мови, вона переводить будь-яке слово в кореневий формат, тобто в режим базового кореня. Лематизація відповідає за групування різних відмінюваних форм слів у кореневу форму, що має однакове значення. Лематизація зазвичай передбачає використання словникового та морфологічного аналізу слів, видалення відмінкових закінчень та відновлення словникової форми слова (леми). Морфологічний аналіз потребує вилучення правильної леми кожного слова. Для спрощення можна вважати, що лематизація - це лінгвістичний термін, що означає акт групування слів, які мають однаковий корінь або лему, але мають різні закінчення або похідні значення, щоб їх можна було аналізувати як одне ціле. Процес лематизації має на меті позбутися словотворчих суфіксів та префіксів з метою виведення словникової форми слова.

Підхід нормалізації порцій текстових даних методом лематизації може бути дещо повільнішим за інші методи, оскільки під час виконання алгоритму необхідно звертатись до джерела слів, які відповідають вхідному слову для того щоб знайти або співставити правильну морфологічну форму. Але цей підхід дає свої плоди, на рисунку 2.6 показано, наскільки даний метод краще справляється із приведенням слів англійської мови до базової форми в порівнянні із стеммінгом.

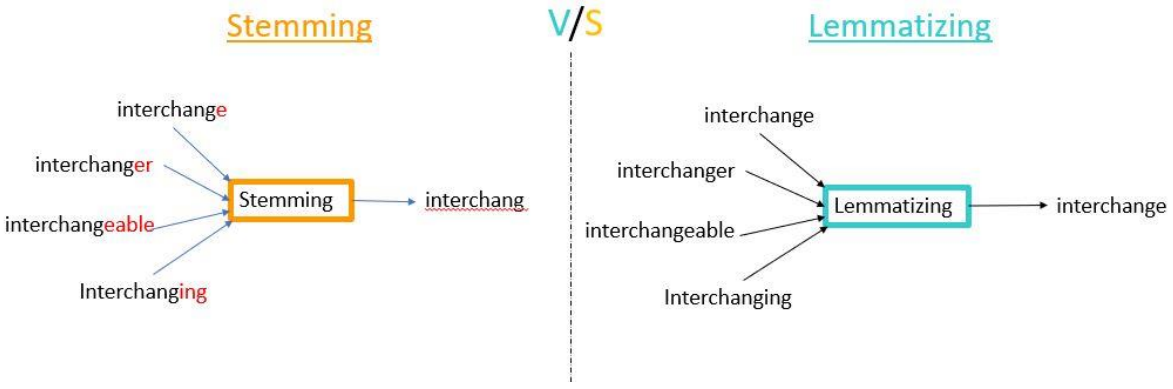


Рисунок 2.6 – Результативність лематизації на прикладі слова «interchange»

На рисунку 2.7 зображена діаграма процесу виконання лематизації

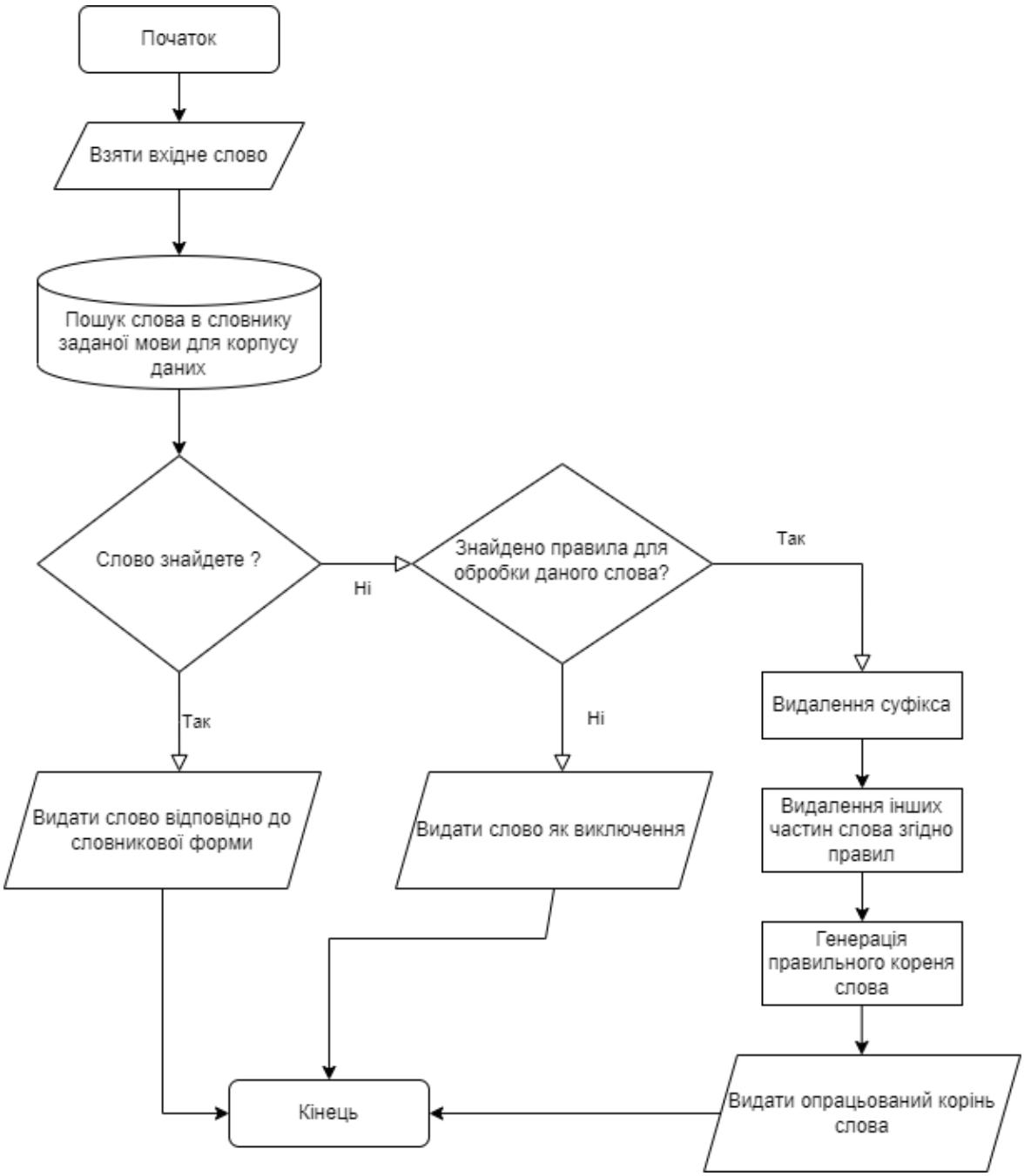


Рисунок 2.7 – Діаграма виконання лематизації

Як видно на рисунку вище, даний процес обробки даних звертається до джерела правил та слів. Оскільки лематизація передбачає отримання значення слова з чогось на зразок словника, це дуже трудомісткий процес. Тому більшість алгоритмів лематизації працюють повільніше порівняно з їхніми аналогами, що виводять слова. Крім того, лематизація пов'язана з обчислювальними витратами,

проте в задачах МН обчислювальні ресурси рідко стають причиною для занепокоєння.

2.5 Векторизація нормалізованих даних

Алгоритми машинного навчання працюють на числовому просторі ознак, очікуючи вхідні дані у вигляді двовимірного масиву, де рядки є екземплярами, а стовпці - ознаками. Для того, щоб застосовувати алгоритми машинного навчання для роботи з текстом, потрібно перетворити документи у векторне представлення, щоб можна було застосувати алгоритми дискретного обчислення. Цей процес називається вилученням ознак або, простіше кажучи, векторизацією.

Числове представлення документів надає можливість проводити повноцінну аналітику, а також створює екземпляри, на яких застосовуються алгоритми машинного навчання. У текстовому аналізі екземпляри - це цілі документи або фрагменти висловлювань, які можуть варіюватися за обсягом від цитат або твітів до цілих книг, але вектори яких завжди мають рівномірну довжину. Кожна характеристика векторного представлення є ознакою. Для тексту ознаки представляють атрибути та властивості документів, включаючи їх зміст. В цілому, атрибути документа описують багатовимірний простір ознак, до якого можна застосувати методи машинного навчання.

У машинному навчанні векторизація є кроком у вилученні ознак. Ідея полягає в тому, щоб виділити з тексту певні характерні риси, на яких може навчатися модель, шляхом перетворення тексту в числові вектори. Найбільш оптимальним методом векторизації для задач у даній роботі буде TF-IDF (від англ. TF — term frequency, IDF — inverse document frequency) [29]. Цей алгоритм є статистичним критерієм, який визначає, наскільки релевантним є слово в документі в сукупності інших документів. Для цього перемножуються два показники: кількість разів, коли слово зустрічається в документі, і величина, зворотна частоті цього слова в наборі документів. Якщо слово часто використовується в певному тексті, але рідко в інших, то воно має високу

релевантність до цього тексту. Отже, слова, які часто зустрічаються в кожному документі, такі як "цей", "той", "якщо", мають низький рейтинг, навіть якщо вони можуть зустрічатися багато разів, оскільки вони не є дуже важливими саме для цього документа. Однак, якщо слово "помилка" зустрічається багато разів в одному документі і не часто в інших, то, ймовірно, це означає, що воно є дуже важливим.

Заглиблюючись у деталі алгоритму TF-IDF, варто зазначити, що TF (Term Frequency) означає, як часто термін зустрічається в документі. Це відношення кількості згадувань слова до суми всіх слів документа, тобто частота слова зокрема:

$$TF = \frac{n_i}{\sum_k n_k}, \tag{2.1}$$

де n_i – число входжень слова в документ, $\sum_k n_k$ – загальна кількість слів в документі.

IDF (Inverse Document Frequency) - відношення загальної кількості документів до тих, в яких зустрічається задане слово. Зменшує вагу слова в залежності від його частоти і показує рівень релевантності тексту ключовому запиту:

$$IDF = log\frac{|D|}{|d_i \in t_i|}, \tag{2.2}$$

де $|D|$ - кількість документів в наборі, $|d_i \in t_i|$ - кількість документів, в яких зустрічається слово t_i (коли $n_i \neq 0$).

В результаті отримується значимість конкретного слова в межах одного документа:

$$TF - IDF(t, d, D) = TF(t, d) \cdot IDF(t, D), \tag{2.3}$$

Задля наглядності прикладної простоти застосування алгоритмічного методу TF-IDF, наведено зразок, зокрема: взято документ з 10000 символів в якому слово "інкапсуляція" зустрічається 25 разів, а колекція складається з 2 мільйонів документів, у 2000 з яких також зустрічається дане слово. Тоді: $TF = \frac{25}{10000} = 0.0025$; $IDF = lg\frac{2000}{2000000} = lg\frac{1}{1000} = -3(\lg - \text{логарифм за основою } 10)$; $TF - IDF = 0.0025 \cdot 3 = 0.075$.

Переваги методу TF-IDF:

– простота обчислень;

– містить базову метрику для вилучення найбільш описових термінів в документі;

– за допомогою базової метрики легко обчислюється схожість між двома документами.

Недоліки методу TF-IDF:

– TF-IDF базується на моделі bag-of-words (BoW), тому не фіксує позицію в тексті, семантику, наявність збігів у різних документах. Даний алгоритм вираховує зважений показник слова в документі, виходячи з його частоти;

– з наведених вище причин TF-IDF корисний лише як функція лексичного рівня. Цей алгоритм не може передати семантику (наприклад, у порівнянні з тематичними моделями).

2.6 Анотація оброблених даних

Анотування даних - це процес позначення окремих елементів навчальних даних щоб допомогти машинному алгоритму зрозуміти, які саме дані є важливими і що саме в них є важливим. Ці анотовані дані потім використовуються для навчання моделі. Анотування даних також відіграє важливу роль у більш широкому процесі контролю якості збору даних, оскільки добре анотовані набори даних стають наборами істинних даних: даними, які вважаються золотим стандартом і використовуються для вимірювання продуктивності моделі та якості інших наборів даних.

Завдання анотатора даних полягає в тому, щоб показати моделі машинного навчання, що саме слід передбачити. На рисунку 2.8 зображено діаграму процесу анотації даних. На практиці анотація даних - це процес транскрибування, тегування та маркування важливих ознак даних. Це ті ознаки, які система машинного навчання повинна розпізнавати самостійно, використовуючи реальні дані, які не були анотовані.

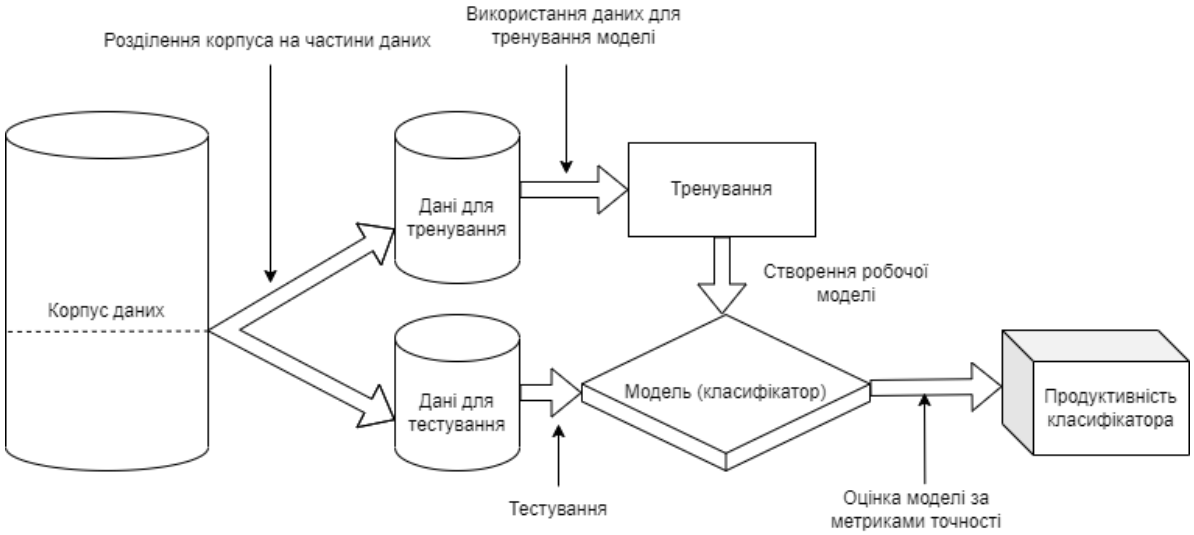


Рисунок 2.8 – Схема навчання та валідації алгоритму МН

Висновки до розділу 2

Було розроблено модель, яка виконує необхідні кроки опрацювання документів. Спочатку здійснюється сегментація тексту, тобто, процес, який розділяє довгі вирази тексту на менші шматки, або токени. Після виконується очищення даних, а саме, позбавлення малокорисних частин тексту завдяки видалення «stop- words» (слова-шуми). Далі отримані дані з попередніх кроків нормалізуються, зокрема, перетворюються форми слова, відповідаючи правилам лінгвістичних скорочень. На виході оброблені дані розподіляються на набори для тренування класифікаційних моделей та тестування їх.

Розділ 3 Розробка системи класифікації даних із використанням кластерного аналізу в якості агрегатора рішень моделей

3.1 Базові моделі класифікації. Аналіз та загальні положення

Система класифікації даних в даній роботі являє собою мета-алгоритм, який складається із наборів ансамблів, які, в свою чергу, містять в собі унікальні комбінації базових моделей. Основою даного ансамблевого підходу виступають алгоритми класифікації. Слід зазначити, що набір базових моделей не є константим, його можна замінити при необхідності подальшого дослідження.

В якості базових класифікаторів можна використовувати безліч популярних алгоритмів. В даному випадку, в якості базових моделей використовуються наступні алгоритми:

- метод опорних векторів (Support vector machine);
- наївний Баєсів класифікатор (Naive Bayes classifier);
- метод k-найближчих сусідів (K-neighbors);
- adaptive Boosting (AdaBoost);
- випадковий ліс (Random forest);
- логістична регресія (Logistic regression).

Машина опорних векторів (Support vector machine) - це алгоритм машинного навчання, який застосовується як для класифікації, так і для регресії [30]. Утім, здебільшого він використовується в задачах класифікації. В алгоритмі SVM кожен елемент даних трактується як точка в n-вимірному просторі (де n - кількість об'єктів, які є в наявності), значенням кожного об'єкта є значення певної координати. Потім виконується класифікація шляхом знаходження гіперплощини, яка добре розмежовує два класи, на рисунку 3.1 зображений приклад розмежування даних.

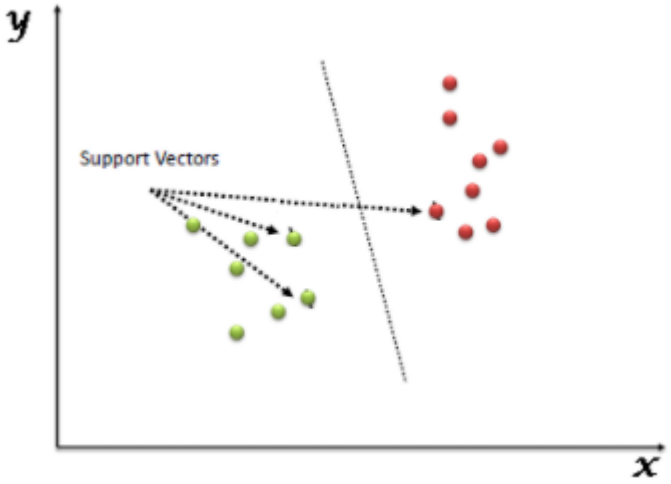


Рисунок 3.1 – Результат роботи SVM

Наївний байєсівський класифікатор (Naive Bayes classifier) - це класифікатор, базований на теоремі Байєса з припущенням про незалежність між предикторами [31]. Іншими словами, класифікація за Байєсом передбачає, що присутність певної ознаки в класі не впливає на наявність якої-небудь іншої ознаки в цьому класі. Наприклад, фруктом можна вважати яблуко, якщо воно червоного кольору, круглої форми і має приблизно 8 сантиметрів у діаметрі. Незважаючи на те, що ці ознаки можуть залежати одна від одної або від наявності інших ознак, всі ці властивості незалежно один від одного вносять свій внесок у ймовірність того, що цей фрукт є яблуком. Наївна модель Байєса проста в побудові і особливо ефективна для дуже великих масивів даних. Разом з простотою, цей підхід, як відомо, випереджає навіть дуже хитромудрі класифікаційні методи. Теорема Байєса забезпечує спосіб обчислення апостеріорної ймовірності $P\ (c \mid x)$ з $P\ (c)$, $P\ (x)$ та $P\ (x \mid c)$. Формула наведена нижче:

$$P\ (c \mid x)\ =\ \frac{P\ (x \mid c)P\ (c)}{P\ (x)}, \tag{3.1}$$

де $P\ (c \mid x)$ - це апостеріорна ймовірність класу (c, цілі), заданого предиктором (x, атрибути);

$P\ (c)$ - апріорна ймовірність класу;

$P\ (x \mid c)$- це ймовірність, яка є ймовірністю передбачувача даного класу;

$P\ (x)$ - апріорна ймовірність предиктора.

З формули (3.1) можна вивести наступну закономірність:

$$P\ (c \mid x) = \ P\ (x_1 \mid c)\ \times\ P\ (x_2 \mid c) \times \ldots \times\ P\ (x_n \mid c)\ \times\ P\ (c) \quad (3.2)$$

Метод k-найближчих сусідів (K-neighbors) визначає схожість між новим прикладом/даними та існуючими зразками, а потім відносить нові дані до категорії, яка є найбільш схожою з існуючими категоріями [32]. Алгоритм KNN зберігає набір даних на етапі навчання, а при появі нових даних відносить ці дані до категорії, релевантної вхідним даним.

Роботу K-NN можна пояснити на основі наведеного нижче алгоритму:

1. Визначається кількість K сусідів.

2. Вираховується евклідова відстань до K кількості сусідів;

3. Обирається K найближчих сусідів згідно з розрахованою евклідовою відстанню.

4. Серед даних k сусідів рахується кількість точок даних для кожної категорії.

5. Привласнюються нові точки даних тій категорії, для якої кількість сусідів максимальна.

Припустимо, що є нова точка даних, і необхідно віднести її до потрібної категорії, рисунок 3.2.

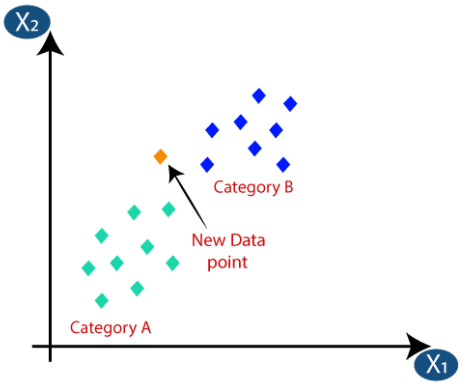


Рисунок 3.2 – Приклад даних

Спочатку обирається кількість сусідів k = 5. Далі визначається евклідова відстань між точками даних [36]. Знайшовши евклідову відстань, отримується K найближчих сусідів: 3 найближчих сусіда категорії «A» та два найближчих сусіда категорії «B», рисунок 3.3.

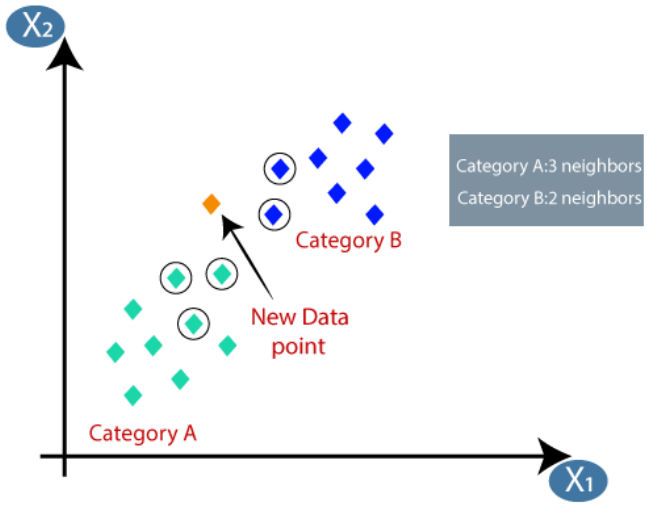


Рисунок 3.3 – Кількість найближчих сусідів нової точки

Як видно, 3 сусіди належать до категорії «A», 2 до категорії «B», отже, нова точка даних повинна належати до категорії «A».

Adaptive Boost (AdaBoost) - це один з бустерних ансамблів класифікаторів, розроблений Йоавом Фройндом та Робертом Шапіро у 1996 році [33]. Він об'єднує кілька алгоритмів для підвищення точності класифікації. AdaBoost є ансамблевим ітеративним методом. AdaBoost створюється шляхом об'єднання декількох неефективних класифікаторів, що створює ефективний класифікатор. Основна суть AdaBoost заключается у встановленні вагових коефіцієнтів для класифікаторів та підготовці вибірки даних в кожній ітерації так, щоб забезпечити точне передбачення нетипових вхідних даних.

В якості базового класифікатора може бути використаний будь-який алгоритм машинного навчання, якщо він приймає ваги на навчальній множині. Adaboost повинен відповідати двом умовам:

– навчання класифікатора повинно здійснюватися в інтерактивному режимі на різних навчальних прикладах з різними ваговими коефіцієнтами;

– навчання на вхідних даних повинно здійснюватися на кожній ітерації з метою мінімізації помилок навчання.

Алгоритм роботи класифікатора AdaBoost можна описати наступними кроками:

1. Спочатку Adaboost випадковим чином вибирає навчальну підмножину.

2. AdaBoost навчається ітеративно, обираючи навчальну множину на підставі даних про результати останнього тренування.

3. Неправильно відкласифікованим даним надається більша вага, щоб на наступній ітерації ці дані мали більшу вірогідність класифікації.

4. Крім того, вага навченого класифікатора присвоюється на кожній ітерації згідно з точністю класифікатора. Точніший класифікатор матиме вищі вагові коефіцієнти.

5. Цей процес виконується до тих пір, поки точність класифікації навчених моделей на навчальних даних не досягне максимальної точності, або поки не буде досягнута задана максимальна кількість ітерацій навчання.

Схема роботи алгоритму Adaptive Boosting зображена на рисунку 3.4.

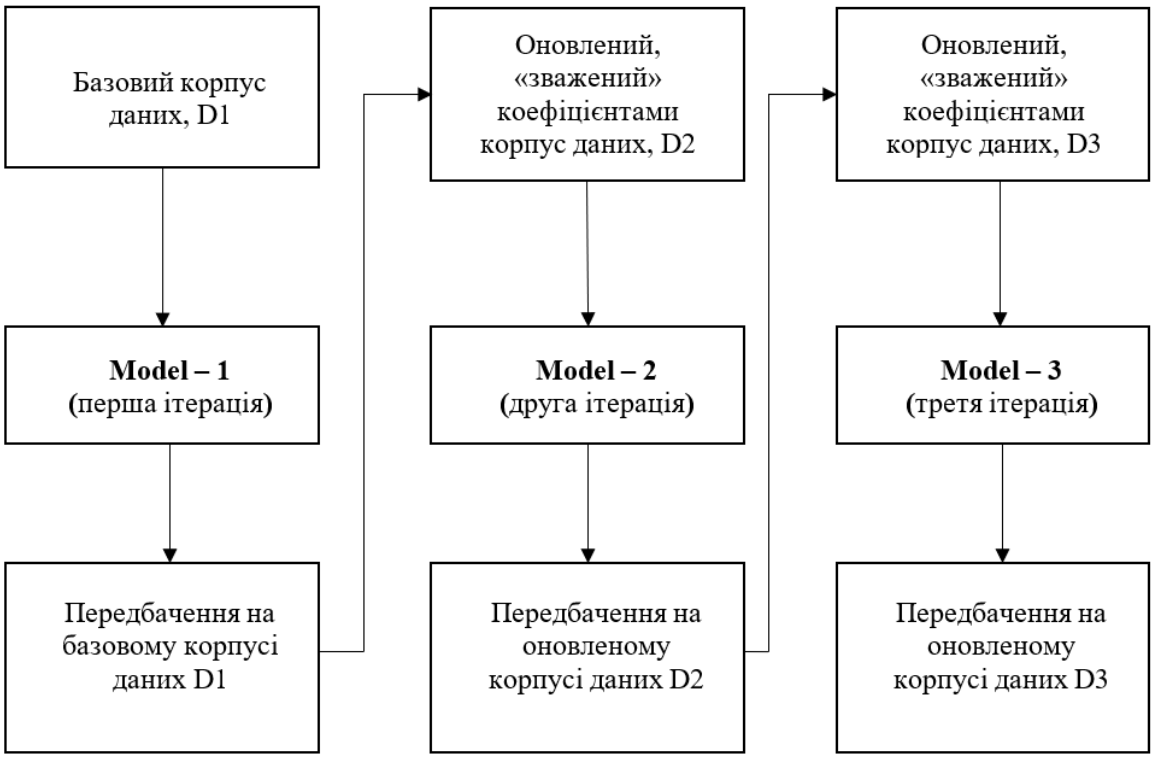


Рисунок 3.4 – Схема роботи класифікатора AdaBoost

Випадковий ліс (Random Forest) - це класифікатор, котрий складається з декількох дерев рішень для окремих підмножин заданого набору даних, беручи середнє рішення дерев для покращення точності прогнозування на даному наборі даних [34]. Метод базується на принципі ансамблевого навчання, який полягає в об'єднанні кількох класифікаторів для вирішення складної задачі та покращення продуктивності моделі. Random Forest отримує передбачення від кожного окремого дерева, а потім прогнозує кінцевий результат на підставі голосування за більшістю передбачень. Чим більша кількість дерев у лісі, тим вища точність.

Діаграма, що зображена на рисунку 3.5, показує принцип роботи класифікатора Random forest.

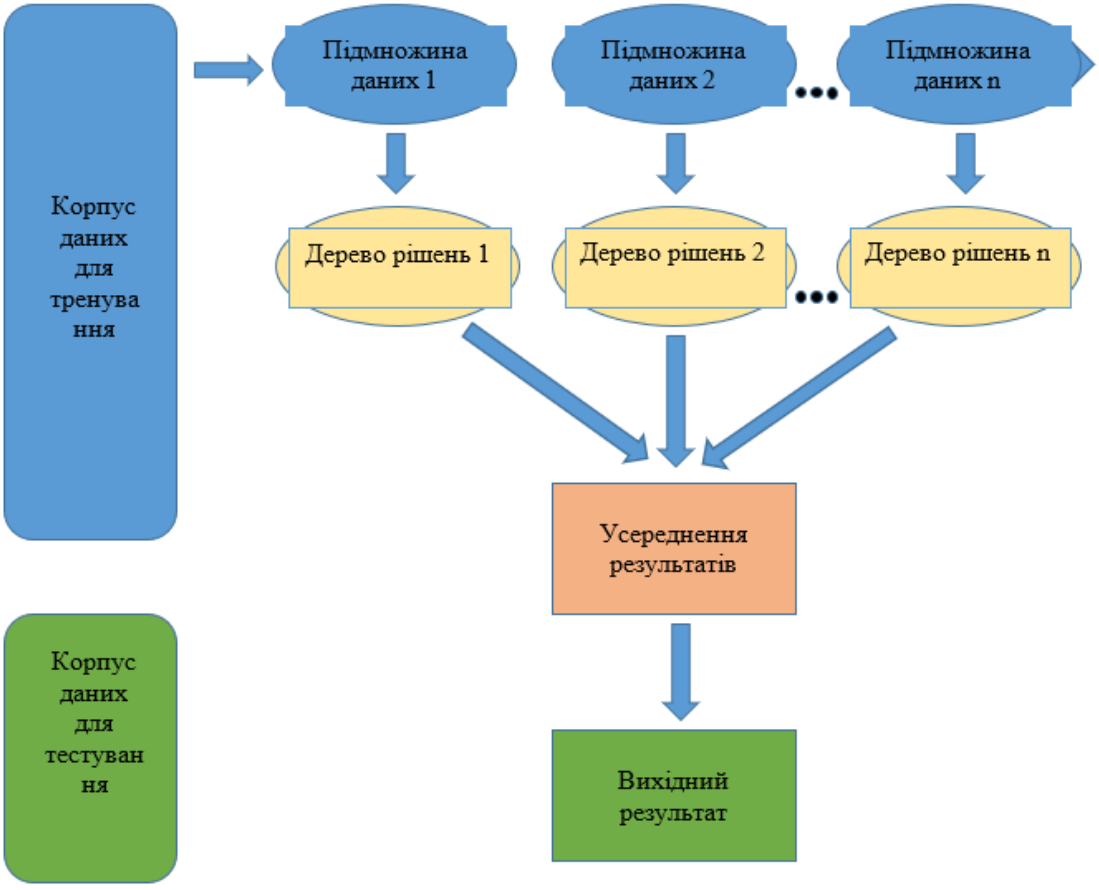


Рисунок 3.5 – Принцип роботи класифікатора Random forest

Так як випадковий ліс об'єднує декілька дерев для передбачення класу набору даних, цілком можливо, що окремі дерева рішень прогнозують правильний

результат, а інші - ні. Проте спільно всі дерева передбачають вірний результат. Тому нижче наведено два припущення щодо оптимального класифікатора:

– у наборі даних повинні бути деякі фактичні значення ознакової змінної, щоб класифікатор міг прогнозувати точні результати, а не наближені;

– прогнози кожного дерева повинні мати дуже низьку кореляцію.

Випадковий ліс функціонує в два етапи: спершу шляхом об'єднання N дерев рішень створюється випадковий ліс, потім для кожного дерева, створеного на першому етапі, робиться прогноз.

Робочий процес зазначеного алгоритму можна охарактеризувати такими етапами:

1. З навчальної вибірки вибираються випадкові значення K елементів даних.

2.Створюються дерева рішень, пов'язані з обраними даними (підмножини).

3. Вибирається кількість N дерев рішень, які необхідно створити;

4. Повторюються кроки 1 та 2.

5. Для нових даних виконуються прогнози кожного дерева рішень, після чого нові дані розподіляються в категорію, яка набрала найбільшу кількість голосів.

Логістична регресія вимірює зв'язок між залежною змінною (міткою, яка є предметом прогнозування) і однією або декількома незалежними величинами (властивостями навчального набору даних) шляхом оцінки ймовірностей за допомогою базової логістичної функції [35]. На виході цього класифікатора є ймовірності приналежності одиниці даних до того чи іншого кластеру, ці ймовірності необхідно перетворити в двійкові значення, щоб зробити певний прогноз. Цим завданням займається логістична функція, яку ще називають сигмоїдною функцією. Сигмоїдна функція - це S-подібна крива, яка дозволяє взяти будь-яке дійсне число і відобразити значення в діапазоні від 0 до 1. Ці значення між 0 і 1 потім будуть перетворені в 0 або 1 за пороговим показником.

На рисунку 3.6 зображені кроки, через які проходить алгоритм логістичної регресії, щоб видати результат класифікації.

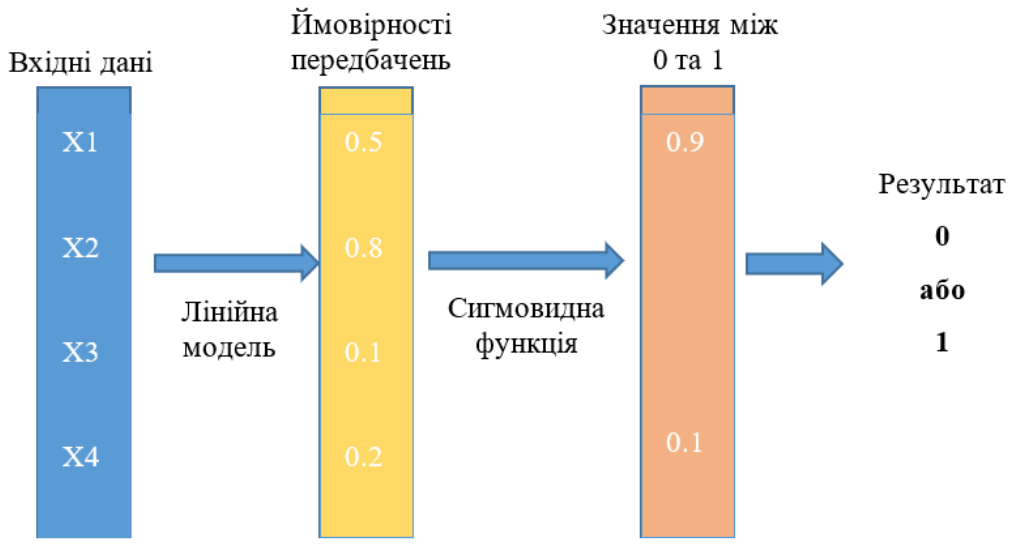


Рисунок 3.6 – Кроки, через які проходить алгоритм логістичної регресії, щоб видати результат класифікації

Вище згадані базові класифікатори слугують окремим модулем в розроблюваній системі, даний модуль буде використовуватись щоб порівнювати ефективність запропонованого методу на осново групування (агрегації) рішень моделей ансамблю. Після обробки вхідних текстових даних, базові моделі роблять свої прогнози, ці прогнози оцінюються згідно основних метрик МН, далі настає черга оцінки мета-алгоритму, який складається з ансамблів комбінацій базових класифікаторів. Після оцінювання продуктивності розробленого підходу, показники результатів базових алгоритмів та мета-алгоритму порівнюються, тобто, відбувається валідація запропонованого підходу.

Не зважаючи на те, що вище були описані базові класифікатори, які використовуються в поточному методі застосування агрегативних підходів для класифікації на базі ансамблевих моделей, застосунок реалізований так, щоб набір базових моделей міг бути змінюваний, тобто алгоритми, які використовуються для класифікації тексту можуть бути змінені на інші, або до базового набору алгоритмів класифікації уможливлено додавання нових. Опція заміни, або додавання нових «примітивів» в ансамбль дозволяє проводити експерименти, ціль

який полягає в тому, щоб знайти найбільш оптимальний набір базових моделей, при якому точність всього ансамблю (мета алгоритму) буде найвища.

На рисунку 3.7 зображена схема роботи модуля, який виконує класифікацію без застосування колективних рішень ансамблю.

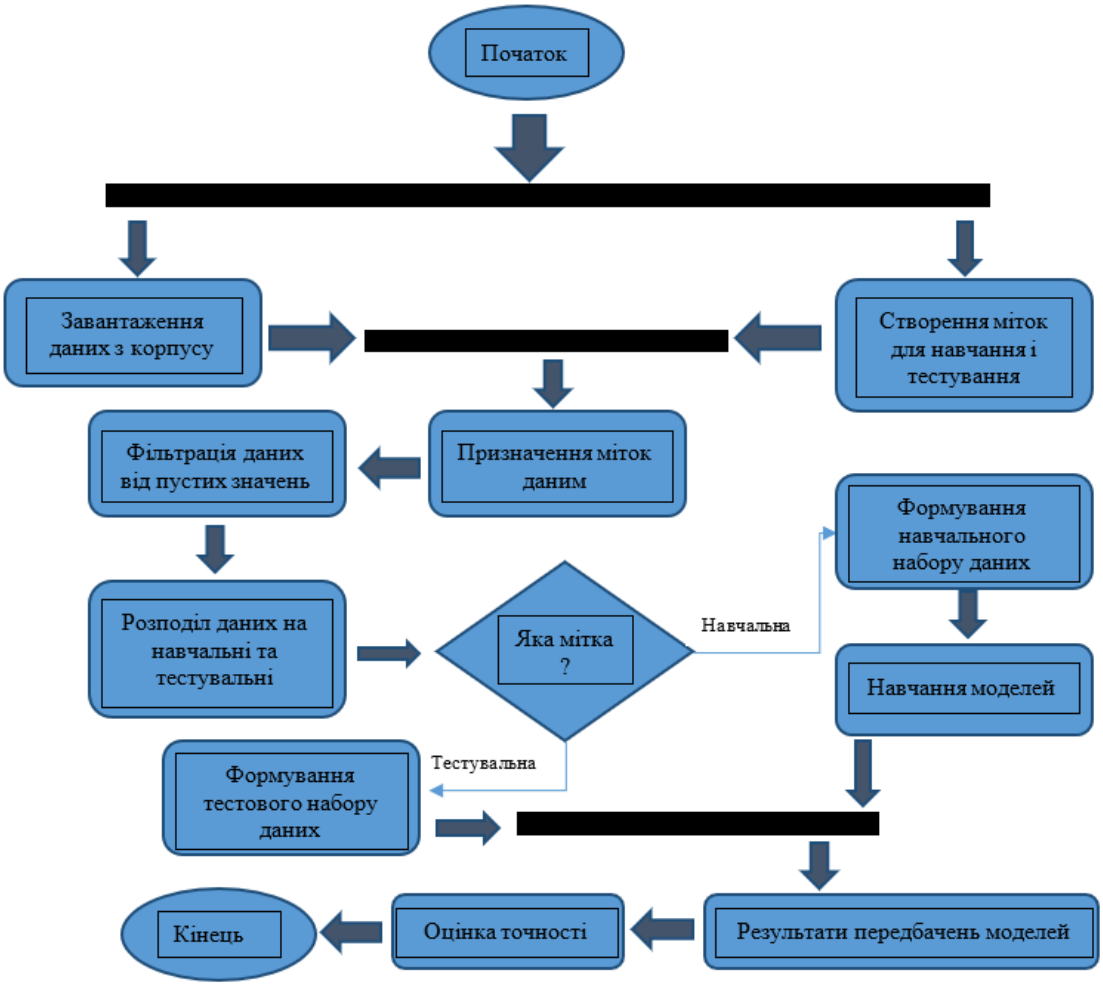


Рисунок 3.7 – Узагальнена схема роботи модуля базових класифікаторів

3.2 Модель формування ансамблю прийняття рішень

Після вибору стеку базових алгоритмів класифікації, необхідно сформувати ансамблевий підхід, за допомогою якого система зможе виконувати поставлені завдання щодо оптимізації результативності. Суть завдання не полягає в тому, щоб просто створити один ансамбль із набору примітивів, стоїть задача

створення набору таких ансамблів, які будуть містити в собі тільки унікальні комбінації базових моделей. Методами комбінаторики потрібно створити набори унікальних ансамблів, щоб в подальшому вибрати найкращий ансамбль із створеного набору.

Функція, яка задовільняє поставлену комбінаторну задачу, наведена на рисунку 3.8 та реалізована на мові програмування Python [36]. Слід зазначити, що інформаційна система класифікації також реалізована на попередньо зазначеній мові програмування, оскільки дана «платформа» містить в собі багато рішень, пов'язаних із роботою МН що, в свою чергу, полегшує процес проведення дослідження, адже не потрібно відволікатись на реалізацію примітивів при написанні мета-алгоритму.

```
def y_function(x):
    y = 0
    for i in range(0, x):
        for j in range(i + 1, x):
            for k in range(j, x):
                y += 1
    return y
```

Рисунок 3.8 - Функція залежності кількості примітивів (x) до кількості унікальних комбінацій (y)

За допомогою вище зображеного алгоритму створюються унікальні набори комбінацій. Схема створення ансамблів, які містять унікальні набори алгоритмічних примітивів класифікації, зображена на рисунку 3.9. Графік залежності кількості базових моделей класифікації до кількості створених унікальних ансамблів зображений на рисунку 3.10.

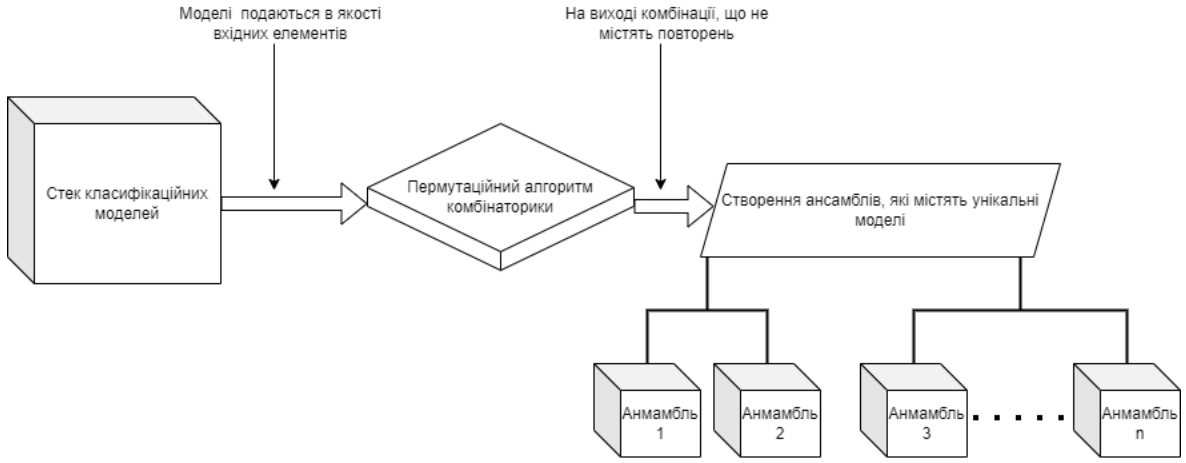


Рисунок 3.9 – Підхід до створення наборів ансамблів комбінаторним методом

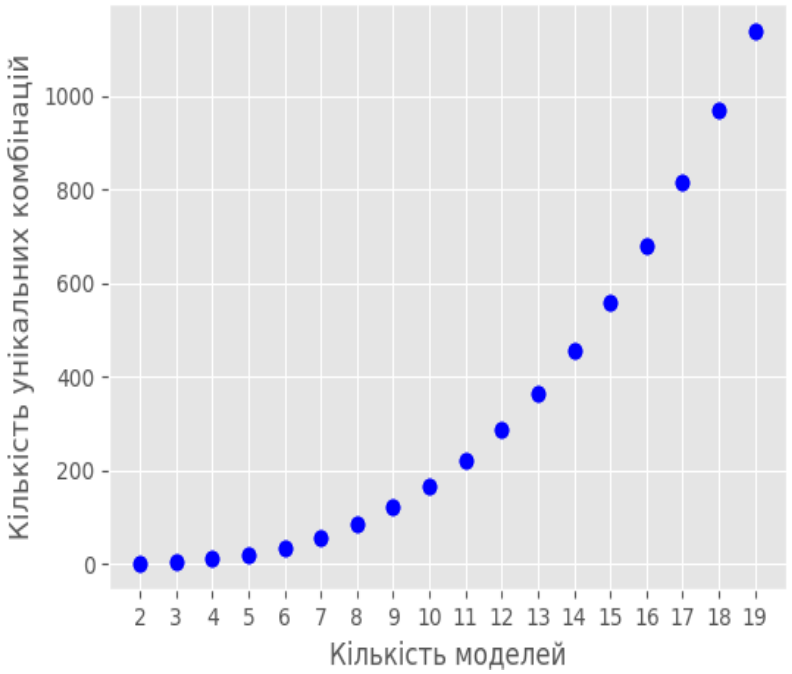


Рисунок 3.10 – Залежність кількості базових моделей класифікації до кількості створених унікальних ансамблів

Даний графік дозволяє орієнтуватись у складності системи, адже дивлячись на нього, можна обрати оптимальну з точки зору навантаження на систему кількість базових класифікаторів.

Виходячи із вище зазначеного матеріалу, задача складання мета-алгоритму, який містить в собі набори унікальних ансамблів, вважається

виконаною. Наступним кроком буде реалізація алгоритму усереднення (оптимізації) передбачень базових моделей у кожному унікальному ансамблі.

3.3 Метод виведення колективного рішення на основі групування передбачень моделей ансамблю

Після успішного конструювання мета-алгоритму потрібно вирішити наступне завдання, а саме, потрібно узагальнити результати базових класифікаторів у кожному ансамблі, який складається з унікального набору моделей класифікації.

Перед тим як виконувати оптимізацію рішення ансамблю слід зазначити принцип по якому буде відбуватись усереднення передбачень. Особливістю у задачі усереднення передбачень базових класифікаторів є те, що потрібно агрегувати рішення базових класифікаторів у кожному ансамблі тільки на тих порціях даних, де моделі не зійшлись у передбаченнях, тобто, агрегуються результати передбачень кожного алгоритму в ансамблі на даних, передбачення над якими хоча б у однієї моделі з комбінації були відмінні від інших передбачень у даному наборі.

Під час створення колективного рішення для кожного ансамблю моделей використовується кластериний аналіз даних. Кластерний аналіз - це статистичний метод обробки даних, який використовується з матрицями даних, в яких змінні не були попередньо розбиті на підмножини критеріїв та предикторів. Припускається, що в основі даних лежить невпорядкований набір дискретних класів. Всі вони різні, і жоден з них не має більшої ваги, ніж інший. Кластерний аналіз полягає в тому, що елементи об'єднуються в групи, або кластери, на основі того, наскільки тісно вони пов'язані між собою. Його можна окреслити як завдання визначення підгруп даних таким чином, щоб точки даних в одній підгрупі (кластері) були схожі, в той час як точки даних в різних кластерах суттєво відрізнялися.

Процедури кластеризації можна розглядати як "пре-класифікаційні" в тому сенсі, що дослідник не використовував попередні судження для розбиття об'єктів.

Однак вважається, що існують неоднорідні групи даних, тобто існують "кластери".

Виконувати агрегацію рішень базових моделей у ансамблі буде алгоритм k-середніх (k-means) [37]. Алгоритм кластеризації k-середніх обчислює центроїди та ітераційно повторюється до тих пір, поки не знайде оптимальний центроїд. Передбачається, що кількість кластерів вже відома. Його також називають алгоритмом плоскої кластеризації. Кількість кластерів, визначених з даних за допомогою алгоритму, позначається «K».У цьому алгоритмі точки даних відносяться до кластеру таким чином, щоб сума квадратів відстаней між точками даних та центроїдом була мінімальною.

Ключовими даними у вище згаданому алгоритмі кластеризації є критерії відстані. Вхідними даними для алгоритму k-means є матриця, що складається з відстаней між кожним об’єктом. Щоб визначити відстані між об’єктами, потрібно мати міру відстаней.

У знаходженні відстані між об’єктами допоможе Евклідова відстань [38]. Це геометрична відстань в багатовимірному просторі. Якщо об’єкти визначаються багатовимірними точками, або мають багато характеристик (стимулів), $X_i = \{x_{i1}, x_{i2}, x_{i3}, \dots, x_{in}\}$, де $i = 1 \dots n$, відстань може бути визначена відстанню між точками $d(X_i, X_j)$, де

$$d(X_i, X_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2} . \qquad (3.3)$$

Метод базується на мінімізації суми квадратів відстаней між кожною точкою та центром її кластера, тобто функції:

$$\sum_{i=1}^{N} d\left(x_i, m_j(x_i)\right)^2 , \qquad (3.4)$$

де d – метрика, x_i - і-тий об’єкт даних, а $m_j(x_i)$ – центр кластера, якому на j-тій операції присвоєний елемент x_i .

Принцип алгоритму полягає в пошуку таких центрів кластерів та наборів елементів кожного кластера при наявності деякої функції Ф(°), що виражає якість поточного розбиття множини на k кластерів, коли сумарне квадратичне відхилення елементів кластерів від центрів цих кластерів буде найменшим:

$$V = arg\min \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2, \quad (3.5)$$

де k – число кластерів, S_i - отримані кластери, $i = 1,2,\dots,k$, μ_i – центри мас векторів $x_j \in S_i$.

В межах цього дослідження вище зазначений алгоритм буде виконувати розбиття вхідних даних на два кластери і працювати наступним чином:

– на вхід подаються ймовірності передбачень базових моделей. Оскільки виконується бінарна класифікація, відповідно, ймовірності можуть бути додатними та від'ємними;

– вираховуються значення центроїдів кожного кластера, у даному випадку їх два (бінарне розбиття);

– відбувається обчислення середнього значення отриманих на попередньому кроці центроїдів;

– в залежності від того, чи середнє значення додатне, чи від'ємне, усереднене рішення моделей на даній порції даних буде наступне: відноситься порція даних до заданої теми, чи ні.

Процес виконання алгоритму k-means в якості усереднювача (агрегатора) передбачень базових моделей у ансамблі можна розгледіти поетапно на блок-схемі, зображеній на рисунку 3.11. Зупинка алгоритму проводиться тоді, коли границі кластерів і розташування центроїдів не перестануть змінюватися від ітерації до ітерації, тобто на кожній ітерації в кожному кластері буде залишатися один і той же набір об'єктів. На практиці алгоритм зазвичай знаходить набір стабільних кластерів за кілька десятків ітерацій.

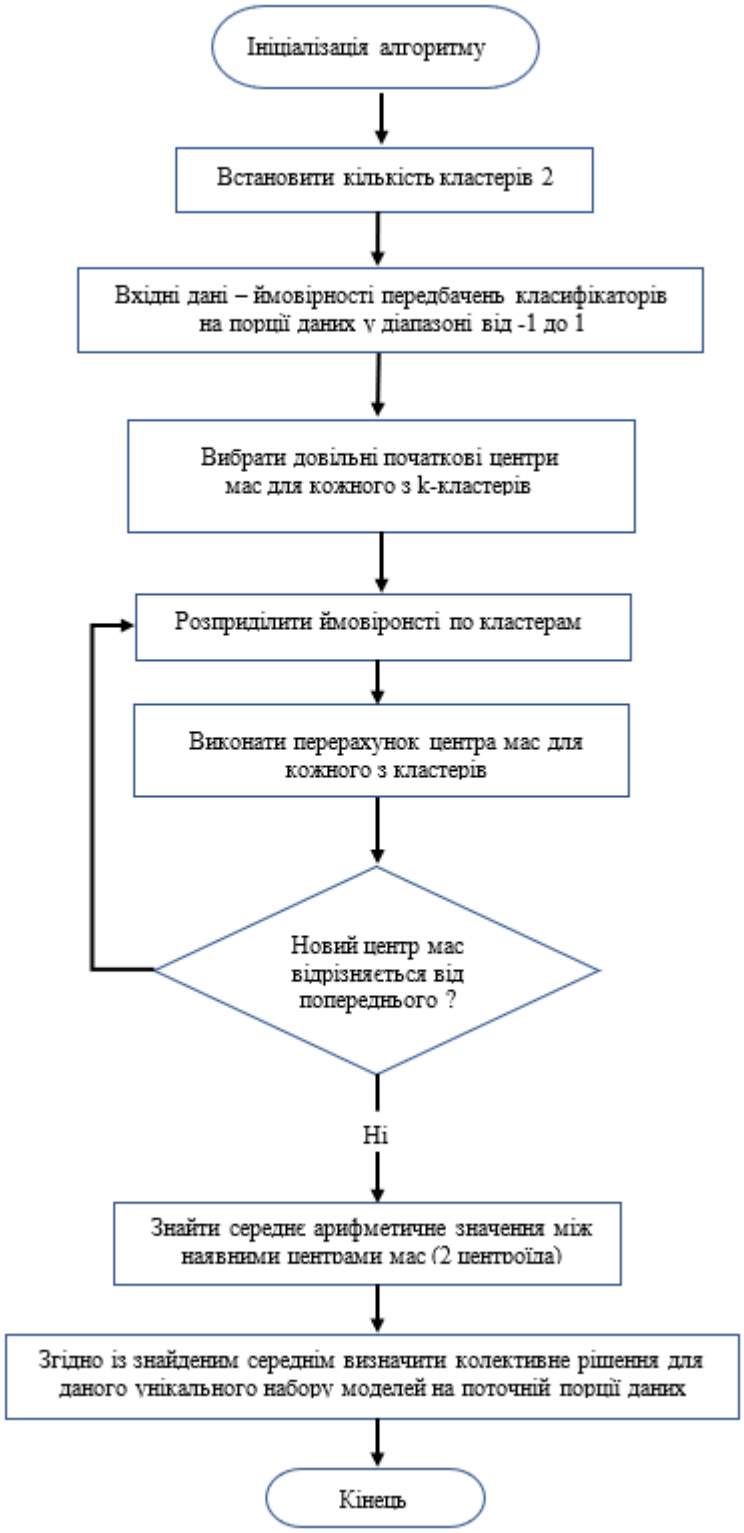


Рисунок 3.11 – Блок-схема алгоритму усереднення результатів на основі k-середніх (k-means)

Задля більш наочного представлення роботи даного методу виведення колективного рішення, на рисунку 3.12 зображений приклад кластеризації даних результатів базових моделей . Вхідні дані на початку алгоритму являють собою

ймовірності передбачень кожної моделі на порціях даних, де результати класифікації відрізнялись хоча б у однієї моделі у даному унікальному ансамблі. Порції даних, на яких результат класифікації алгоритмічних примітивів був однаковий, ігноруються.

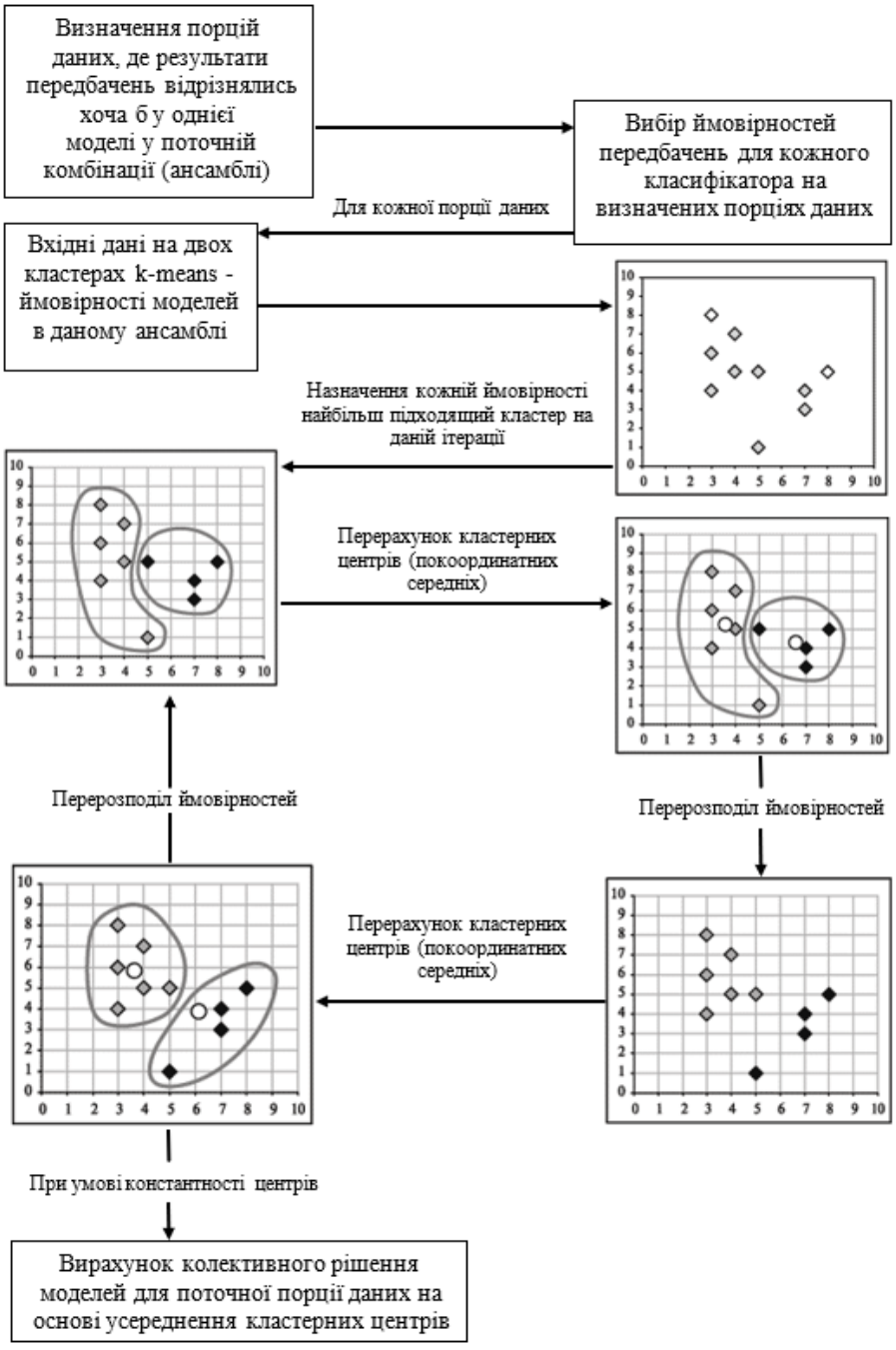


Рисунок 3.12 – Приклад роботи алгоритму виведення колективного рішення

Таким чином, оптимізувавши рішення базових моделей у кожному унікальному ансамблі на тих порціях даних, передбачення над якими було розбіжне хоча б у одного примітивного класифікатора, можна приступити до вирахування кореляційних зв’язків рішень моделей та істинних значень корпусу даних.

3.4 Розрахунок кореляційних зв’язків моделей

Однією з основних цілей дослідження є візуалізація отриманих показників. Основні метрики для тієї чи іншої класифікаційної області повинні бути візуально представлені у вигляді результуючих показників. Для того, щоб відобразити всі ці метрики, було додано модуль для побудови графіків у дво- та тривимірному просторі. Цей модуль також використовується для відображення коефіцієнтів кореляції між корпусом тестових даних та прогнозами базових моделей.

Коефіцієнт кореляції є статистичною мірою сили зв'язку між взаємними залежностями двох перемінних. Діапазон значень коливається від -1,0 до 1,0. Обчислене число більше 1,0 або менше -1,0 означає, що при обчисленні кореляції була допущена помилка. Кореляція -1,0 свідчить про наявність ідеальної негативної кореляції, а кореляція 1,0 - про наявність ідеальної позитивної кореляції. Кореляція 0,0 вказує на відсутність лінійного зв'язку між динамікою двох змінних.

Негативна кореляція - це такий зв'язок між двома змінними, при якому одна змінна збільшується, коли інша зменшується, і навпаки.

Позитивна кореляція - це зв'язок між двома змінними, при якому обидві величини рухаються в тандемі - іншими словами, в одному і тому ж напрямку. Можна стверджувати, що за позитивної кореляції одна змінна зменшується, якщо зменшується інша змінна, або одна змінна збільшується, якщо збільшується інша змінна.

Сила зв'язку варіюється в залежності від значення коефіцієнта кореляції. Приміром, значення 0,2 вказує на те, що між двома змінними існує позитивний

зв'язок, але він є слабким і, ймовірно, не є важливим. Аналітики в певних наукових галузях не розглядають кореляцію як важливу доти, доки її значення не перевищує принаймні 0,8. Проте коефіцієнт кореляції з абсолютним показником 0,9 і більше буде свідчити про дуже сильний зв'язок.

Оскільки алгоритм вирахування кореляційного коефіцієнта приймає на вхід 2 множини даних для порівняння, в даній роботі одна з множин являє собою масив передбачень класифікаційної моделі, інша множина представляється істинними значеннями відповідних наборів документів, щодо яких класифікатор робив свої передбачення.

Застосовується кореляція за Пірсоном [39]. Коефіцієнт кореляції Пірсона між двома змінними дорівнює коваріації двох змінних, або сумі добутків відхилень, поділеній на добуток їх стандартних відхилень. Нехай, є дві вибірки $x^m = (x_1, \cdots, x_m),\ y^m = (y_1, \cdots, y_m)$. Для даних вибірок коефіцієнт кореляції Пірсона розраховують за формулою:

$$r_{xy} = \frac{\sum_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{m}(x_i - \bar{x})^2 \sum_{i=1}^{m}(y_i - \bar{y})^2}} = \frac{\operatorname{cov}(x,y)}{\sqrt{s_x^2 s_y^2}}, \quad (3.6)$$

Де $\bar{x},\ \bar{y}$ – вибіркові середні x^m і y^m, s_x^2, s_y^2 - вибіркові дисперсії, $r_{xy} \in [-1,1]$.

Важливо зауважити, що коефіцієнти кореляцій у двох, трьох-вимірному просторі будуються для обох випадків: статичний, динамічний. Коефіцієнт кореляції в статиці обраховується відносно сталого набору документів для кожного класифікатора, а от динамічний коефіцієнт кореляції обраховується на тих документах, які не входять до «ядра» передбачень кожного набору моделей. Ядром передбачень вважається така множина документів, на яких була дана однакова відповідь класифікаторів у наборі. Отож, «ядро» передбачень змінюється відносно кожного універсального набору класифікаторів.

Враховуючи вищевикладене, коефіцієнт кореляції є необхідною метрикою, яка показує, наскільки "хорошою" є модель, що лежить в основі

класифікатора. Перспективність використання класифікатора визначається силою кореляційного зв'язку між його прогнозами та правильними відповідями, зокрема, чим вищий коефіцієнт кореляції, тим точнішим є класифікатор.

3.5 Реалізація інформаційної системи класифікації текстових даних

3.5.1 Структура і функціональне призначення модулів системи

Під час реалізації програмного засобу, було реалізовано структуру, яка відповідає поставленому завданню. Діаграма класів зображена на рисунку 3.13.

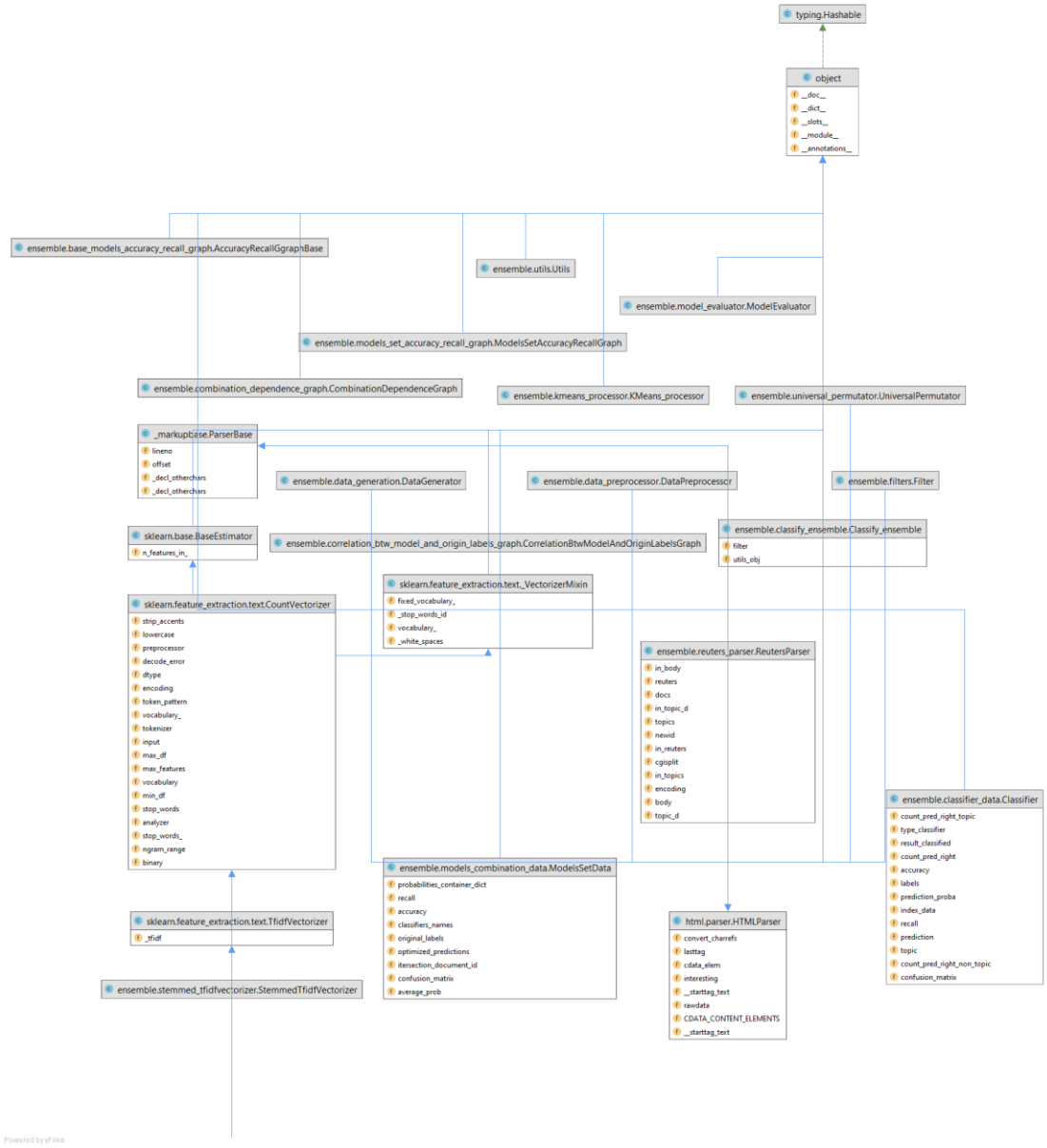


Рисунок 3.13 – Діаграма класів

Клас «ReutersParser», який наслідується від класу «HTMLParser», дозволяє завантажувати дані з корпусу даних (файлів) і розбивати їх на окремі сегменти. Це зроблено для того, щоб вибрати тільки ті документи, в яких є необхідні блоки тексту.

Класи «DataGenetator», «DataPreprocessor» відфільтровують завантажені дані, розбивають вхідні дані на вибірки для навчання та тестування моделей, навчають «vectorizer», «tf-idftransformer».

Клас «Filter» розмічає вхідні дані по темах, та видаляє пусті документи.

Клас «ClassifyEnsemble» являється одним із найбільших в програмному продукті, від відповідіє за навчання моделей, тестування моделей.

Клас «KMeans_processor» використовується в якості модуля кластеризації передбачень базових моделей. Саме в цьому класі відбувається оптимізація передбачень класифікаторів.

Клас «ModelEvaluator» призначений для оцінювання основних метрик класифікаторів, таких як точність, повнота, матриця невідповідностей.

Класи, які являються контейнерами для даних окремих класифікаторів, або контейнерами для даних комбінації моделей: «ClassifierData», «Model», «ModelsCombinationData».

За створення графіків залежності комбінацій від кількості базових моделей, точності базових моделей, точності комбінацій, та графіків, які показують коефікаєнти кореляції у двох, трьох-вимірному просторі, відповідають наступні класи: «AccuracyRecallGgraphBase», «CombinationDependenceGraph», «CorrelationBtwModelAndOriginLabelsGraph», «ModelsSetAccuracyRecallGraph».

Вище згадані компоненти системи працюють в тандемі, для того щоб мета алгоритм, який опрацьовує документи текстових даних, міг створювати ансамблі моделей, оцінювати їх, виводити графічні показники продуктивності кожного набору та візуалізувати кореляційні показники рішень моделей.

Узагальнена діаграма виконання системи зображена на рисунку 3.14.

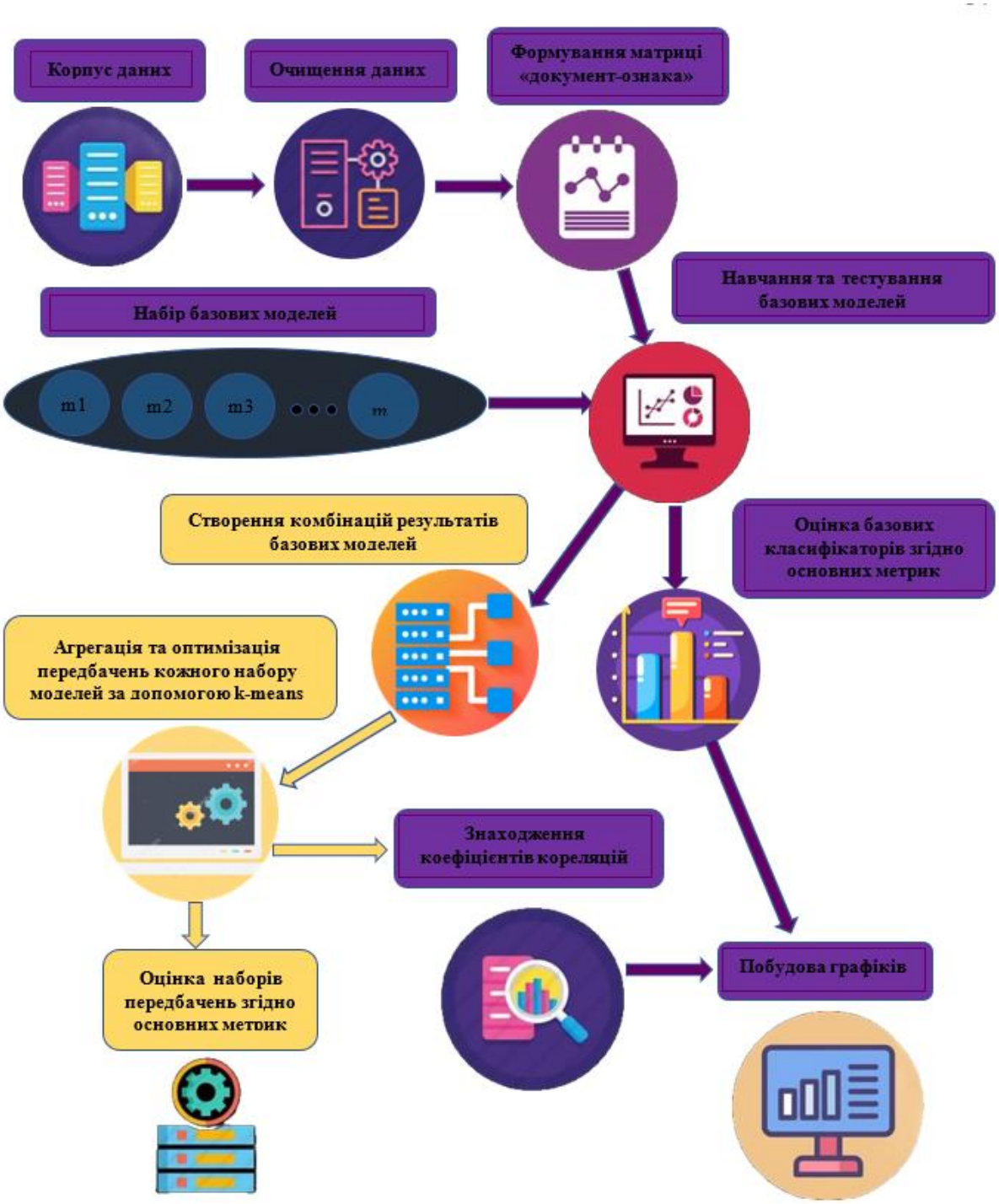


Рисунок 3.14 – Узагальнена діаграма кроків виконання системи

3.5.2 Тестування інформаційної системи

Щоб перевірити коректність роботи застосунку, здійснено модульне тестування його роботи. Модульне тестування - це різновид тестування програмного забезпечення, при якому тестуються окремі модулі або складові частини програми. Метою є перевірка правильності роботи кожної одиниці

програмного коду. Виконується модульне тестування розробниками під час створення (етапу кодування) програми. При проведенні модульних тестів ізолюється ділянка коду і перевіряється його коректність. Одиницею може бути окрема функція, метод, процедура, модуль або об'єкт.

Причинами проведення тестування системи слугують наступні:

– модульні тести допомагають виправити помилки на початку циклу розробки та заощадити витрати часу;

– це допомагає зрозуміти базу коду тестування та дозволяє швидко вносити зміни;

– хороші модульні тести служать проектною документацією;

– модульні тести допомагають з повторним використанням коду. Можна перенести код і тести в новий проект, після чого Змінювати код, поки тести не запустяться знову.

Інформаційна система класифікації даних на базі ансамблевий моделей включає в себе чимало модулів, для основних з яких були зроблені наступні модульні тести:

– тест модуля завантаження даних;

– тест фільтрування даних;

– тест навчання базових класифікаторів;

– тест дампу навчених моделей;

– тест завантаження дампу;

– тест модуля тестування навчених класифікаторів;

– тест модуля створення комбінацій;

– тест модуля оцінки наборів класифікаторів;

– тест функцій побудови двох, трьох-вимірних графіків (декілька різних тестів).

Окрім модульних тестів, система також підтримує логування (logging) для зручного відслідковування процесу виконання. Результат зображений на рисунках 3.15, 3.16.

```
With remove stopwords and stemming
Teaching of classifiers starts

---------- Labels  earn non-earn ----------
Using svm Classifier for teach

Training with svm...
Training time 80.95 seconds.
Using naive-bayes Classifier for teach

Training with naive-bayes...
Training time 0.03 seconds.
Using k-neighbors Classifier for teach

Training with k-neighbors...
Training time 0.02 seconds.
Using ada-boost Classifier for teach

Training with ada-boost...
Training time 28.53 seconds.
Using random-forest Classifier for teach
```

Рисунок 3.15 – Фрагмент логування під час проходження навчання моделей

```
Teaching of classifiers takes 283.77 seconds.

Testing of classifiers starts

---------- Labels  earn non-earn ----------
Using svm Classifier for test

Testing time of svm takes 7.26 seconds.
Saved to file...
Using naive-bayes Classifier for test

Testing time of naive-bayes takes 2.51 seconds.
Saved to file...
Using k-neighbors Classifier for test

Testing time of k-neighbors takes 4.06 seconds.
Saved to file...
Using ada-boost Classifier for test

Testing time of ada-boost takes 2.39 seconds.
Saved to file...
Using random-forest Classifier for test
```

Рисунок 3.16 – Фрагмент логування процесу тестування

Результат проходження модульних тестів зображений на рисунку 3.17.

```
>>> ____Start__Testing____
... test_data_generation (__main__.Unit_Testing) ... ok
... test_filters (__main__.Unit_Testing) ... ok
... test_teach_classifiers (__main__.Unit_Testing) ... ok
... test_dump_teached_data_save (__main__.Unit_Testing) ... ok
... test_dump_teached_data_load (__main__.Unit_Testing) ... ok
... test_test_classifiers (__main__.Unit_Testing) ... ok
... test_permutate_universally (__main__.Unit_Testing) ... ok
... test_evaluate_by_important_metrics (__main__.Unit_Testing) ... ok
... test_build_static_correlation_graphs2d (__main__.Unit_Testing) ... ok
... test_build_dynamic_correlation_graphs2d (__main__.Unit_Testing) ... ok
... test_build_dynamic_correlation_graphs3d (__main__.Unit_Testing) ... ok
... test_build_static_correlation_graphs3d (__main__.Unit_Testing) ... ok
... test_build_accuracy_recall_graph (__main__.Unit_Testing) ... ok
... test_build_combination_dependence_graph (__main__.Unit_Testing) ... ok
... ----------------------------------------------------------------------
... Ran 14 tests in 38 seconds
...
... OK
```

Рисунок 3.17 – Результати проходження модульних тестів

Висновки до розділу 3

На даному етапі розроблено модуль, який використовує базові моделі класифікації задля отримання показників ефективності рішень. Розроблено модель складання мета-алгоритму із використанням наборів унікальних комбінацій базових алгоритмів класифікації. Даний мета-алгоритм являє собою набори ансамблів, задля оптимізації рішень всередині кожного ансамблю використовується запропонований метод групування рішень моделі на основі неієрархічного алгоритму кластеризації. У висновку передбачення класифікаторів порівнюються з істинними значеннями, при цьому вираховуються їхні кореляційні зв'язки рішень на відповідних порціях даних. Даний підхід уможливлює знаходження найпродуктивніших моделей з точки зору МН.

Розділ 4 Дослідження ефективності застосування методу класифікації даних на основі групування рішень моделей ансамблю

4.1 Результати базових моделей

В якості оцінювальних метрик використовуються точність «accuracy», повнота «recall», та матриця невідповідностей «confusion matrics».

«confusion matrics» являє собою матрицю розміром 2 на 2. На рисунку 4.1 зображена структура цієї матриці.

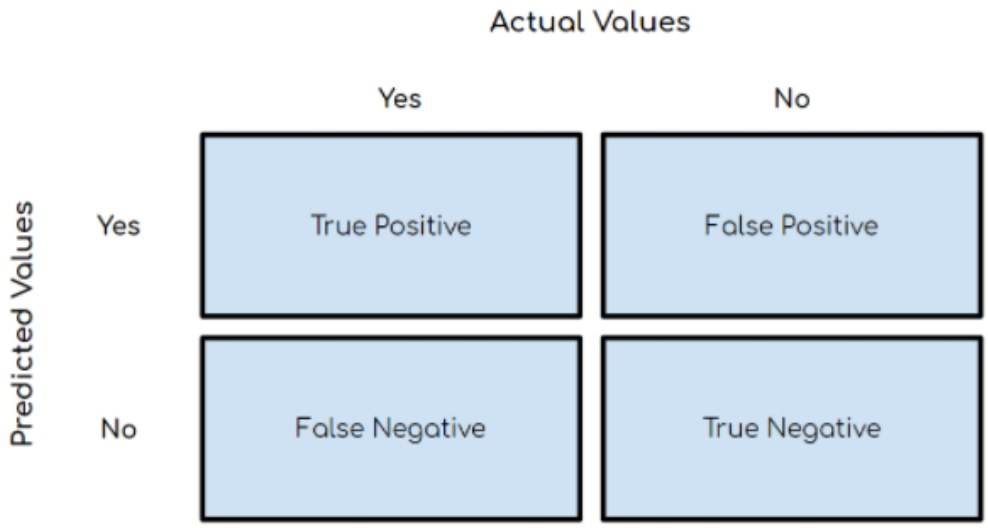


Рисунок 4.1 – Структура матриці невідповідностей

У схемі, зображеній на рисунку 4.1 міститься інформація скільки разів система прийняла вірне і скільки разів невірне рішення за документами заданого класу. А саме:

- TP – істинно позитивне рішення;
- TN – істинно негативне рішення;
- FP – хибно позитивне рішення;
- FN – хибно негативне рішення.

Найбільш інтуїтивно зрозумілою, очевидною є «accuracy» - частка правильних відповідей алгоритму:

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (4.1)$$

Наприклад, треба оцінити роботу спам-фільтра пошти. Отже, отримано 100 не-спам листів, 90 з яких класифікатор визначив правильно (True Negative = 90, False Positive = 10), і 10 спам-листів, 5 з яких класифікатор також визначив правильно (True Positive = 5, False Negative = 5).

Тоді точність буде наступна:

$$accuracy = \frac{5+90}{5+90+10+5} = 86{,}4 \quad (4.2)$$

Проте точність не є абсолютно об’єктивним показником. Вона сама по собі не дає повної картини, якщо працювати з незбалансованим за класами набором даних, таким набором, де існує значна диспропорція між кількістю позитивних і негативних маркувань.

«recall» показує, яку частку об'єктів позитивного класу з усіх об'єктів позитивного класу знайшов алгоритм.

$$recall = \frac{TP}{TP+FN} \quad (4.3)$$

На рисунку 4.2 зображений графік точності та повноти, побудований на основі оцінювання базових класифікаторів. В даному випадку – це «svm», «naive-bayes», «k-neighbors», «ada-boost», «random-forest», «logistic-regression».

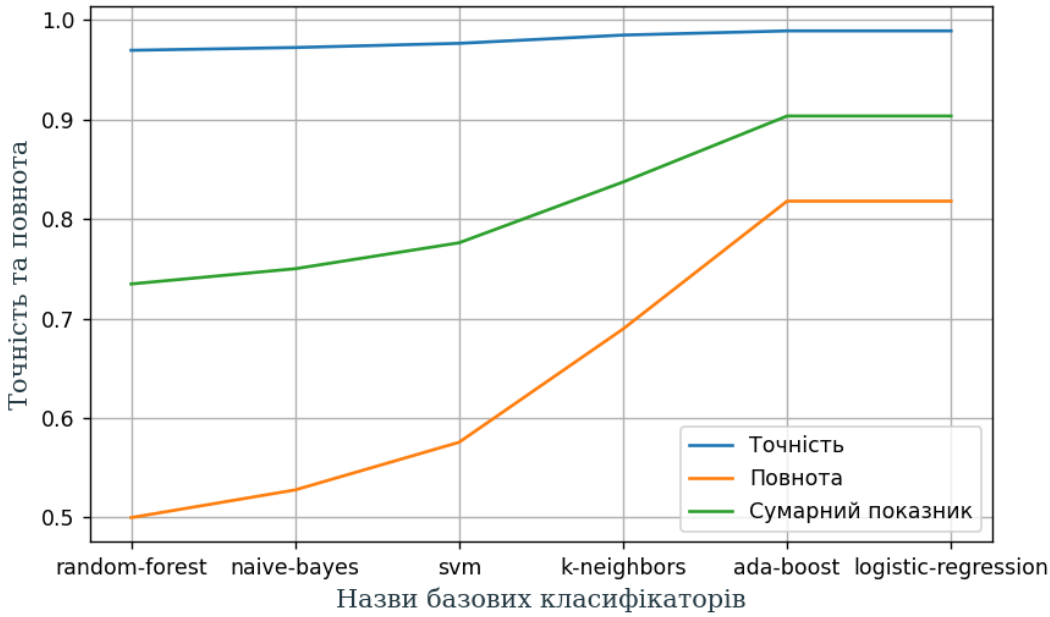


Рисунок 4.2 – Оцінка якісних показників ефективності базових класифікаторів: кращий результат модель класифікатора «ada-boost» за точністю 0.9889, повнотою 0.8179

В таблиці 4.1 наведені результати базових класифікаторів, оцінених згідно вище зазначених метрик.

Таблиця 4.1 – Результати базових класифікаторів

№	Назва класифікатора	Точність	Повнота	Матриця невідповідностей
1	«random-forest»	0.9694	0.5	TP – 13 FP - 7 FN – 22 TN - 678
2	«naive-bayes»	0.9722	0.5278	TP – 15 FP - 5 FN – 13 TN -687
3	«svm»	0.9764	0.5757	TP – 17 FP - 3 FN – 10 TN -690
4	«logistic-regression»	0.9884	0.8179	TP – 18 FP - 2

				FN – 7 TN -693
5	«k-neighbors»	0.9847	0.6895	TP – 17 FP - 3 FN – 5 TN -695
6	«ada-boost»	0.9889	0.8179	TP – 17 FP - 3 FN – 4 TN -696

4.2 Результати групованих та агрегованих рішень моделей

Групування та подальше агрегування, іншими словами, виведення колективного рішення мало на меті підвищити показники класифікаційної системи. На рисунку 4.3 зображений графік точності та повноти, побудований на основі оцінювання комбінацій (наборів) моделей.

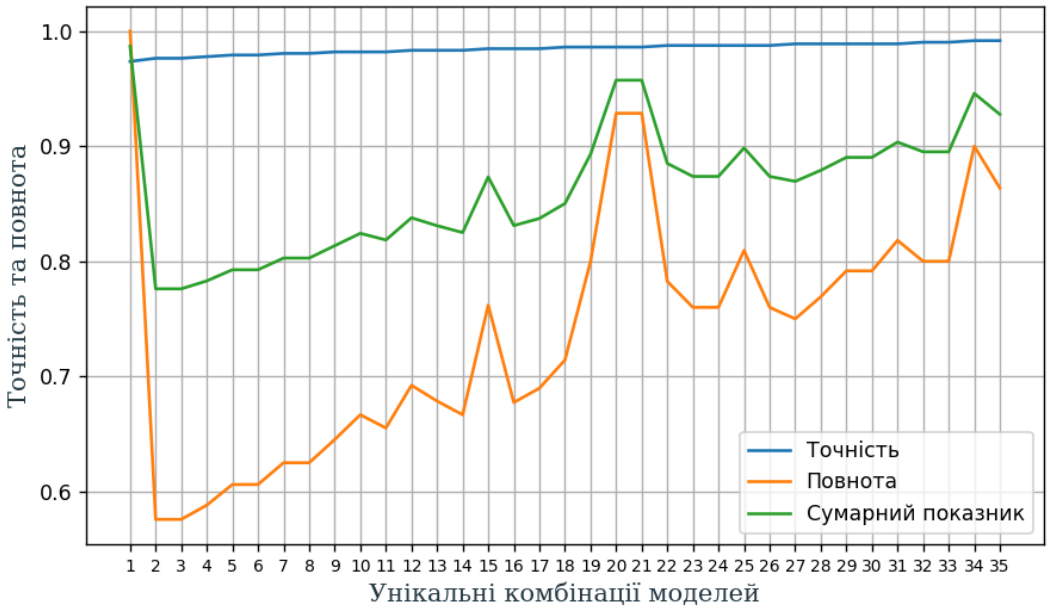


Рисунок 4.3 – Оцінка якісних показників класифікації комбінацій (наборів) моделей: кращий результат набір «ada-boost, random-forest» за точністю 0.9736, повнотою 1.0

На рисунку 4.3 по осі «y» віднумеровані комбінації базових класифікаторів, список назв алгоритмів, які використовуються у кожному наборі згідно з нумерацією наступний:

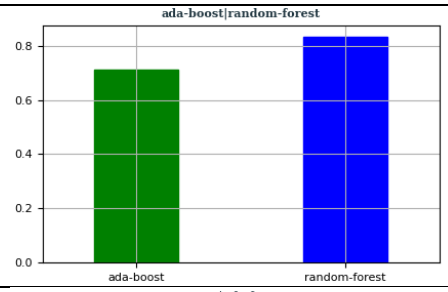
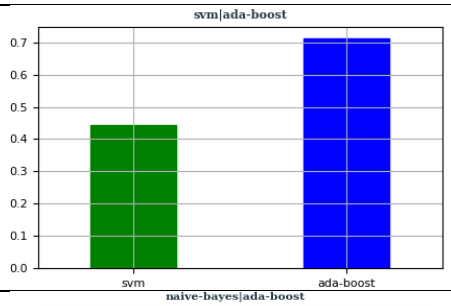
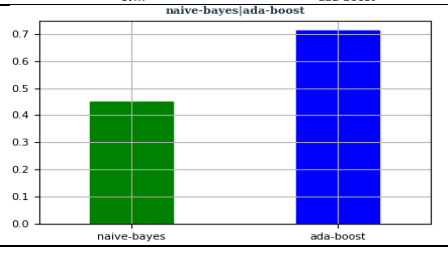
1) ada-boost|random-forest;
2) svm|ada-boost;
3) naive-bayes|ada-boost;
4) svm|naive-bayes|k-neighbors|ada-boost;
5) svm|naive-bayes|k-neighbors|ada-boost|logistic-regression;
6) naive-bayes|k-neighbors|ada-boost;
7) svm|k-neighbors;
8) svm|naive-bayes|k-neighbors|ada-boost|random-forest;
9) svm|naive-bayes|ada-boost;
10) naive-bayes|k-neighbors|ada-boost|random-forest;
11) naive-bayes|k-neighbors|ada-boost|logistic-regression;
12) svm|logistic-regression;
13) svm|naive-bayes|k-neighbors|ada-boost|random-forest|logistic-regression;
14) naive-bayes|k-neighbors;
15) svm|random-forest;
16) svm|naive-bayes|k-neighbors;
17) svm|naive-bayes|k-neighbors|logistic-regression;
18) k-neighbors|ada-boost;
19) k-neighbors|ada-boost|random-forest|logistic-regression;
20) ada-boost|random-forest|logistic-regression;
21) random-forest|logistic-regression;
22) naive-bayes|random-forest;
23) naive-bayes|k-neighbors|random-forest;
24) naive-bayes|k-neighbors|ada-boost|random-forest|logistic-regression;
25) k-neighbors|ada-boost|random-forest;
26) k-neighbors|ada-boost|logistic-regression;
27) svm|naive-bayes;
28) svm|naive-bayes|random-forest;
29) svm|naive-bayes|k-neighbors|random-forest;
30) naive-bayes|logistic-regression;
31) ada-boost|logistic-regression;
32) svm|naive-bayes|logistic-regression;

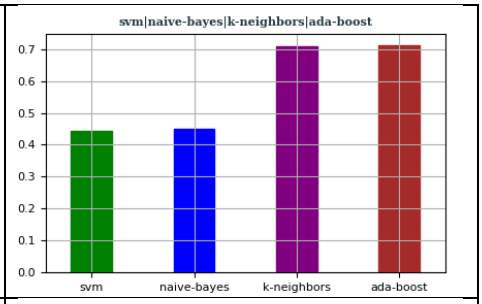
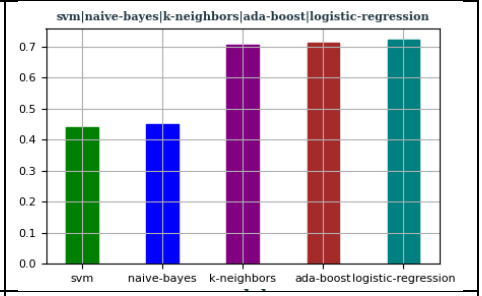
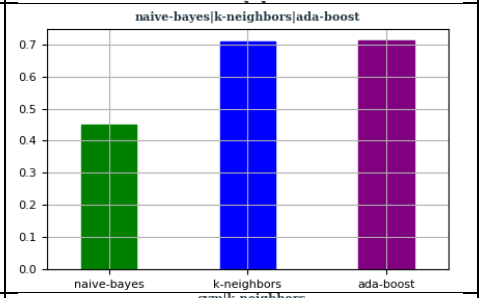
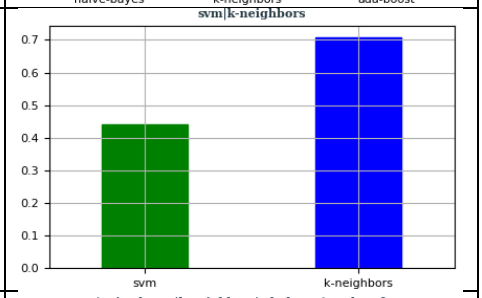
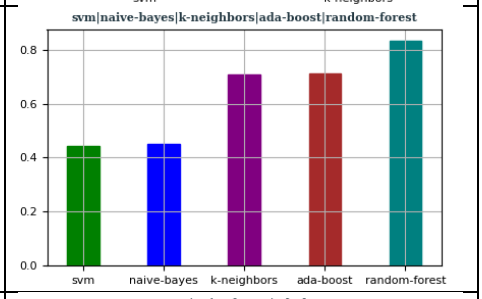
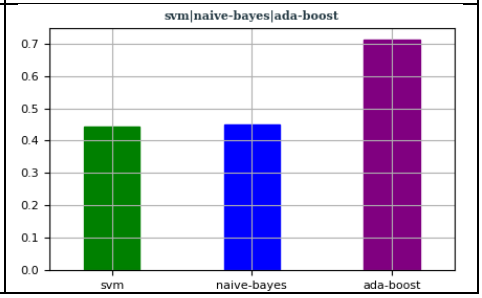
33) naive-bayes|k-neighbors|logistic-regression;
34) k-neighbors|random-forest;
35) k-neighbors|logistic-regression

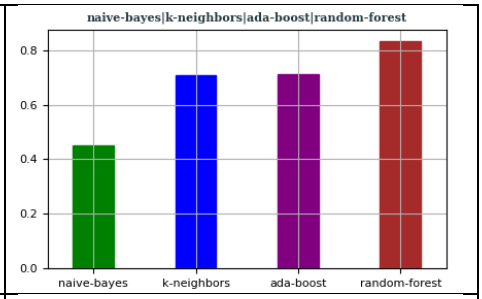
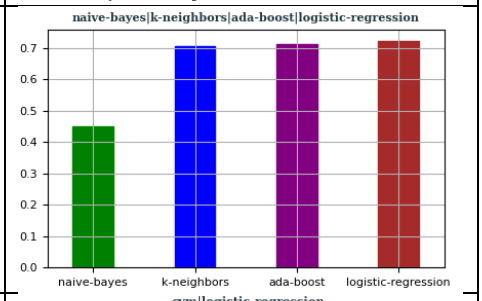
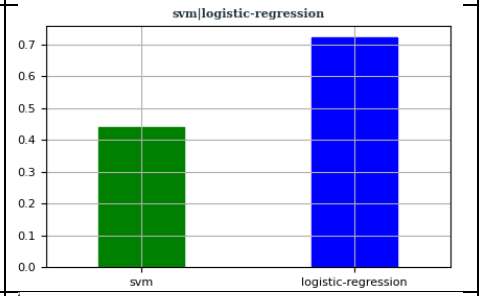
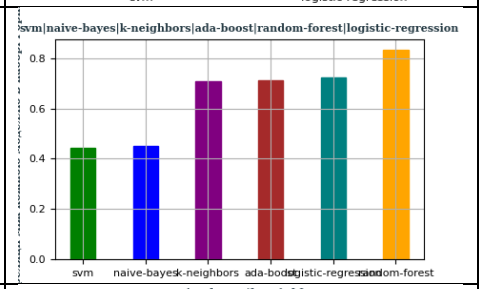
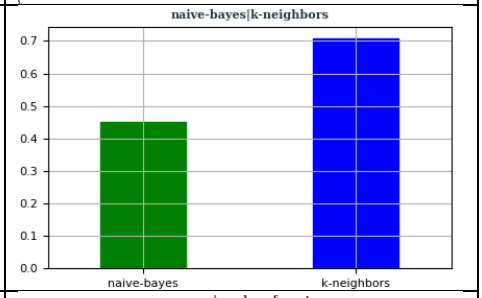
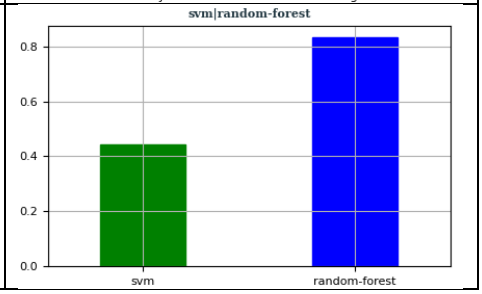
Як видно з графіків, зображених на рисунках 4.2-4.3, точність та повнота більша при застосуванні агрегативного підходу, аніж при застосуванні базових моделей. Цими даними доведено, що застосування агрегативних підходів для класифікації на базі ансамблевих моделей «дає свої плоди». Цікаво примітити, що повнота комбінованих наборів класифікаторів значно перевищує повноту при звичайній класифікації.

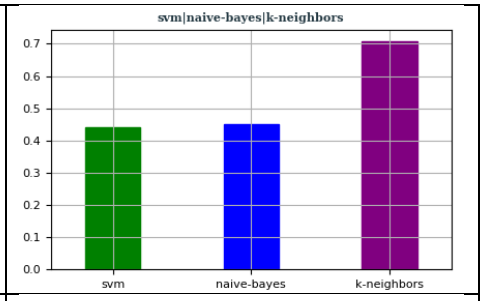
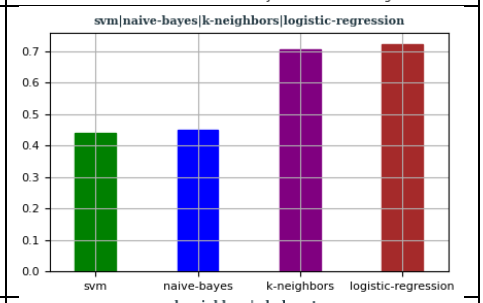
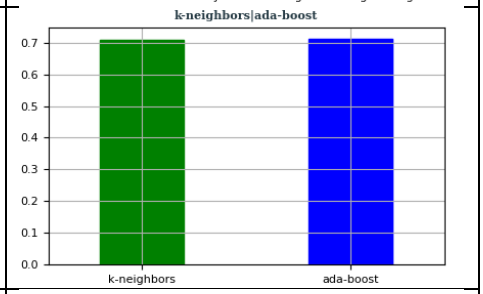
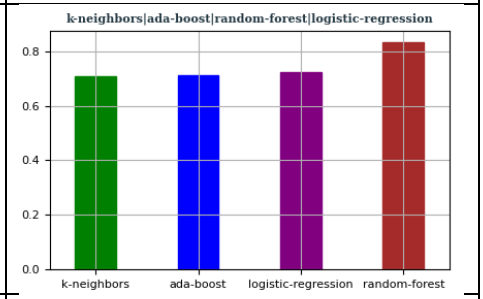
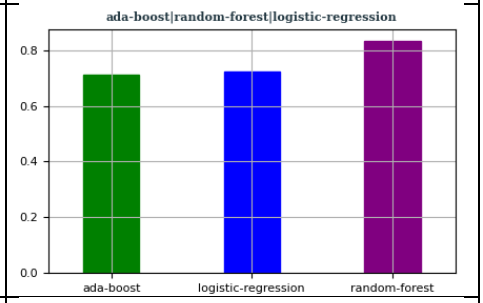
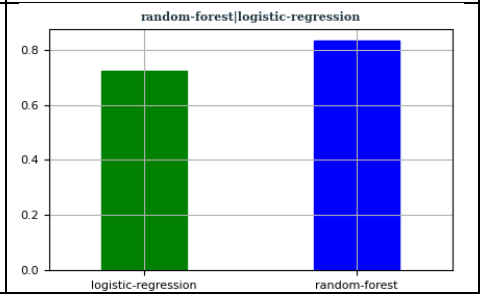
В таблиці 4.2 наведені результати наборів класифікаторів, оцінених згідно базових метрик. Слід зазначити, що в цій таблиці також наведені графіки коефіцієнтів кореляцій, які показують, наскільки істинні значення документів (оригінальні надписи – теми) пов’язані із передбаченням кожного класифікатора.

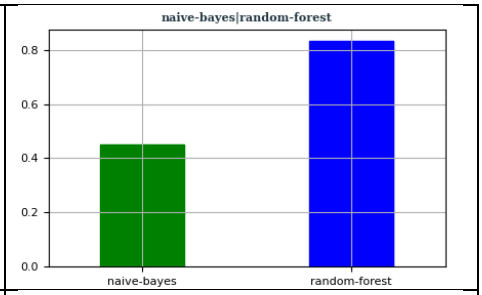
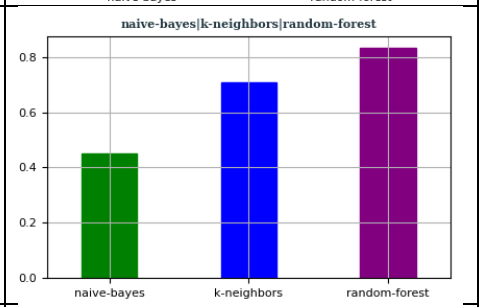
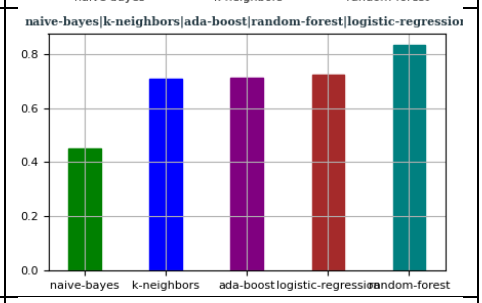
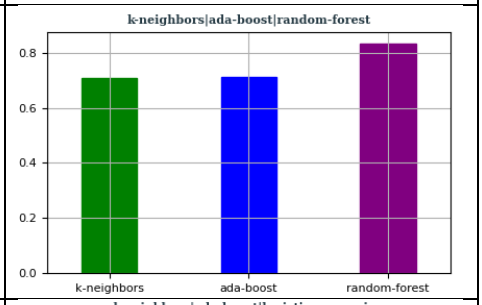
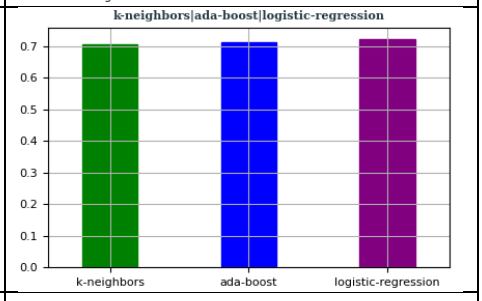
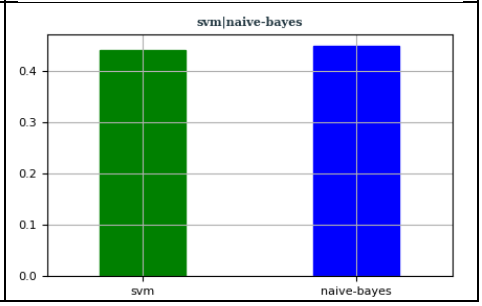
Таблиця 4.2 – Результати комбінацій моделей

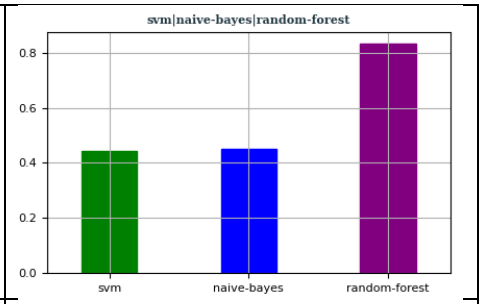
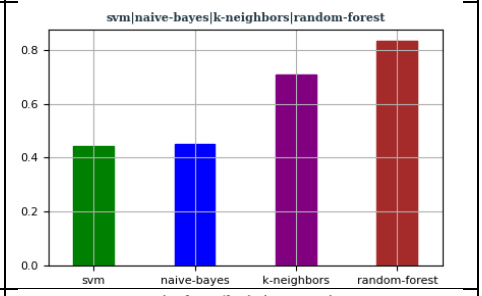
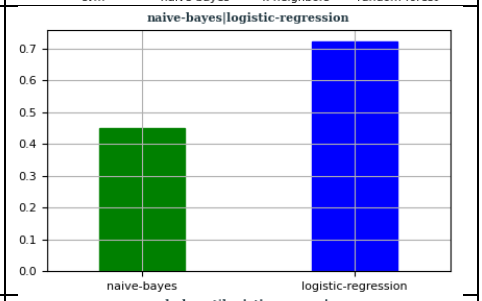
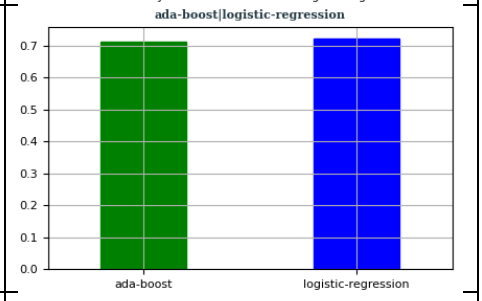
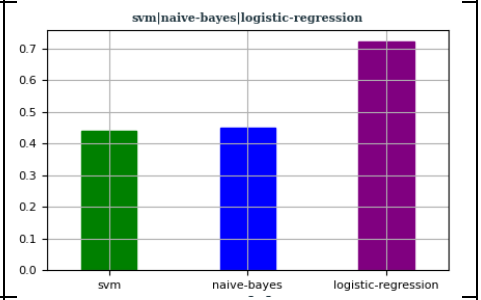
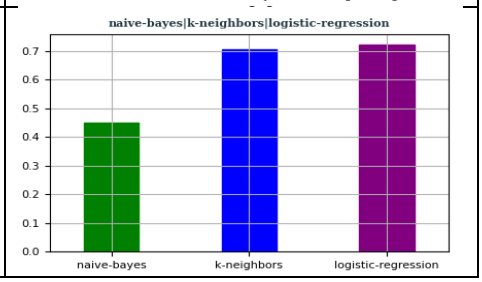
№	Назва моделей в комбінації	Точність	Повнота	Матриця невідповідностей	Гістограми показників кореляційних зв’язків
1	['ada-boost', 'random-forest']	0.9736	1	TP – 3 FP - 19 FN – 0 TN - 698	
2	['svm', 'ada-boost']	0.9764	0.5768	TP – 19 FP - 3 FN – 14 TN - 684	
3	['naive-bayes', 'ada-boost']	0.9764	0.5758	TP – 19 FP - 3 FN – 14 TN – 684	

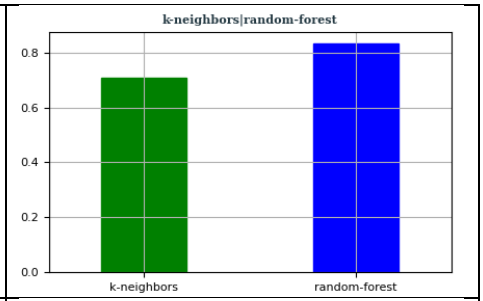
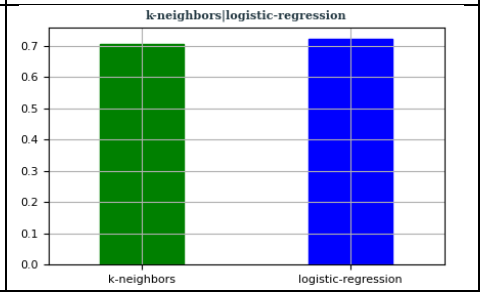
4	['svm', 'naive-bayes', 'k-neighbors', 'ada-boost']	0.9778	0.5882	TP – 20 FP - 2 FN – 14 TN - 684	
5	['svm', 'naive-bayes', 'k-neighbors', 'ada-boost', 'logistic-regression']	0.9792	0.6061	TP – 20 FP - 2 FN – 13 TN - 685	
6	['naive-bayes', 'k-neighbors', 'ada-boost']	0.9792	0.6061	TP – 20 FP - 2 FN – 13 TN - 685	
7	['svm', 'k-neighbors']	0.9806	0.625	TP – 20 FP - 2 FN – 12 TN – 686	
8	['svm', 'naive-bayes', 'k-neighbors', 'ada-boost', 'random-forest']	0.9806	0.625	TP – 20 FP - 2 FN – 12 TN - 686	
9	['svm', 'naive-bayes', 'ada-boost']	0.9819	0.6452	TP – 20 FP - 2 FN – 11 TN - 687	

10	['naive-bayes', 'k-neighbors', 'ada-boost', 'random-forest']	0.9819	0.6667	TP – 18 FP - 4 FN – 9 TN -689	
11	['naive-bayes', 'k-neighbors', 'ada-boost', 'logistic-regression']	0.9819	0.6552	TP –19 FP - 3 FN – 10 TN -688	
12	['svm', 'logistic-regression']	0.9833	0.6923	TP – 18 FP - 4 FN – 8 TN -690	
13	['svm', 'naive-bayes', 'k-neighbors', 'ada-boost', 'random-forest', 'logistic-regression']	0.9833	0.6786	TP – 19 FP - 3 FN – 9 TN -689	
14	['naive-bayes', 'k-neighbors']	0.9833	0.6667	TP –20 FP - 2 FN – 10 TN -688	
15	['svm', 'random-forest']	0.9847	0.7619	TP – 16 FP - 6 FN – 5 TN -693	

16	['svm', 'naive-bayes', 'k-neighbors']	0.9847	0.6774	TP – 21 FP - 1 FN – 10 TN -688	
17	['svm', 'naive-bayes', 'k-neighbors', 'logistic-regression']	0.9847	0.6897	TP – 20 FP - 2 FN – 9 TN -689	
18	['k-neighbors', 'ada-boost']	0.9861	0.7143	TP – 20 FP - 2 FN – 8 TN -690	
19	['k-neighbors', 'ada-boost', 'random-forest', 'logistic-regression']	0.9861	0.8	TP – 16 FP - 6 FN – 4 TN -694	
20	['ada-boost', 'random-forest', 'logistic-regression']	0.9861	0.9286	TP – 13 FP - 9 FN – 1 TN -697	
21	['random-forest', 'logistic-regression']	0.9861	0.9286	TP – 13 FP - 9 FN – 1 TN -697	

22	['naive-bayes', 'random-forest']	0.9875	0.7626	TP – 18 FP - 4 FN – 5 TN -693	
23	['naive-bayes', 'k-neighbors', 'random-forest']	0.9875	0.76	TP – 19 FP - 3 FN – 6 TN -692	
24	['naive-bayes', 'k-neighbors', 'ada-boost', 'random-forest', 'logistic-regression']	0.9875	0.76	TP – 19 FP - 3 FN – 6 TN -692	
25	['k-neighbors', 'ada-boost', 'random-forest']	0.9875	0.8095	TP – 17 FP - 5 FN – 4 TN -694	
26	['k-neighbors', 'ada-boost', 'logistic-regression']	0.9875	0.76	TP – 19 FP - 3 FN – 6 TN -692	
27	['svm', 'naive-bayes']	0.9889	0.75	TP – 21 FP - 1 FN – 7 TN -691	

28	['svm', 'naive-bayes', 'random-forest']	0.9889	0.7692	TP – 20 FP - 2 FN – 6 TN -692	
29	['svm', 'naive-bayes', 'k-neighbors', 'random-forest']	0.9889	0.7917	TP – 19 FP - 3 FN – 5 TN -693	
30	['naive-bayes', 'logistic-regression']	0.9889	0.7917	TP – 19 FP - 3 FN – 5 TN -693	
31	['ada-boost', 'logistic-regression']	0.9889	0.8182	TP – 18 FP - 4 FN – 4 TN -694	
32	['svm', 'naive-bayes', 'logistic-regression']	0.9903	0.8	TP – 20 FP - 2 FN – 5 TN -693	
33	['naive-bayes', 'k-neighbors', 'logistic-regression']	0.9903	0.8	TP – 20 FP - 2 FN – 5 TN -693	

34	['k-neighbors', 'random-forest']	0.9917	0.9	TP – 18 FP - 4 FN – 2 TN -696	
35	['k-neighbors', 'logistic-regression']	0.9917	0.8636	TP – 19 FP - 3 FN – 3 TN -695	

Ще одним аспектом дослідження є показ коефіцієнтів кореляцій між істинними значеннями та передбаченнями базових класифікаторів у кожному унікальному ансамблі.

Кореляційні зв'язки істинних значень та передбачень базових моделей всередині кожного ансамблю у відсортованому вигляді зображені на рисунках 4.4, 4.5 в різних перспективах.

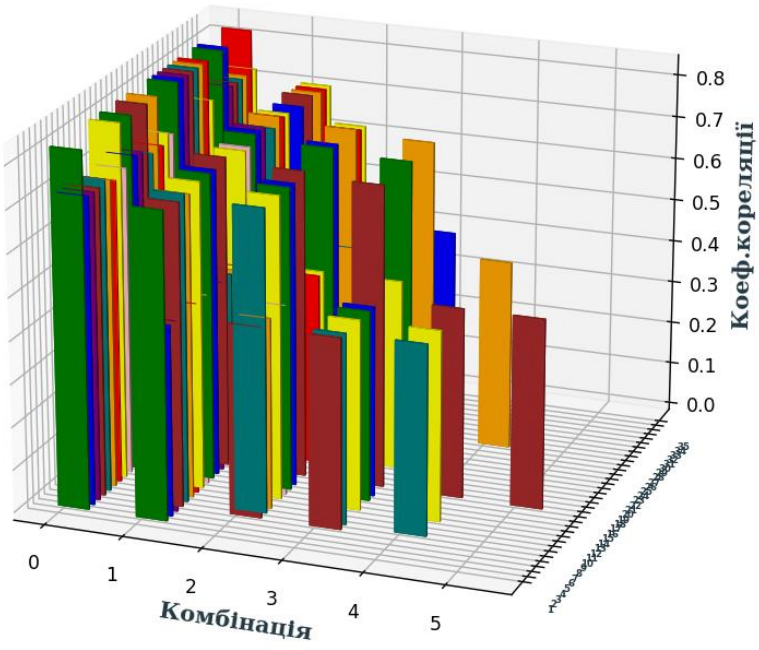


Рисунок 4.4 – Графіки гістограм коефіцієнтів кореляцій

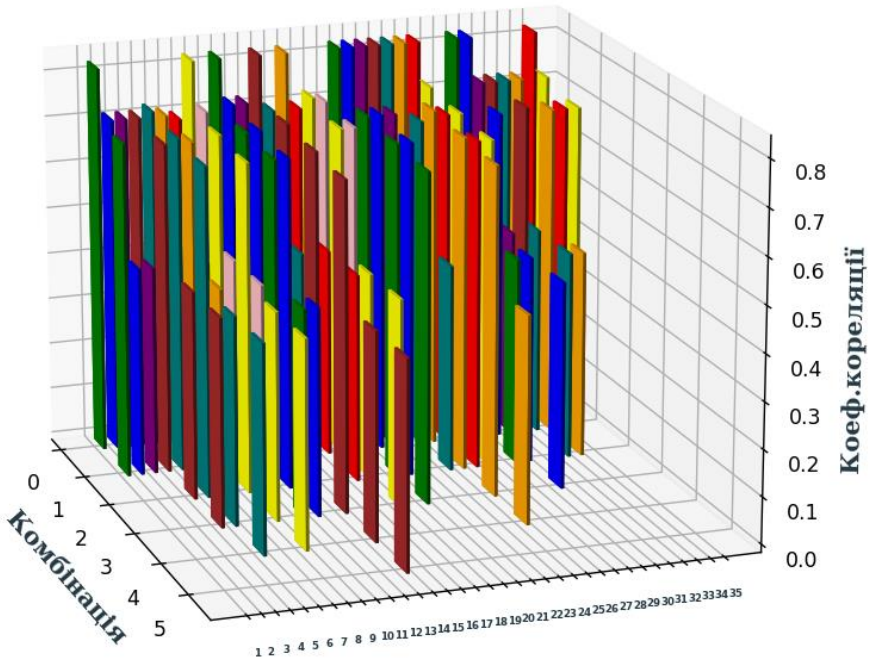


Рисунок 4.5 – Графіки гістограм коефіцієнтів кореляцій. Бокова перспектива

Варто відзначити, що інформація подається у вигляді наборів даних в тривимірному просторі, це сприяє полегшенню в подальшому дослідженні нових закономірностей між наборами комбінацій, точністю кожної комбінації, величини залежності передбачення кожної моделі від істинного значення в середині кожної комбінації.

Висновки до розділу 4

Дослідження запропонованого методу класифікації даних на основі групування рішень моделей ансамблю показало, що класифікаційна система, яка утворена методом виведення колективного рішення працює стабільніше та видає кращі результати в порівнянні з базовими класифікаторами. В особливому порядку покращення набуває повнота передбачень. За допомогою кореляційних зв’язків між прогнозами моделей у ансамблі та істинними значеннями прослідковується закономірність схильності набору моделей до хороших передбачень.

Загальні висновки

В ході виконання роботи розроблено метод виведення колективного рішення моделей ансамблю на основі кластерного аналізу. Під час процесу розробки даного методу проведено аналіз методів класифікації та підходи до побудови ансамблів моделей. Задля приведення вхідних даних у допустиму форму, розроблено модель обробки природної мови. Реалізовано систему класифікації даних із використанням алгоритму кластеризації в якості функції агрегатора передбачень.

Важливим аспектом наукової роботи було знаходження коефіцієнтів кореляцій між істинними значеннями та передбаченнями кожного класифікатора у кожній комбінації. З'ясувавши кореляційні показники, було визначено «схильність» конкретної моделі у конкретній комбінації до похибки або правильного рішення. Іншими словами, кореляційна залежність між правильною відповіддю і результатом базової моделі у комбінації з іншими моделями показує наскільки даній моделі «комфортно» у поєднанні з іншими моделями, тобто, дослідивши кореляційні зв'язки, можна винайти такі комбінації базових класифікаторів, в яких кожен примітив (модель) покращує колективне рішення.

В кінцевому результаті проведено дослідження ефективності запропонованого методу класифікації на основі групування рішень моделей. З'ясовано, що даний підхід покращує результативність класифікаційної системи, підвищенню підлягають такі метрики, як точність та повнота.

Одним із шляхів вдосконалення реалізованої інформаційної системи класифікації є добавлення можливості зміни корпусу даних та варіативність у методах нормалізації тексту (винесення в окремий модуль нормалізації даних), що уможливить конфігурування стосовно методу підготовки даних.

Наступним шляхом вдосконалення результативності мета-алгоритму є підбір базових класифікаторів на основі вирахуваного коефіцієнта кореляції, таким чином, система зможе створювати ансамблі з найвищими показниками метрики точності.

Перелік посилань

1. Baud RH, Rassinoux AM, Scherrer JR. Natural language processing and semantical representation. – 2017. – С. 117-125.

2. Mitchell T. M., McGraw Hill. Machine Learning. – 2018. – С. 10-16.

3. Abdollahi B., Nasraoui O. Transparency in fair machine learning: the case of explainable recommender systems. – 2019. – С. 21-35.

4. Kowsari K, Jafari Meimandi K, Heidarysafa M, Mendu S, Barnes L, Brown D. Text Classification Algorithms: A Survey. Information. – 2019. – С. 47-55.

5. Microsoft. [Електронний ресурс]. – Режим доступу: https://www.microsoft.com/uk-ua.

6. Alphabet. [Електронний ресурс]. – Режим доступу: https://www.alphabet.com/en-ww/contact.

7. Baud RH, Rassinoux AM, Scherrer JR. Natural language processing and semantical representation. – 2017. – С. 225-139.

8. Jones K.S. A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation. – 2015. – С. 11-21.

9. Chan P.K., Stolfo S.J. Toward Scalable Learning with Non-uniform Class and Cost Distributions. – 2021. – С. 115-121.

10. Chaganty N.R., Joe H. Range of correlation matrices for dependent Bernoulli random variables. Biometrika. – 2022. – С. 45-61.

11. Glenn. V. Ostir. Logistic Regression: A Nontechnical Review // American Journal of Physical Medicine & Rehabilitation. – 2016. – Volume 6. – С. 565-572.

12. Bogdan Zagajewski, Marcin Kluczek, Edwin Raczko, Ajda Njegovec, Anca Dabija and Marlena Kycko. Comparison of Random Forest Support Vector Machines and Neural Networks for Post-Disaster Forest Species Mapping of the Krkono // Remote Sensing. – 2021. – Volume 13, № 13. – С. 2581-2600.

13. J. Maudes, J. J. Rodríguez and C. García-Osorio. Disturbing neighbors diversity for decision forests in Applications of supervised and unsupervised ensemble methods // Heidelberg:Springer. – 2018. – С. 113-133.

14. V. Y. Kulkarni and P. K. Sinha. Random forest classifiers: A survey and future research directions // International Journal of Advanced Computer Technology. – 2013. - Volume 36, № 1. – C. 1144-1153.

15. Liu S., Cai W., Che H., Pujol S., Kikinis R. Multimodal neuroimaging feature learning for multiclass diagnosis of Alzheimer's disease. IEEE Trans Biomed Eng. – 2016. – C. 153-175.

16. Conroy B., Eshelman L., Potes C., Xu-Wilson M. A dynamic ensemble approach to robust classification in the presence of missing data [journal article] Machine Learning. – 2021. – C 443-463.

17. Sathitratanacheewin S., Sunanta P., Pongpirul K. Deep learning for automated classification of tuberculosis-related chest X-Ray: dataset distribution shift limits diagnostic performance generalizability. – 2021. – C 325-397.

18. H. M. Gomes J. P., Barddal F. Enembreck and A. Bifet. A survey on ensemble learning for data stream classification // ACM Computing Surveys (CSUR). – 2019. – C 25-97.

19. Ul-Saufie A.Z., Yahaya A.S., Ramli N.A., Rosaida N., Hamid H.A. Future daily PM10 concentrations prediction by combining regression models and feedforward backpropagation models with principle component analysis (PCA). Atmos. Environ. – 2017. – C 621-630.

20. Bartlett, P., Y. Freund, W. S. Lee, and R. E. Schapire. Boosting the margin: A new explanation for the effectiveness of voting methods. – 2012. – C 121-139.

21. Arabameri A., Pradhan B., Lombardo L. Comparative assessment using boosted regression trees, binary logistic regression, frequency ratio and numerical risk factor for gully erosion susceptibility modelling. – 2021. – C 234-286.

22. Galvez-López D, Tardos J.D. Bags of binary words for fast place recognition in image sequences. IEEE Trans. – 2021. – C 1188-1197.

23. Colson Jean-Pierre. From Chinese word segmentation to extraction of constructions: two sides of the same algorithmic coin, in Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions. – 2018. – C 41-50.

24. Hewitt J., Manning C.D. A structural probe for finding syntax in word representations. – 2019. – С 156-211.

25. Hewitt J., Manning C.D. A structural probe for finding syntax in word representations. – 2019. – С 234-251.

26. LinMaxMatch. [Електронний ресурс]. – Режим доступу: https://ai.googleblog.com/2021/12/a-fast-wordpiece-tokenization-system.html.

27. J. Singh, V. Gupta. Text stemming: approaches, applications, and challenges ACM Comput. Surv. – 2016. – С 1-46.

28. J.B. Lovins, Development of a stemming algorithm. Mech. Transl. Comput. Linguistics 11. – 2017. – С 22-31.

29. Gaydhani A., Doma V., Kendre S., Bhagwat L. Detecting hate speech and offensive language on twitter using machine learning: an N-gram and TFIDF based approach. – 2018. – С 87-112.

30. Bogdan Zagajewski, Marcin Kluczek, Edwin Raczko, Ajda Njegovec, Anca Dabija and Marlena Kycko. Comparison of Random Forest Support Vector Machines and Neural Networks for Post-Disaster Forest Species Mapping of the Krkono // Remote Sensing. – 2021. – Volume 13, № 13. – С. 2481.

31. Domingos, P., Pazzani. M. Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier // Machine Learning 29. – 2017. – С. 103–130.

32. J. Maudes, J. J. Rodríguez and C. García-Osorio. Disturbing neighbors diversity for decision forests in Applications of supervised and unsupervised ensemble methods // Heidelberg:Springer. – 2018. – С. 113-133.

33. C. X. Zhang and J. S. Zhang. RotBoost: A technique for combining rotation forest and AdaBoost // Pattern Recognition Letters. – 2019. Volume 29, № 10. – С. 1524-1536.

34. V. Y. Kulkarni and P. K. Sinha. Random forest classifiers: A survey and future research directions // International Journal of Advanced Computer Technology. – 2013. - Volume 36, № 1. – С. 1144-1153.

35. Glenn. V. Ostir. Logistic Regression: A Nontechnical Review // American Journal of Physical Medicine & Rehabilitation. – 2016. – Volume 6. - С. 565-572.

36. Python. [Електронний ресурс]. – Режим доступу: https://www.python.org/doc/.

37. Jigui Sun, Jie Liu and Lianyu Zhao. Clustering algorithms Research, Journal of Software. – 2014. - Volume 19, № 1. – С. 48-61.

38. P. E. Danielsson. Euclidean distance mapping // Comput. Graphics Image Proc. – 2019. - Volume 14. – С. 227-248.

39. Коефіцієнт кореляції Пірсона. [Електронний ресурс]. – Режим доступу: https://subject.com.ua/sociology/dict/205.html.

ДОДАТКИ

Додаток А – Наукова публікація на тему *«МЕТОД ПОБУДОВИ АНСАМБЛІВ МОДЕЛЕЙ ДЛЯ КЛАСИФІКАЦІЇ ДАНИХ НА ОСНОВІ КОРЕЛЯЦІЙНИХ ЗВ'ЯЗКІВ РІШЕНЬ»*

УДК 004.4

СТЕБЕЛЕЦЬКИЙ М.М., МАНЗЮК Е.А., СКРИПНИК Т.К., БАГРІЙ Р.О.
Хмельницький національний університет

МЕТОД ПОБУДОВИ АНСАМБЛІВ МОДЕЛЕЙ ДЛЯ КЛАСИФІКАЦІЇ ДАНИХ НА ОСНОВІ КОРЕЛЯЦІЙНИХ ЗВ'ЯЗКІВ РІШЕНЬ

У науковій роботі висвітлюється проблема підвищення точності передбачень бінарної класифікації із використанням алгоритмів машинного навчання.

Основою інформаційної системи бінарної класифікації виступає ансамблева модель. Ця модель, в свою чергу, містить набір унікальних комбінацій базових класифікаторів – свого роду алгоритмічні примітиви. Ансамблева модель може розглядатись як деякий мета-алгоритм, який складається із унікальних наборів алгоритмів класифікації машинного навчання (ML).

Завданням ансамблевої моделі являється знаходження такої комбінації базових алгоритмів класифікації, яка б давала найвищі показнити результативності. Результативність оцінюється згідно з основними метриками ML у завданнях класифікації.

Іншим аспектом наукової роботи є створення агрегаційного механізму задля поєднання результатів базових алгоритмів класифікації. Тобто, кожна унікальна комбінація у середині ансамблю складається із набору базових моделей (передвісників), результати яких потрібно агрегувати. У даній роботі задля агрегування (усереднення) передбачень базових моделей використовується неієрархічний метод кластеризації.

Особливістю цього дослідження є знаходження коефіцієнтів кореляцій базових моделей у кожній комбінації. За допомогою величини кореляцій встановлюється залежність між передбаченням класифікатора (базова модель) та істинним значенням, в результаті чого відкривається простір для подальших досліджень щодо покращення ансамблевої моделі (мета-алгоритму)

Ключові слова: підвищення точності, бінарна класифікація, ансамблева модель, інформаційна система, машинне навчання, коефіцієнт кореляції, унікальна комбінація, модель, алгоритм класифікації.

MYROSLAV STEBELETSKYI, EDUARD MANZIUK, TETYANA SKRYPNYK, RUSLAN BAHRIY
Khmelnytskyi National University

METHOD OF BUILDING ENSEMBLES OF MODELS FOR DATA CLASSIFICATION BASED ON DECISION CORRELATIONS

The scientific work highlights the problem of increasing the accuracy of binary classification predictions using machine learning algorithms.

Over the past few decades, systems that consist of many machine learning algorithms, also called ensemble models, have received increasing attention in the computational intelligence and machine learning community. This attention is well deserved, as ensemble systems have proven to be very effective and extremely versatile in a wide range of problem domains and real-world applications.

One algorithm may not make a perfect prediction for a particular data set. Machine learning algorithms have their limitations, so creating a model with high accuracy is a difficult task. If you create and combine several models by combining and aggregating the results of each model, there is a chance to improve the overall accuracy, this problem is dealt with by ensembling.

The basis of the information system of binary classification is the ensemble model. This model, in turn, contains a set of unique combinations of basic classifiers - a kind of algorithmic primitives. An ensemble model can be considered as some kind of meta-algorithm, which consists of unique sets of machine learning (ML) classification algorithms.

The task of the ensemble model is to find such a combination of basic classification algorithms that would give the highest performance. The performance is evaluated according to the main ML metrics in classification tasks.

Another aspect of scientific work is the creation of an aggregation mechanism for combining the results of basic classification algorithms. That is, each unique combination within the ensemble consists of a set of basic models (harbingers), the results of which must be aggregated. In this work, a non-hierarchical clustering method is used to aggregate (average) the predictions of the base models.

A feature of this study is to find the correlation coefficients of the base models in each combination. With the help of the magnitude of correlations, the relationship between the prediction of the classifier (base model) and the true value is established, as a result of which space is opened for further research on improving the ensemble model (meta-algorithm)

Keywords: accuracy improvement, binary classification, ensemble model, information system, machine learning, correlation coefficient, unique combination, model, classification algorithm.

Мета роботи. Постановка завдання

Метою науокової роботи є розробка метода побудови ансамблів моделей для класифікації даних на основі кореляційних зв'язків рішень. Використовується агрегаційний алгоритм, який усереднює показники передбачень базових моделей у кожній унікальній ансамблевій комбінації. Додатковим завданням являється вирахування кореляційних показників між результатами алгоритмів класифікації та істинними значеннями, ці показники повинні вираховуватись для кожної унікальної комбінації моделей у ансамблі.

Задля наглядності результативності наукової роботи, слід реалізувати систему візуалізації показників основних метрик алгоритмів. Візуалізація показників – важлива частина роботи, оскільки на базі цього можна наочно оцінити результати моделей, провести аналіз задля подальшого дослідження.

Виходячи із поставленої мети, поставленою задачею буде розробка інформаційної системи бінарної класифікації, яка буде мати наступні особливості:

- система реалізована модульно, кожен етап дослідження (обробка тексту, завантаження базових класифікаторів, складання унікальних комбінацій, усереднення результатів моделей, візуалізація даних) повинен представляти собою окремий програмний модуль;
- можливість заміни базових алгоритмів класифікації;
- можливість заміни функції агрегатора моделей;
- графічна візуалізація результативності базових класифікаторів та ансамблю задля порівняння;
- графічна візуалізація кореляцій між істинними значеннями та передбаченнями ансамблевих комбінацій у вигляді набору гістограм в двохвимірному та трьохвимірному просторах.

В ході дослідження потрібно виконати наступні завдання:

- провести аналіз сучасних ансамблевих підходів;
- порівняти застосування відомих методів у предметній області на показнику ефективності;

— дослідити ефективність запропонованих рішень методом експериментальних досліджень на відомих корпусах даних.

Особливу увагу слід зазначити на модульності системи класифікації, оскільки така архітектура додатку сприятиме на продуктивність дослідження, адже система дозволятиме виконувати конфігурацію базових моделей (алгоритмів класифікації), можлива заміна функції агрегатора, модулі візуалізації та оцінювання теж підлягають заміні при необхідності.

Складання мета-алгоритму із використанням наборів унікальних комбінацій базових алгоритмів класифікації. Створення ансамблю

Завданням дослідження є покращення результатів мета-алгоритму машинного навчання у задачах класифікації, тому детальне висвітлення інформації стосовно базових алгоритмів класифікації в цій роботі не проводиться, так само як і не розказується про нормалізацію даних (тексту) та навчання і тренування моделей [1, 2]. Також слід зазначити, що набір базових моделей не є константим, його можна замінити при необхідності подальшого дослідження.

В якості базових класифікаторів можна використовувати безліч популярних алгоритмів. В даному випадку, в якості базових моделей використовуються наступні алгоритми:

— метод опорних векторів (Support vector machine)[3];
— наївний Баєсів класифікатор (Naive Bayes classifier)[4];
— метод k-найближчих сусідів (K-neighbors)[5];
— adaptive Boosting (AdaBoost)[6];
— випадковий ліс (Random forest)[7];
— логістична регресія (Logistic regression)[8].

Процес поєднання вище згаданих моделей та усереднення (оптимізація) результатів передбачень цих базових класифікаторів буде називатись створенням ансамблевої моделі.

Суть завдання не полягає в тому, щоб просто створити один ансамбль із набору примітивів, стоїть задача створення набору таких ансамблів, які будуть містити в собі тільки унікальні комбінації базових моделей. Методами комбінаторики потрібно створити набори унікальних ансамблів, щоб в подальшому вибрати найкращий ансамбль із створеного набору.

Функція, яка задовільняє поставлену комбінаторну задачу, наведена у лістингу 1 та реалізована на мові програмування Python [9]. Слід зазначити, що інформаційна система класифікації також реалізована на попередньо зазначеній мові програмування, оскільки дана «платформа» містить в собі багато рішень, пов’язаних із роботою ML (machine learning), що, в свою чергу, полегшує процес проведення дослідження, адже не потрібно відволікатись на реалізацію примітивів при написанні мета-алгоритму.

Лістинг 1 - функція залежності кількості примітивів (x) до кількості унікальних комбінацій (y)

```
def y_function(x):
    y = 0
    for i in range(0, x):
        for j in range(i + 1, x):
            for k in range(j, x):
                y += 1
    return y
```

За допомогою вище зображеного алгоритму створюються унікальні набори комбінацій. Графік залежності кількості базових моделей класифікації до кількості створених унікальних ансамблів зображений на рисунку 1. Даний графік дозволяє орієнтуватись у складності системи, адже дивлячись на нього, можна обрати оптимальну з точки зору навантаження на систему кількість базових класифікаторів.

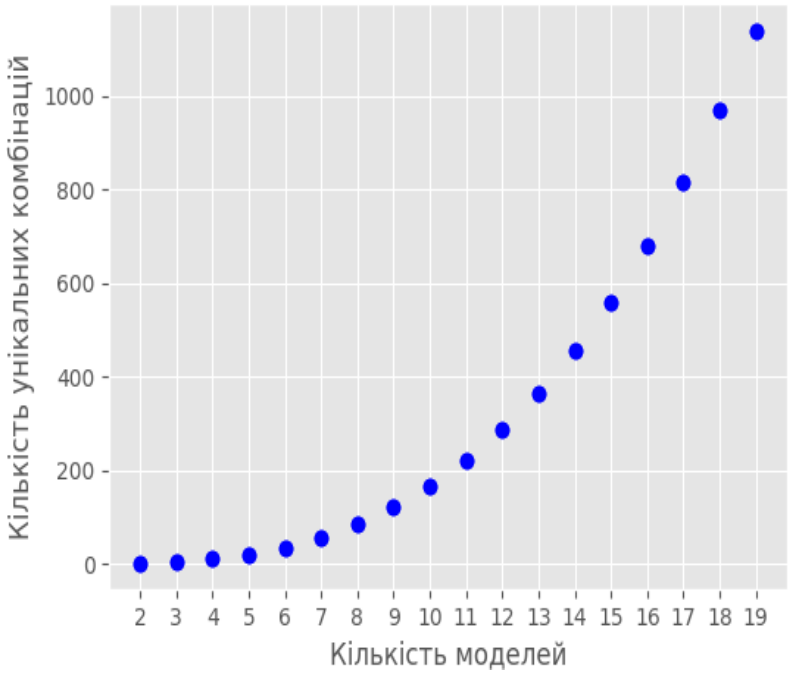


Рис. 1. Залежність кількості базових моделей класифікації до кількості створених унікальних ансамблів

Виходячи із вище зазначеного матеріалу, задача складання мета-алгоритму, який містить в собі набори унікальних ансамблів, вважається виконаною. Наступним кроком буде реалізація алгоритму усереднення (оптимізації) передбачень базових моделей у кожному унікальному ансамблі.

Алгоритм виведення колективного рішення базових моделей у ансамблі на основі неієрархічного методу кластеризації

Після успішного конструювання мета-алгоритму потрібно вирішити наступне завдання, а саме, потрібно узагальнити результати базових класифікаторів у кожному ансамблі, який складається з унікального набору моделей класифікації.

Перед тим як виконувати оптимізацію рішення ансамблю слід зазначити принцип по якому буде відбуватись усереднення передбачень [10]. Особливістю у задачі усереднення передбачень базових класифікаторів є те, що потрібно агрегувати рішення базових класифікаторів у кожному ансамблі тільки на тих порціях даних, де моделі не зійшлись у передбаченнях, тобто, агрегуються результати передбачень кожного алгоритму в ансамблі на даних, передбачення над якими хоча б у однієї моделі з комбінації були відмінні від інших передбачень у даному наборі.

Під час створення колективного рішення для кожного ансамблю моделей використовується кластериний аналіз даних. Кластерний аналіз - це статистичний метод обробки даних, який використовується з матрицями даних, в яких змінні не були попередньо розбиті на підмножини критеріїв та предикторів. Припускається, що в

основі даних лежить невпорядкований набір дискретних класів. Всі вони різні, і жоден з них не має більшої ваги, ніж інший. Кластерний аналіз полягає в тому, що елементи об'єднуються в групи, або кластери, на основі того, наскільки тісно вони пов'язані між собою. Його можна окреслити як завдання визначення підгруп даних таким чином, щоб точки даних в одній підгрупі (кластері) були схожі, в той час як точки даних в різних кластерах суттєво відрізнялися.

Процедури кластеризації можна розглядати як "пре-класифікаційні" в тому сенсі, що дослідник не використовував попередні судження для розбиття об'єктів. Однак вважається, що існують неоднорідні групи даних, тобто існують "кластери".

Виконувати агрегацію рішень базових моделей у ансамблі буде алгоритм k-середніх (k-means) [11]. Алгоритм кластеризації k-середніх обчислює центроїди та ітераційно повторюється до тих пір, поки не знайде оптимальний центроїд. Передбачається, що кількість кластерів вже відома. Його також називають алгоритмом плоскої кластеризації [12]. Кількість кластерів, визначених з даних за допомогою алгоритму, позначається «K».У цьому алгоритмі точки даних відносяться до кластеру таким чином, щоб сума квадратів відстаней між точками даних та центроїдом була мінімальною.

Ключовими даними у вище згаданому алгоритмі кластеризації є критерії відстані. Вхідними даними для алгоритму k-means є матриця, що складається з відстаней між кожним об’єктом. Щоб визначити відстані між об’єктами, потрібно мати міру відстаней.

У знаходженні відстані між об’єктами допоможе Евклідова відстань [13]. Це геометрична відстань в багатовимірному просторі. Якщо об’єкти визначаються багатовимірними точками, або мають багато характеристик (стимулів), $X_i = \{ x_{i1}, x_{i2}, x_{i3}, \dots, x_{in}\}$, де $i = 1 \dots n$, відстань може бути визначена відстанню між точками $d(X_i, X_j)$, де

$$d\big(X_i, X_j\big) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2} \ . \tag{1}$$

Метод базується на мінімізації суми квадратів відстаней між кожною точкою та центром її кластера, тобто функції:

$$\sum_{i=1}^{N} d\left(x_i, m_j(x_i)\right)^2 , \tag{2}$$

де d – метрика, x_i - і-тий об’єкт даних, а $m_j(x_i)$ – центр кластера, якому на j-тій операції присвоєний елемент x_i .

Принцип алгоритму полягає в пошуку таких центрів кластерів та наборів елементів кожного кластера при наявності деякої функції Ф(°), що виражає якість поточного розбиття множини на k кластерів, коли сумарне квадратичне відхилення елементів кластерів від центрів цих кластерів буде найменшим:

$$V = arg\ \min \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 , \tag{3}$$

де k – число кластерів, S_i - отримані кластери, $i = 1,2, \dots, k$, μ_i – центри мас векторів $x_j \in S_i$.

Процес виконання алгоритму k-means в якості усереднювача (агрегатора) передбачень базових моделей у ансамблі можна розгледіти поетапно на блок-схемі, зображеній на рисунку 2. Зупинка алгоритму проводиться тоді, коли границі кластерів і розташування центроїдів не перестануть змінюватися від ітерації до ітерації, тобто на кожній ітерації в кожному кластері буде залишатися один і той же набір об’єктів. На практиці алгоритм зазвичай знаходить набір стабільних кластерів за кілька десятків ітерацій.

Задля більш наочного представлення роботи даного методу виведення колективного рішення, на рисунку 3 зображений приклад кластеризації даних результатів базових моделей . Вхідні дані на початку алгоритму являють собою ймовірності передбачень кожної моделі на порціях даних, де результати класифікації відрізнялись хоча б у однієї моделі у даному унікальному ансамблі. Порції даних, на яких результат класифікації алгоритмічних примітивів був однаковий, ігноруються.

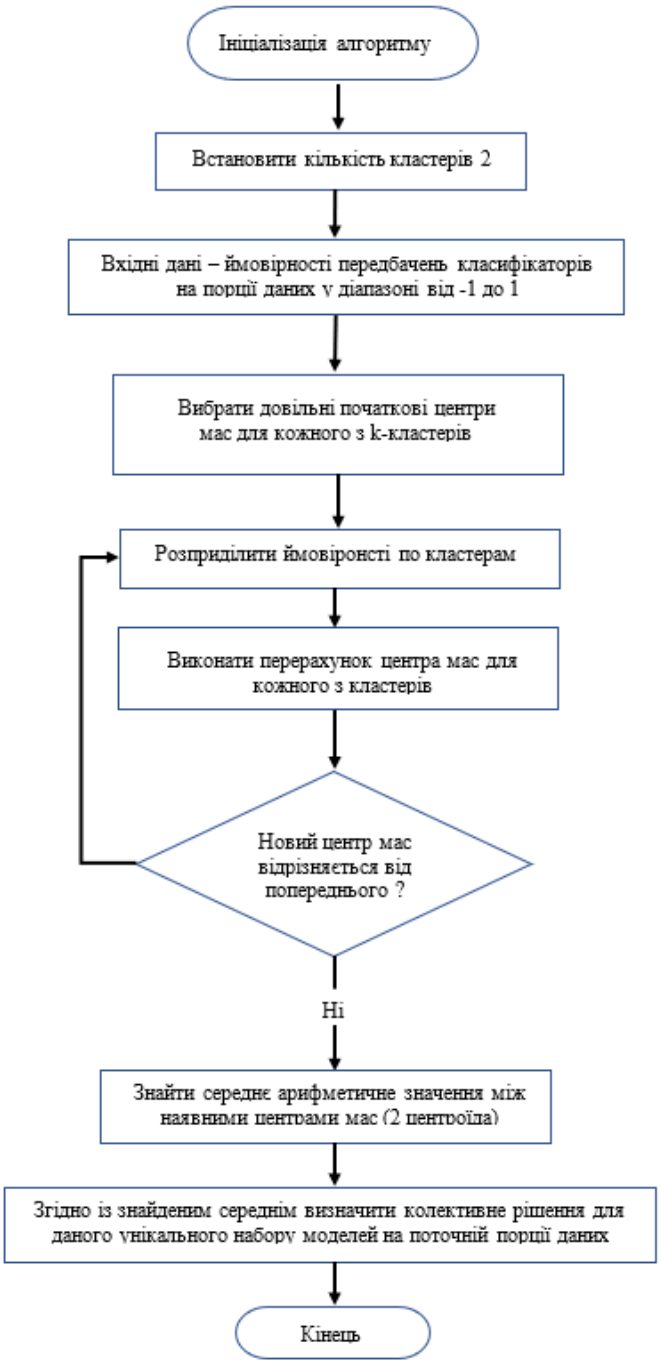


Рис. 2. Блок-схема алгоритму усереднення результатів на основі k-середніх (k-means)

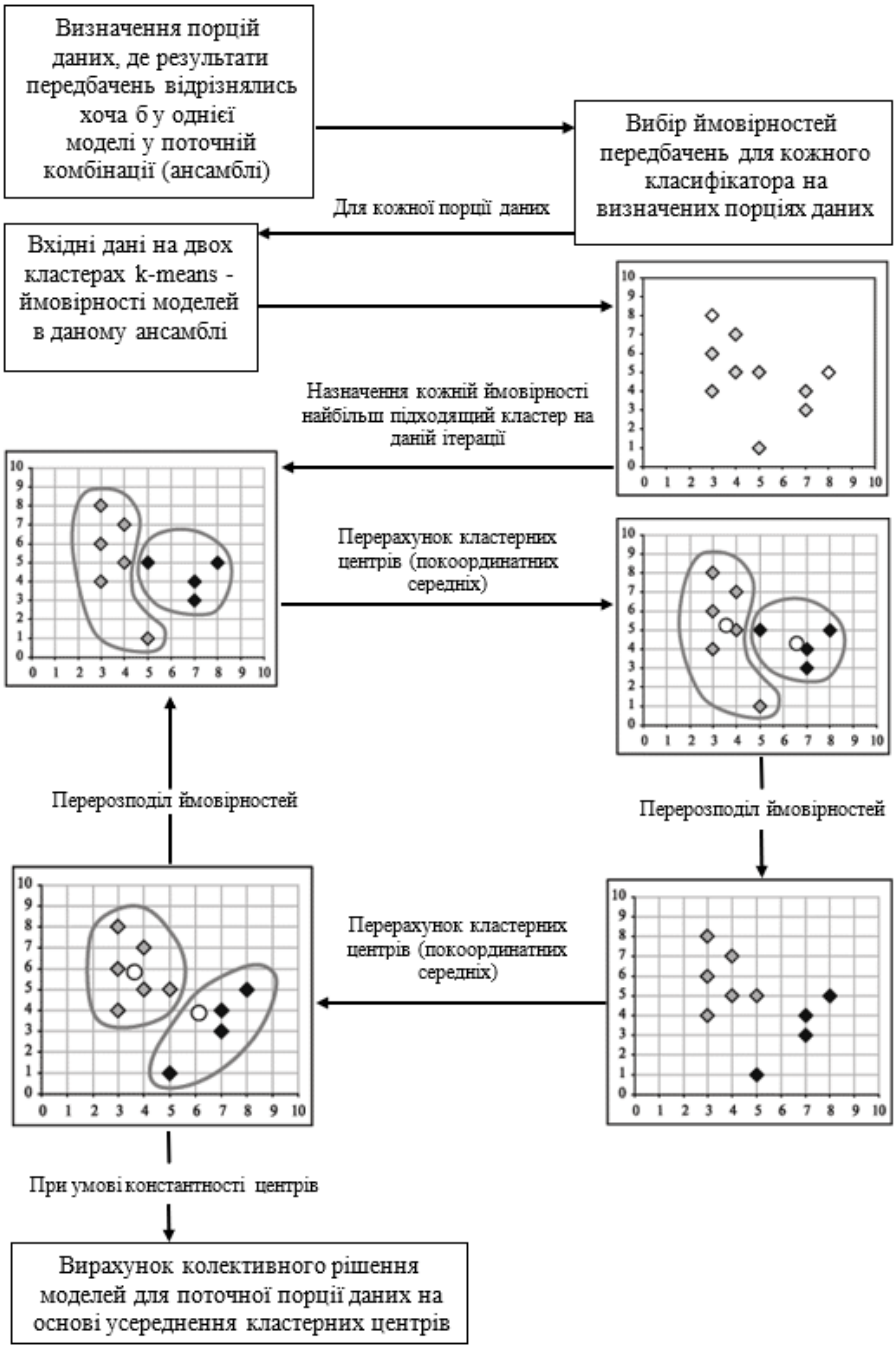


Рис. 3. Приклад роботи алгоритму виведення колективного рішення

В межах цього дослідження вище зазначений алгоритм буде виконувати розбиття вхідних даних на два кластери і працювати наступним чином:

- на вхід подаються ймовірності передбачень базових моделей. Оскільки виконується бінарна класифікація, відповідно, ймовірності можуть бути додатніми та від’ємними;
- вираховуються значення центроїдів кожного кластера, у даному випадку їх два (бінарне розбиття)
- відбувається обчислення середнього значення отриманих на попередньому кроці центроїдів

— в залежності від того, чи середнє значення додатне, чи від'ємне, усереднене рішення моделей на даній порції даних буде наступне: відноситься порція даних до заданої теми, чи ні.

Таким чином, оптимізувавши рішення базових моделей у кожному унікальному ансамблі на тих порціях даних, передбачення над якими було розбіжне хоча б у одного примітивного класифікатора, можна приступити до вирахування коефіцієнтів кореляцій та оцінки продуктивності класифікаційної системи згідно з основними метриками ML.

Результати дослідження. Порівняння продуктивності ансамблевої системи класифікації та базових алгоритмів. Показ кореляційних зв'язків

В якості оцінювальних метрик при порівнянні результативності ансамблевої моделі та базових моделей використовується точність «accuracy» та повнота «recall» [14,15]. На рисунку 4 зображений графік точності та повноти, побудований на основі оцінювання базових класифікаторів. В даному випадку – це «svm», «naive-bayes», «k-neighbors», «ada-boost», «random-forest», «logistic-regression». На рисунку 5 зображений графік точності та повноти, побудований на основі оцінювання комбінацій (наборів) моделей, або унікальних ансамблів.

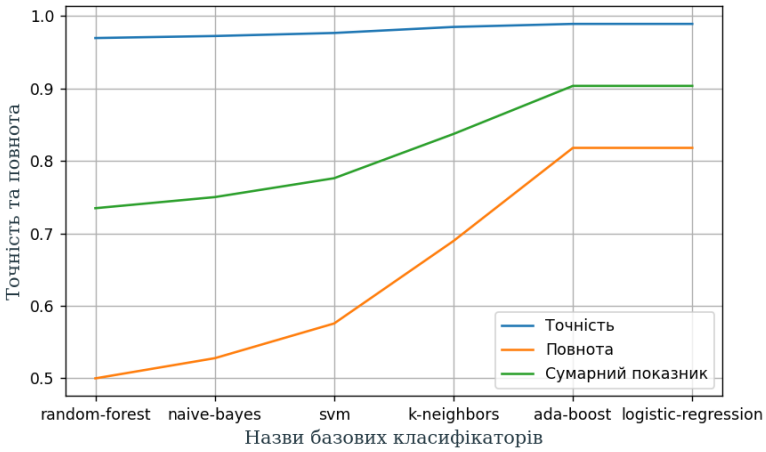


Рис. 4. Оцінка якісних показників класифікації базових класифікаторів: кращий результат модель класифікатора «ada-boost» за точністю 0.9889, повнотою 0.8179

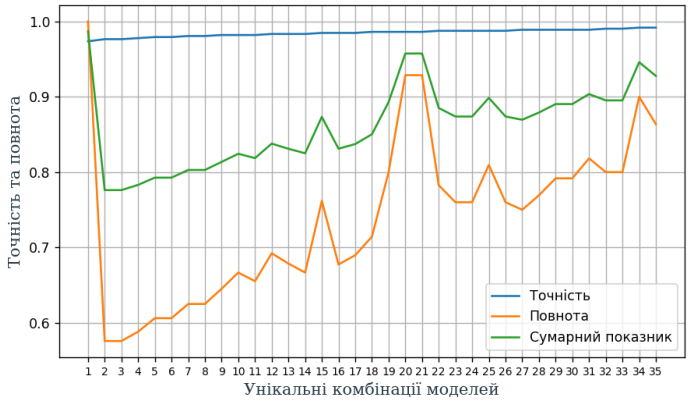


Рис. 5. Оцінка якісних показників класифікації комбінацій (наборів) моделей: кращий результат набір «ada-boost, random-forest» за точністю 0.9736, повнотою 1.0

Виходячи з даних, які зображені вище на рисунках, можна сказати, що одна з цілей дослідження виконана, адже графіки показують, що продуктивність ансамблевих моделей значно вища за продуктивність базових алгоритмів класифікації.

Ще одним аспектом дослідження є показ коефіцієнтів кореляцій між істинними значеннями та передбаченнями базових класифікаторів у кожному унікальному ансамблі [16].

Слід зазначити, що коефіцієнт кореляції є статистичним показником сили взаємозв'язку між відносними закономірностями двох змінних [17]. Значення коливаються від -1,0 до 1,0. Кореляція -1,0 показує ідеальну негативну кореляцію, тоді як кореляція 1,0 - ідеальну позитивну кореляцію. Кореляція 0,0 показує відсутність лінійної залежності між закономірностями двох змінних.

Негативна кореляція - це зв’язок між двома змінними, в яких одна змінна збільшується в міру зменшення іншої, і навпаки.

Позитивна кореляція - це взаємозв'язок між двома змінними, в яких обидві змінні рухаються в тандемі - тобто в одному напрямку.

Вирахування кореляційних зв’язків робиться для того, щоб дізнатись, наскільки кожен класифікатор у комбінації з іншими класифікаторами в межах ансамблю приносить користь у завданні підвищення точності мета-алгоритму [18].

Кореляційні зв'язки істинних значень та передбачень базових моделей всередині кожного ансамблю у відсортованому вигляді зображені на рисунках 6, 7 в різних перспективах. Варто відзначити, що інформація подається у вигляді наборів даних в трьохвимірному просторі, це сприяє полегшенню в подальшому дослідженні нових закономірностей між наборами комбінацій, точністю кожної комбінації, величини залежності передбачення кожної моделі від істинного значення в середині кожної комбінації.

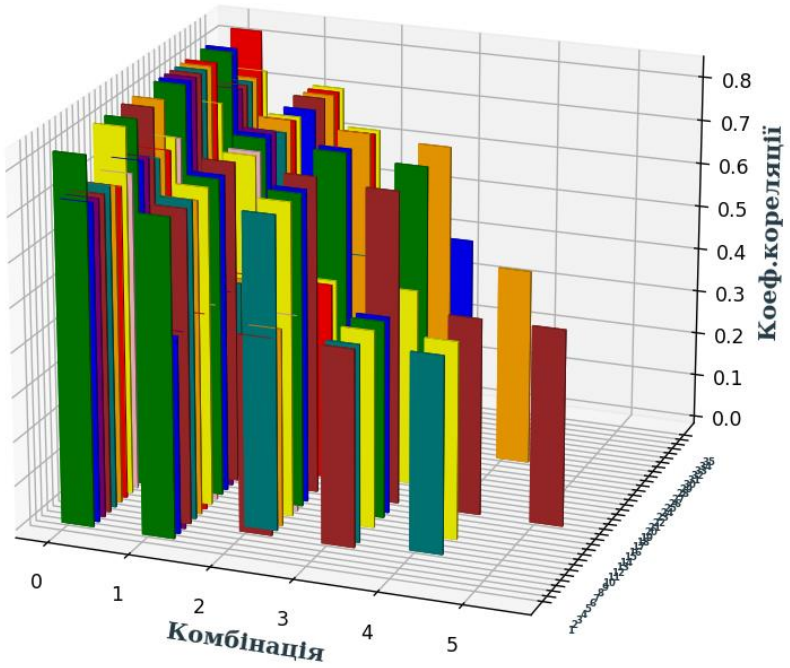


Рис. 6. Графіки гістограм коефіцієнтів кореляцій

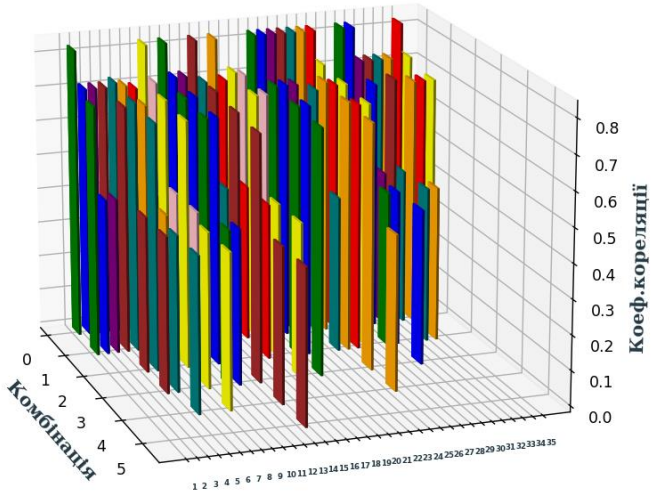


Рис. 7. Графіки гістограм коефіцієнтів кореляцій. Бокова перспектива

На рисунку 7 по осі «y» віднумеровані комбінації базових класифікаторів, список назв алгоритмів, які використовуються у кожному наборі згідно з нумерацією наступний:

1) ada-boost|random-forest;
2) svm|ada-boost;
3) naive-bayes|ada-boost;
4) svm|naive-bayes|k-neighbors|ada-boost;
5) svm|naive-bayes|k-neighbors|ada-boost|logistic-regression;
6) naive-bayes|k-neighbors|ada-boost;
7) svm|k-neighbors;
8) svm|naive-bayes|k-neighbors|ada-boost|random-forest;
9) svm|naive-bayes|ada-boost;
10) naive-bayes|k-neighbors|ada-boost|random-forest;
11) naive-bayes|k-neighbors|ada-boost|logistic-regression;
12) svm|logistic-regression;
13) svm|naive-bayes|k-neighbors|ada-boost|random-forest|logistic-regression;
14) naive-bayes|k-neighbors;
15) svm|random-forest;
16) svm|naive-bayes|k-neighbors;
17) svm|naive-bayes|k-neighbors|logistic-regression;
18) k-neighbors|ada-boost;
19) k-neighbors|ada-boost|random-forest|logistic-regression;
20) ada-boost|random-forest|logistic-regression;
21) random-forest|logistic-regression;
22) naive-bayes|random-forest;
23) naive-bayes|k-neighbors|random-forest;
24) naive-bayes|k-neighbors|ada-boost|random-forest|logistic-regression;
25) k-neighbors|ada-boost|random-forest;
26) k-neighbors|ada-boost|logistic-regression;
27) svm|naive-bayes;
28) svm|naive-bayes|random-forest;
29) svm|naive-bayes|k-neighbors|random-forest;
30) naive-bayes|logistic-regression;
31) ada-boost|logistic-regression;
32) svm|naive-bayes|logistic-regression;
33) naive-bayes|k-neighbors|logistic-regression;
34) k-neighbors|random-forest;
35) k-neighbors|logistic-regression

Висновок

В ході виконання роботи було досліджене застосування агрегативних підходів для класифікації на базі ансамблевих моделей. Реалізована інформаційна система бінарної класифікації дозволяє проводити дослідження на корпусі даних «Reuters», система розроблена таким чином, щоб уможливити всесторонню конфігурацію компонентів мета-алгоритму (ансамблю), зокрема:

- вибір базових алгоритмів класифікації відбувається у окремому модулі, що сприяє швидкій заміні при необхідності
- алгоритм агрегації рішень наборів базових моделей замінюваний, оскільки реалізований як окремий модуль
- візуалізаційні компоненти реалізовані окремо, потрібно дотримуватись контракту між модулями (інтерфейс взаємодії, зокрема, форма вхідних параметрів)

Під час дослідження застосування ансамблевої моделі на основі наборів унікальних комбінацій було проведено порівняння показників результативності даного ансамблю з результатами базових класифікаторів згідно з основними метриками ML. З'ясовано, що запропонована ансамблева модель являється більш точною та стійкою до похибок.

Важливим аспектом наукової роботи було знаходження коефіцієнтів кореляцій між істинними значеннями та передбаченнями кожного класифікатора у кожній комбінації. З'ясувавши кореляційні показники, було визначено «схильність» конкретної моделі у конкретній комбінації до похибки або правильного рішення. Іншими словами, кореляційна залежність між правильною відповіддю і результатом базової моделі у комбінації з іншими моделями показує наскільки даній моделі «комфортно» у поєднанні з іншими моделями, тобто, дослідивши кореляційні зв'язки, можна винайти такі комбінації базових класифікаторів, в яких кожен примітив (модель) покращує колективне рішення.

Задля подання показників класифікаційної системи використовується графічна система, яка складається із графіків у двох та трьох вимірних просторах. Графіки показують результативність окремих базових класифікаторів та результативність ансамблевої моделі. У трьох вимірному просторі розміщені гістограми кореляційності кожної моделі у кожній комбінації, що забезпечує хороше проглядання інформації та уможливлює подальше дослідження на основі цих даних.

Одним із шляхів вдосконалення реалізованої інформаційної системи класифікації є добавлення можливості зміни корпусу даних та варіативність у методах нормалізації тексту (винесення в окремий модуль нормалізації даних), що уможливить конфігурування стосовно методу підготовки даних.

Наступним шляхом вдосконалення результативності мета-алгоритму є підбір базових класифікаторів на основі вирахуваного коефіцієнта кореляції, таким чином, система зможе створювати ансамблі з найвищими показниками метрики точності.

Перелік посилань

1. Conroy B., Eshelman L., Potes C., Xu-Wilson M. A dynamic ensemble approach to robust classification in the presence of missing data [journal article] Machine Learning. - 2017. - P. 443-463.

2. H. M. Gomes, J. P. Barddal, F. Enembreck and A. Bifet. A survey on ensemble learning for data stream classification // ACM Computing Surveys (CSUR). – 2017. -P. 23.

3. Bogdan Zagajewski, Marcin Kluczek, Edwin Raczko, Ajda Njegovec, Anca Dabija and Marlena Kycko. Comparison of Random Forest Support Vector Machines and Neural Networks for Post-Disaster Forest Species Mapping of the Krkono // Remote Sensing. – 2021. – Volume 13, № 13. - P. 2581.

4. Domingos, P., Pazzani. M. Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier // Machine Learning 29. – 2017. – P. 103–130.

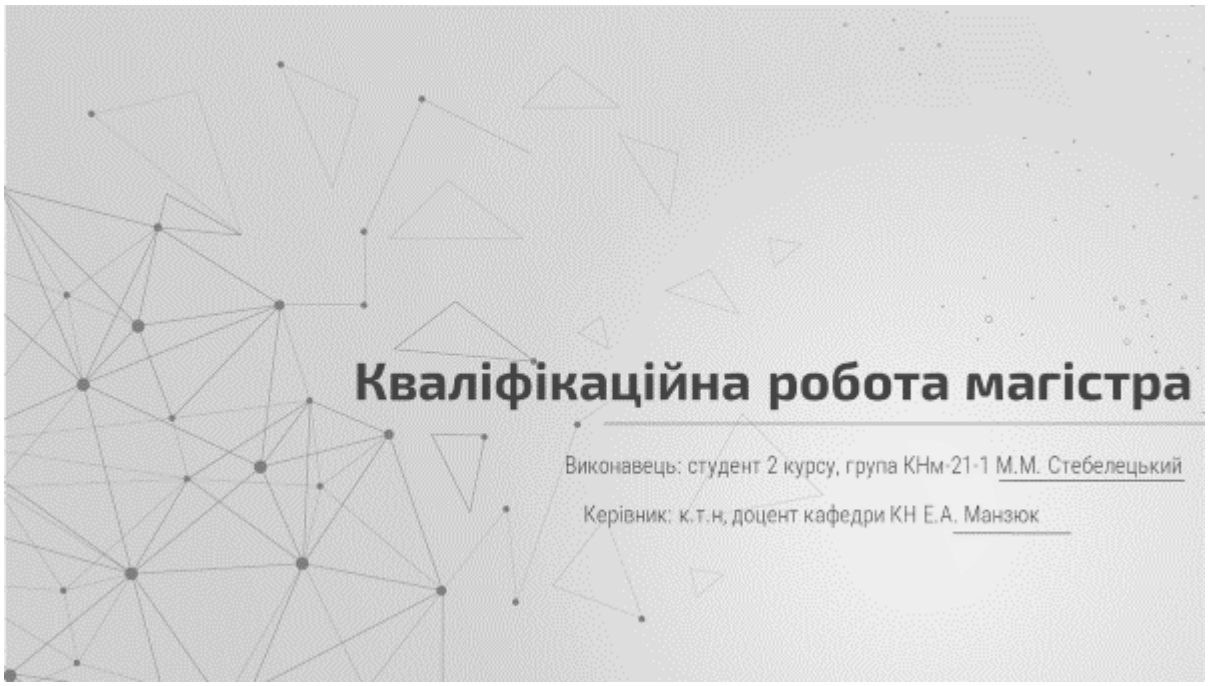
5. J. Maudes, J. J. Rodríguez and C. García-Osorio. Disturbing neighbors diversity for decision forests in Applications of supervised and unsupervised ensemble methods // Heidelberg:Springer. – 2018. – P. 113-133.
6. C. X. Zhang and J. S. Zhang. RotBoost: A technique for combining rotation forest and AdaBoost // Pattern Recognition Letters. – 2019. Volume 29, № 10. – P. 1524-1536.
7. V. Y. Kulkarni and P. K. Sinha. Random forest classifiers: A survey and future research directions // International Journal of Advanced Computer Technology. – 2013. - Volume 36, № 1. – P. 1144-1153.
8. Glenn. V. Ostir. Logistic Regression: A Nontechnical Review // American Journal of Physical Medicine & Rehabilitation. – 2016. – Volume 6. - P. 565-572.
9.Python. [Електронний ресурс]. – Режим доступу: https://www.python.org/doc/
10. A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby and R. Gavaldà. New ensemble methods for evolving data streams // Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. – 2019. – P. 139-148.
11. Jigui Sun, Jie Liu and Lianyu Zhao. Clustering algorithms Research, Journal of Software. – 2014. - Volume 19, № 1. – P. 48-61.
12. Hartigan, J.A., Wong, M.A. Algorithm AS. 136. A k-means clustering algorithm. J. Roy. Stat. Soc. – 2021. – P. 100-108.
13. P. E. Danielsson. Euclidean distance mapping // Comput. Graphics Image Proc. – 2019. - Volume 14. – P. 227-248.
14. C.X. Ling and H. Zhang. Toward Bayesian Classifiers with Accurate Probabilities // Proc. Sixth Pacific-Asia Conf. Knowledge Discovery and Data Mining. – 2012. – P. 123-134.
15. T. G. Dietterich. Ensemble learning. In The handbook of brain theory and neural networks, Cambridge, MA:MIT Press. – 2017. – Volume 2. – P. 110-125.
16. T. M. Cover. The Best Two Independent Measurements Are Not the Two Best // IEEE Trans. Systems, Man, and Cybernetics. – 2010. - Volume 4. – P. 116-117.
17. Коефіцієнт кореляції. [Електронний ресурс]. – Режим доступу: https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/
18. A. Bifet, E. Frank, G. Holmes and B. Pfahringer. Accurate ensembles for data streams: Combining restricted Hoeffding trees using stacking // Proceedings of 2nd Asian conference on machine learning. – 2018. – Volume 13. – P. 225-240.

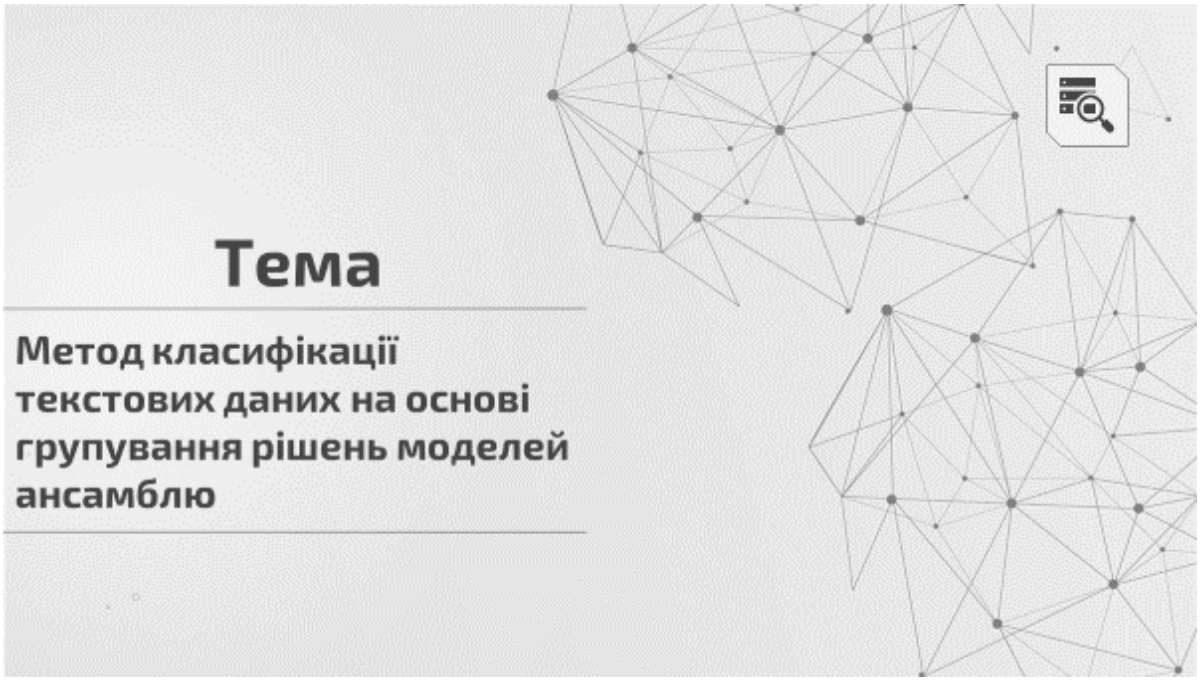
References

1. Conroy B., Eshelman L., Potes C., Xu-Wilson M. A dynamic ensemble approach to robust classification in the presence of missing data [journal article] Machine Learning. - 2017. - P. 443-463.
2. H. M. Gomes, J. P. Barddal, F. Enembreck and A. Bifet. A survey on ensemble learning for data stream classification // ACM Computing Surveys (CSUR). – 2017. -P. 23.
3. Bogdan Zagajewski, Marcin Kluczek, Edwin Raczko, Ajda Njegovec, Anca Dabija and Marlena Kycko. Comparison of Random Forest Support Vector Machines and Neural Networks for Post-Disaster Forest Species Mapping of the Krkono // Remote Sensing. – 2021. – Volume 13, № 13. - P. 2581.
4. Domingos, P., Pazzani. M. Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier // Machine Learning 29. – 2017. – P. 103–130.
5. J. Maudes, J. J. Rodríguez and C. García-Osorio. Disturbing neighbors diversity for decision forests in Applications of supervised and unsupervised ensemble methods // Heidelberg:Springer. – 2018. – P. 113-133.
6. C. X. Zhang and J. S. Zhang. RotBoost: A technique for combining rotation forest and AdaBoost // Pattern Recognition Letters. – 2019. Volume 29, № 10. – P. 1524-1536.
7. V. Y. Kulkarni and P. K. Sinha. Random forest classifiers: A survey and future research directions // International Journal of Advanced Computer Technology. – 2013. - Volume 36, № 1. – P. 1144-1153.
8. Glenn. V. Ostir. Logistic Regression: A Nontechnical Review // American Journal of Physical Medicine & Rehabilitation. – 2016. – Volume 6. - P. 565-572.
9. Python. [Elektronnyi resurs]. – Rezhym dostupu: https://www.python.org/doc/
10. A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby and R. Gavaldà. New ensemble methods for evolving data streams // Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. – 2019. – P. 139-148.
11. Jigui Sun, Jie Liu and Lianyu Zhao. Clustering algorithms Research, Journal of Software. – 2014. - Volume 19, № 1. – P. 48-61.
12. Hartigan, J.A., Wong, M.A. Algorithm AS. 136. A k-means clustering algorithm. J. Roy. Stat. Soc. – 2021. – P. 100-108.
13. P. E. Danielsson. Euclidean distance mapping // Comput. Graphics Image Proc. – 2019. - Volume 14. – P. 227-248.
14. C.X. Ling and H. Zhang. Toward Bayesian Classifiers with Accurate Probabilities // Proc. Sixth Pacific-Asia Conf. Knowledge Discovery and Data Mining. – 2012. – P. 123-134.
15. T. G. Dietterich. Ensemble learning. In The handbook of brain theory and neural networks, Cambridge, MA:MIT Press. – 2017. – Volume 2. – P. 110-125.

16. T. M. Cover. The Best Two Independent Measurements Are Not the Two Best // IEEE Trans. Systems, Man, and Cybernetics. – 2010. - Volume 4. – P. 116-117.
17. Koefitsiient koreliatsii. [Elektronnyi resurs]. – Rezhym dostupu: https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/
18. A. Bifet, E. Frank, G. Holmes and B. Pfahringer. Accurate ensembles for data streams: Combining restricted Hoeffding trees using stacking // Proceedings of 2nd Asian conference on machine learning. – 2018. – Volume 13. – P. 225-240.

Додаток Б – Презентаційні матеріали





Актуальність роботи

В магістерській роботі було **розроблено** та набув практичної реалізації метод створення мета-алгоритму із використанням **агрегаційного механізму** на основі неієрархічного методу кластеризації у завданнях **бінарної класифікації** текстового контенту.

Завданням **ансамблевої** моделі являється знаходження такої **комбінації** базових алгоритмів класифікації, яка б давала **найвищі** показнити результативності.

Важливим аспектом наукової роботи було знаходження **коефіцієнтів кореляцій** між істинними значеннями та передбаченнями кожного класифікатора у кожній комбінації. З'ясувавши кореляційні показники, було визначено «схильність» конкретної моделі у конкретній комбінації до похибки або правильного рішення.

Мета

Метою науокові роботи є розробка метода побудови ансамблів моделей для класифікації даних на основі кореляційних зв'язків рішень. Використовується **агрегаційний алгоритм**, який усереднює показники передбачень базових моделей у кожній унікальній ансамблевій комбінації.

Додатковим завданням являється вирахування **кореляційних показників** між результатами алгоритмів класифікації та істинними значеннями, ці показники повинні вираховуватись для кожної унікальної комбінації моделей у ансамблі.

Завдання

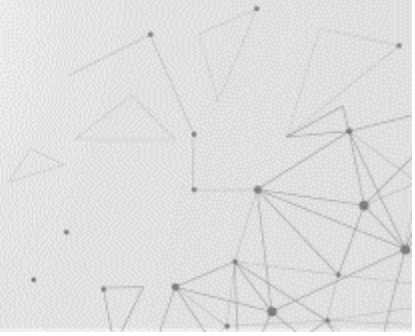
Задля досягнення поставленої мети визначені наступні завдання роботи:

1. Провести аналіз методів закласифікації текстових даних
2. Розробити модель формування ансамблю прийняття рішень з класифікації даних
3. Розробити метод класифікації даних на основі групування рішень моделей ансамблю
4. Реалізувати інформаційну технологію класифікації текстових даних на основі ансамблю моделей
5. Провести валідацію розробленого метода

Об'єкт та предмет дослідження

Об'єктом дослідження є процес забезпечення високих якісних показників класифікації за оцінками повноти та точності.

Предметом дослідження є ансамблеві моделі та методи побудови класифікаційних мета-алгоритмів на основі групування рішень базових класифікаторів.



Наукова новизна

В результаті проведеної роботи були отримані наступні результати:

- ✓ запропоновано інноваційний метод групування моделей в унікальні набори та подальше «відсіювання» результатів моделей задля створення ансамблів та агрегування результату в середині кожного ансамблю;
- ✓ розроблено метод агрегації моделей у задачах бінарної класифікації на основі кластерного аналізу;
- ✓ розроблено метод оцінювання ансамблів з використанням кореляційних показників щодо передбачень базових класифікаторів та істинних значень;
- ✓ реалізовано систему бінарної класифікації, яка містить модуль візуалізації основних метрик у двовимірному та тривимірному просторах для аналізу отриманих результатів та подальшого дослідження.

Практичне значення одержаних результатів

Наукова робота висвітлює проблему **підвищення точності** прогнозування бінарної класифікації з використанням алгоритмів машинного навчання.

Основою інформаційної системи бінарної класифікації виступає **ансамблева модель**. Ця модель, в свою чергу, містить набір унікальних **комбінацій** базових класифікаторів – свого роду алгоритмічні примітиви. Ансамблева модель може розглядатись як деякий мета-алгоритм, який складається із унікальних наборів алгоритмів класифікації машинного навчання (ML).

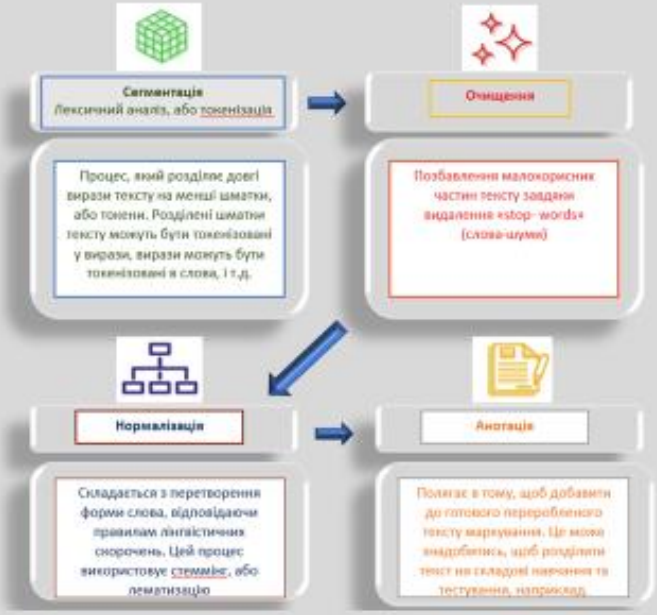
Завданням ансамблевої моделі являється знаходження такої комбінації базових алгоритмів класифікації, яка б давала **найвищі показники** результативності.

Апробація кваліфікаційної роботи

В ході виконання наукової роботи було підготовлено та подано до публікації наступні науково-дослідні матеріали:

- стаття «*МЕТОД ПОБУДОВИ АНСАМБЛІВ МОДЕЛЕЙ ДЛЯ КЛАСИФІКАЦІЇ ДАНИХ НА ОСНОВІ КОРЕЛЯЦІЙНИХ ЗВ'ЯЗКІВ РІШЕНЬ*» подана до наукового журналу «Вісник Хмельницького національного університету», отримала позитивну рецензію.
- тези «*ПОКРАЩЕННЯ РЕЗУЛЬТАТИВНОСТІ КЛАСИФІКАЦІЇ МЕТОДОМ АНСАМБЛЕВОЇ АГРЕГАЦІЇ*» опубліковані на всеукраїнській конференції АПКН-2022, Хмельницький, 31 жовтня;

Обробка природної мови для текстових даних

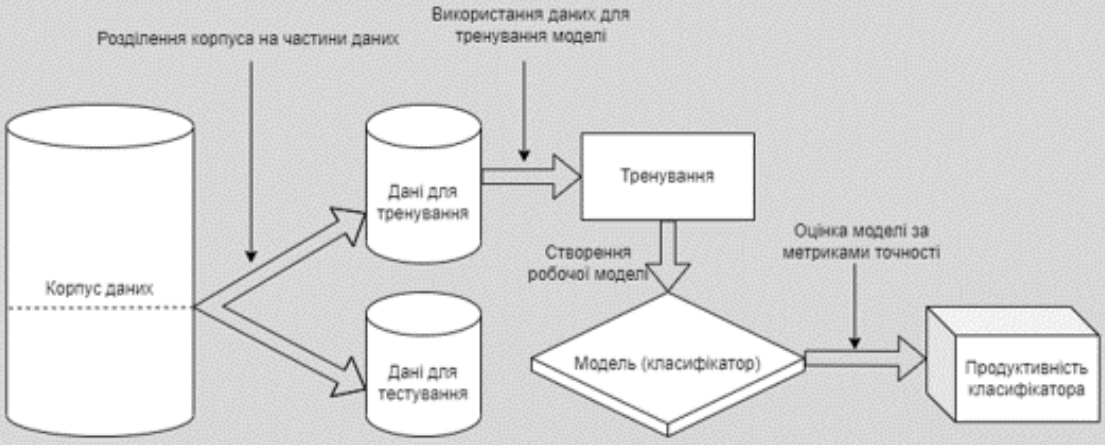


Базові моделі класифікації

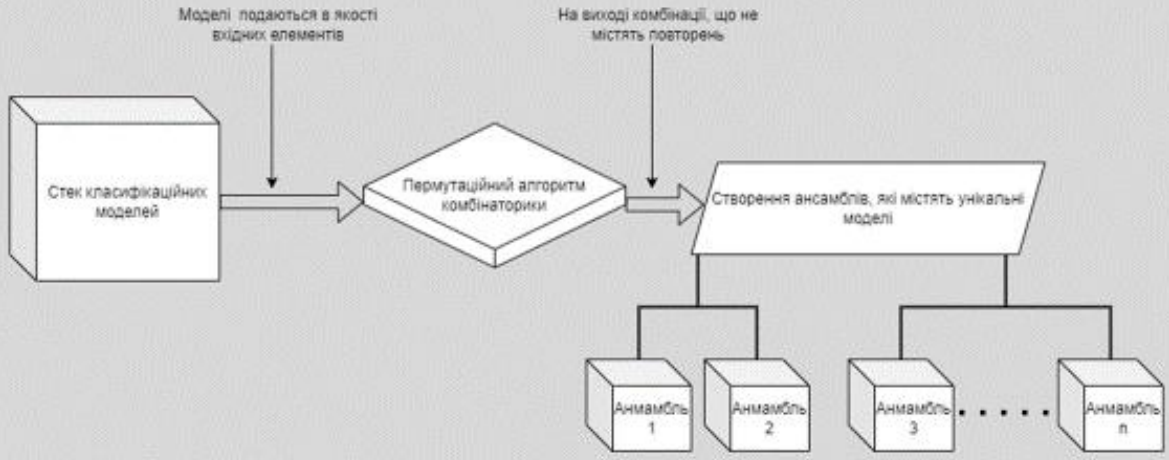
В класифікаційній системі використовуються наступні алгоритми:

- метод опорних векторів (Support vector machine);
- наївний Баєсів класифікатор (Naive Bayes classifier);
- метод k-найближчих сусідів (K-neighbors);
- adaptive Boosting (AdaBoost);
- випадковий ліс (Random forest);
- логістична регресія (Logistic regression).

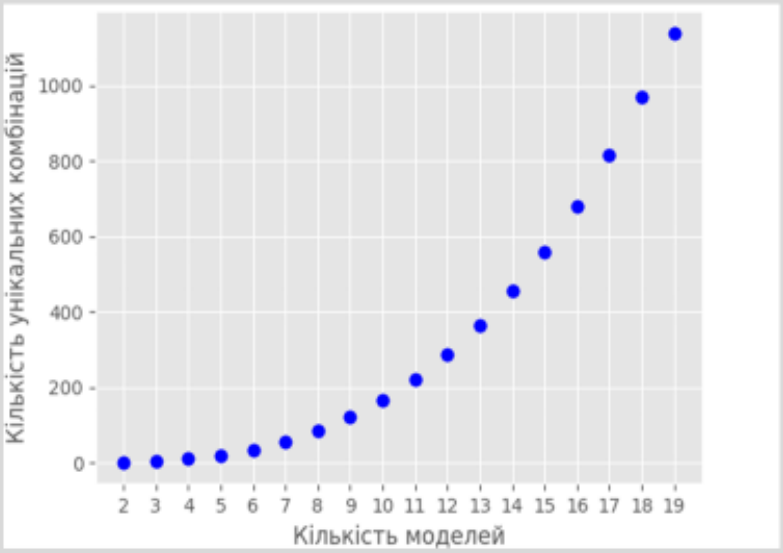
Схема навчання та валідації алгоритму МН



Процес створення наборів ансамблів комбінаторним методом



Залежність кількості базових моделей класифікації до кількості створених унікальних ансамблів



Функція залежності кількості примітивів (x) до кількості унікальних комбінацій (y)

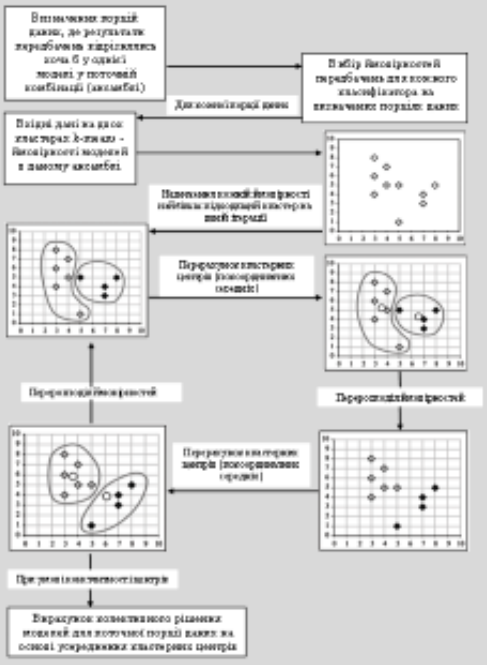
```
def y_function(x):
    y = 0
    for i in range(0, x):
        for j in range(i + 1, x):
            for k in range(j, x):
                y += 1
    return y
```

Утворені унікальні набори моделей класифікації

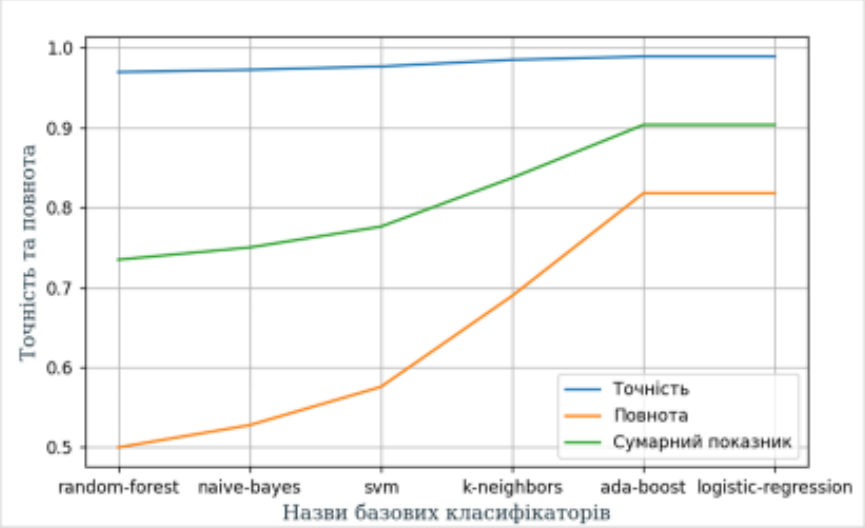
1) ada-boost|random-forest;
2) svm|ada-boost;
3) naive-bayes|ada-boost;
4) svm|naive-bayes|k-neighbors|ada-boost;
5) svm|naive-bayes|k-neighbors|ada-boost|logistic-regression;
6) naive-bayes|k-neighbors|ada-boost;
7) svm|k-neighbors;
8) svm|naive-bayes|k-neighbors|ada-boost|random-forest;
9) svm|naive-bayes|ada-boost;
10) naive-bayes|k-neighbors|ada-boost|random-forest;
11) naive-bayes|k-neighbors|ada-boost|logistic-regression;
12) svm|logistic-regression;
13) svm|naive-bayes|k-neighbors|ada-boost|random-forest|logistic-regression;
14) naive-bayes|k-neighbors;
15) svm|random-forest;
16) svm|naive-bayes|k-neighbors;
17) svm|naive-bayes|k-neighbors|logistic-regression;
18) k-neighbors|ada-boost;
19) k-neighbors|ada-boost|random-forest|logistic-regression;
20) ada-boost|random-forest|logistic-regression;
21) random-forest|logistic-regression;
22) naive-bayes|random-forest;
23) naive-bayes|k-neighbors|random-forest;
24) naive-bayes|k-neighbors|ada-boost|random-forest|logistic-regression;
25) k-neighbors|ada-boost|random-forest;
26) k-neighbors|ada-boost|logistic-regression;
27) svm|naive-bayes;
28) svm|naive-bayes|random-forest;
29) svm|naive-bayes|k-neighbors|random-forest;
30) naive-bayes|logistic-regression;
31) ada-boost|logistic-regression;
32) svm|naive-bayes|logistic-regression;
33) naive-bayes|k-neighbors|logistic-regression;
34) k-neighbors|random-forest;
35) k-neighbors|logistic-regression

Блок-схема алгоритму усереднення результатів на основі k-середніх (k-means)

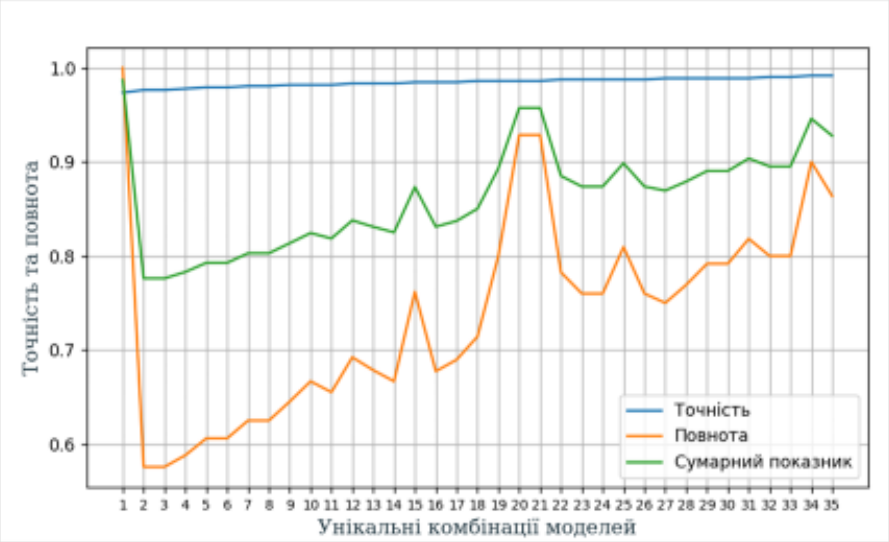
Приклад роботи алгоритму виведення колективного рішення



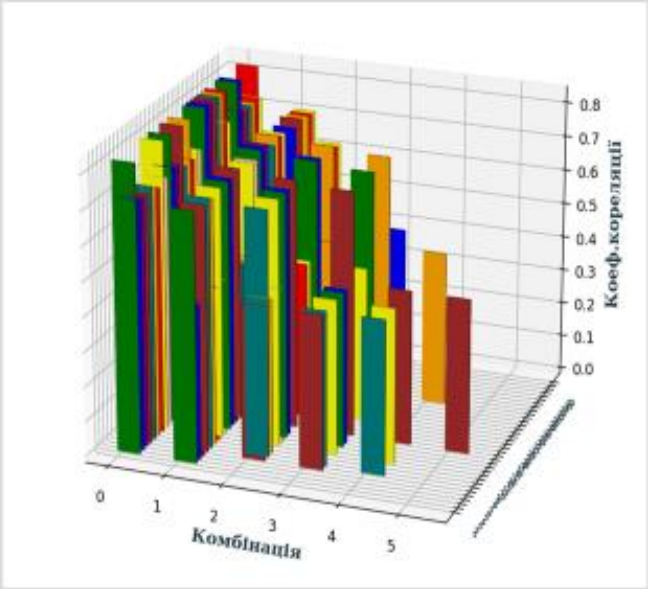
Оцінка якісних показників ефективності базових класифікаторів: кращий результат модель класифікатора «ada-boost» за точністю 0.9889, повнотою 0.8179



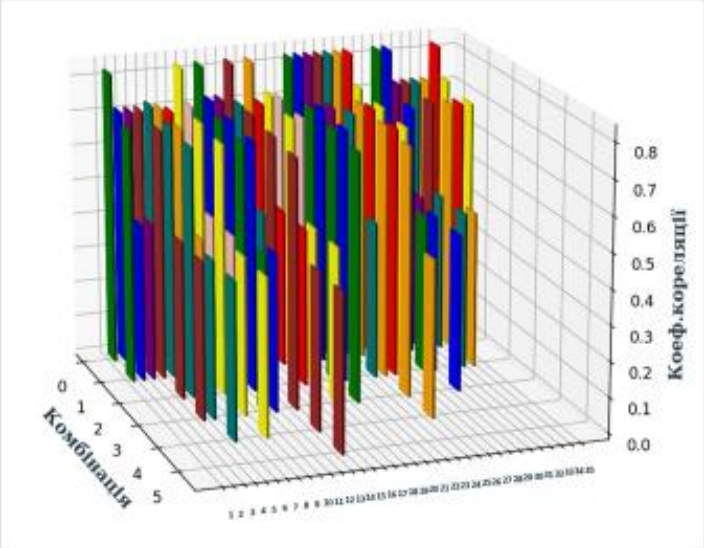
Оцінка якісних показників класифікації комбінацій (наборів) моделей: кращий результат набір «ada-boost, random-forest» за точністю 0.9736, повнотою 1.0



Графіки гістограм коефіцієнтів кореляцій



Графіки гістограм коефіцієнтів кореляцій. Бокова перспектива



Висновки

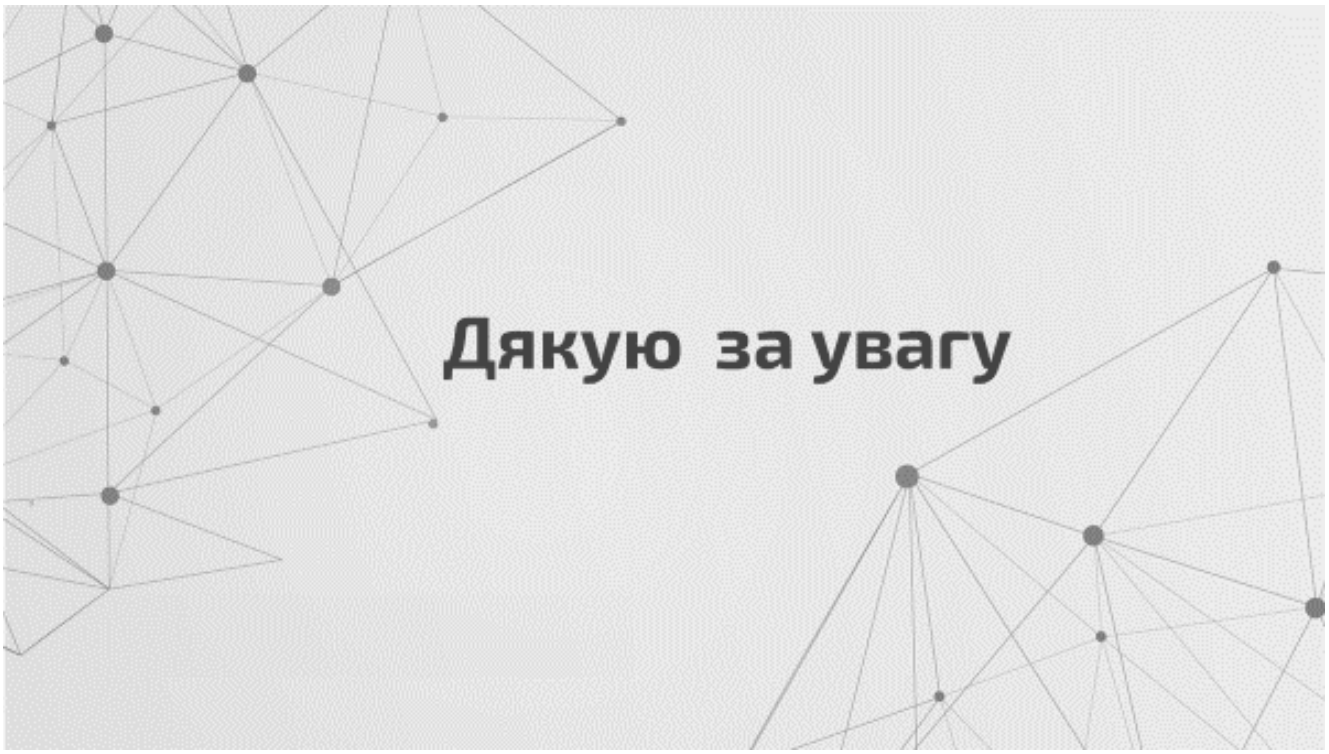
Дослідження ефективногсті запропонованого методу класифікації даних на основі групування рішень моделей ансамблю показало, що класифікаційна система, яка утворена методом **виведення колективного рішення** працює стабільніше та **видає кращі результати** в порівнянні з базовими класифікаторами.

В особливому порядку **покращення набуває повнота** передбачень. За допомогою кореляційних зв'язків між прогнозами моделей у ансамблі та істинними значеннями прослідковується **закономірність схильності** набору моделей до хороших передбачень.

Шляхи вдосконалення

Одним із шляхів вдосконалення реалізованої інформаційної системи класифікації є добавлення можливості **зміни корпусу даних** та варіативність у **методах нормалізації тексту** (винесення в окремий модуль нормалізації даних), що уможливить **конфігурування** стосовно методу підготовки даних.

Наступним шляхом вдосконалення результативності мета-алгоритму є **підбір базових класифікаторів** на основі вирахуваного коефіцієнта кореляції, таким чином, система зможе створювати ансамблі з **найвищими показниками** метрики точності.


Дякую за увагу

Додаток В – Програмні коди

Лістинг № 1 classify_ensemble.py

```
import sys
import time
import os
from copy import deepcopy
import matplotlib.pyplot as plt
import data_generation
import filters
import utils
from data_preprocessor import get_test_docs, get_train_docs
from models_data.classifier_data import Classifier
from kmeans_processor import kmeans_process
from models_data.models import classifiers
from universal_permutator import permutate_universally
from model_evaluator import evaluate_by_important_metrics
from graph_building.correlation_btw_model_and_origin_labels_graph import build_correlation_graphs2d, build_correlation_graphs3d
from graph_building.models_set_accuracy_recall_graph import build_accuracy_recall_graph
from graph_building.base_models_accuracy_recall_graph import build_accuracy_recall_graph_base
from graph_building.combination_dependence_graph import build_combination_dependence_graph

def prediction_function(vectorizer, tfidf_transformer, classifier_data, classifier_name, corpus_test):
    x_test_counts = vectorizer.transform(corpus_test)
    x_test = tfidf_transformer.transform(x_test_counts)

    # Масив передбачень для тестового корпусу даних
    prediction = classifier_data.predict(x_test)
    # Масив ймовірностей передбачень для тестового корпусу даних
    predict_proba = classifier_data.predict_proba(x_test)
    # Вибір істинної ймовірності з масиву, який містить значення ймовірності відношення до topic, non-topic
    filter_proba = [max(pair_value) for pair_value in predict_proba]

    return prediction , filter_proba

def test_and_save(topics, ref_docs, vectorizer,tfidf_transformer, classifier_data, classifier_name):
    start_time = time.time()
    corpus_test = []
    labels = []
    indexed_data_newid = []
    for doc in ref_docs:
        corpus_test.append(doc[1])
        labels.append(doc[0])
        indexed_data_newid.append(doc[3])
    prediction, probabilities = prediction_function(vectorizer,tfidf_transformer, classifier_data,classifier_name, corpus_test)
    print("Testing time of " + classifier_name + " takes %.2f seconds." % (time.time() - start_time))
    status_prediction = []
    count_pred_right = 0
    count_pred_right_topic = 0
    count_pred_right_non_topic = 0
    for i in range(0, len(prediction)):
        if labels[i] == prediction[i]:
            status_prediction.append(1)
            count_pred_right += 1
            if labels[i] == topics[0]:
                count_pred_right_topic += 1
            if labels[i] == topics[1]:
                count_pred_right_non_topic += 1
        else:
            status_prediction.append(0)

    cls = Classifier(
        type_classifier=classifier_name,
        topic=topics[0],
        index_data=indexed_data_newid,
        prediction=prediction,
        result_classified=status_prediction,
        count_pred_right = count_pred_right,
        count_pred_right_topic=count_pred_right_topic,

count_pred_right_non_topic=count_pred_right_non_topic,
        prediction_proba= probabilities,
        labels = labels
    )

    outdir_in = 'test'
```

```
    utils.save_indexed_data_to_file(cls, outdir_in)
    return cls

def train(docs_train, vectorizer, classifier_name, classifier):

    y = [d[0] for d in docs_train]
    corpus = [d[1] for d in docs_train]
    X = vectorizer.transform(corpus)
    print("Training with %s..." % (classifier_name))
    start_time = time.time()
    classifier.fit(X, y)
    print("Training time %.2f seconds." % (time.time() - start_time))
    return classifier



def teach_classifiers(docs_train, vectorizer, topic, classifiers):
    start_time = time.time()
    # Вказуємо labels як topic і non-topic
    topics = [topic, "non-" + topic]
    print("Teaching of classifiers starts\n")
    print("----------", "Labels ", topics[0], topics[1], "----------")
    # Ініціалізація словника класифікаторів
    classifier_name_classifier_data = {classifier_name:classifier_name for classifier_name in classifiers.keys()}
    teached_models = dict()
    for classifier_name, classifier in classifiers.items():
        print("Using %s Classifier for teach\n" % classifier_name)
        classifier_data = train(docs_train,vectorizer, classifier_name, classifier)

        classifier_name_classifier_data[classifier_name] = classifier_data



    teached_models = deepcopy(classifier_name_classifier_data)

    print("Teaching of classifiers takes %.2f seconds.\n" % (time.time() - start_time))
    return teached_models


def test_classifiers(docs_test, vectorizer,tfidf_transformer, topic, classifiers):
    start_time = time.time()
    # створюється контейнер для результатів передбачень класифікаторів
    classifiers_results = []
    # Вказуємо labels як topic і non-topic
    topics = [topic, "non-" + topic]
    print("Testing of classifiers starts\n")
    print("----------", "Labels ", topics[0], topics[1], "----------")
    for classifier_name, classifier in classifiers.items():
        print("Using %s Classifier for test\n" % classifier_name)
        cls = test_and_save(topics, docs_test, vectorizer, tfidf_transformer, classifier, classifier_name)
        classifiers_results.append(cls)
    print("Testing of classifiers takes %.2f seconds.\n" % (time.time() - start_time))
    return classifiers_results



def teach_test_conveyor(docs_filtered, all_topics, k_train_tag,k_test_tag,
                        trained_vectorizer, trained_tfidf_transformer):
    # for topic in all_topics:
    # розмічає данні на два типи 'topic' та 'non-topic'
    docs, count_topic, count_non_topic = filters.labeled_docs_by_topics(all_topics[0], docs_filtered)

    # корпус для тренування
    train_docs = get_train_docs(docs, k_train_tag)
    # корпус для тестування
    test_docs = get_test_docs(docs, k_test_tag)
    # тренування
    if os.path.isfile('dump/list_models.pkl') == False:
        Dict_models = teach_classifiers(train_docs, trained_vectorizer, all_topics[0], classifiers)
        utils.dump_teached_data_save(Dict_models, 'dump/list_models.pkl')
        dict_models = utils.dump_teached_data_load('dump/list_models.pkl')
    else:
        dict_models = utils.dump_teached_data_load('dump/list_models.pkl')
    # тестування
```

```
    classifiers_results = test_classifiers(test_docs,
trained_vectorizer,

trained_tfidf_transformer, all_topics[0], dict_models)
    return classifiers_results



def main(argv):
    start_time = time.time()
    utils.del_directiry('test')
    # завантажує дані, завантажує категорії, завантажує
мітки даних (для навчання і для тренування),
    # відфільтровує дані від пустих значень, вчить
vectorizer, tfidf_transformer на відфільтрованих даних
    types, all_topics, docs_filtered ,
trained_vectorizer , trained_tfidf_transformer =
data_generation.get_data(argv)
    k_train_tag = types[0]
    k_test_tag = types[1]

    # "Навчання і тестування моделей по найбільш
популярним темам"
    classifiers_results =
teach_test_conveyor(docs_filtered , all_topics,
k_train_tag, k_test_tag,
                        trained_vectorizer,
trained_tfidf_transformer)

    original_labels = classifiers_results[0].labels
    every_combination_models_predictions =
permutate_universally(classifiers_results,
kmeans_process_func=kmeans_process)
    every_combination_models_predictions_evaluated =
evaluate_by_important_metrics(original_labels,
every_combination_models_predictions)

build_correlation_graphs2d(every_combination_models_pr
edictions_evaluated)
    #
build_correlation_graphs3d(every_combination_models_pr
edictions_evaluated)
    #
build_accuracy_recall_graph(every_combination_models_p
redictions_evaluated)
    #
build_accuracy_recall_graph_base(classifiers_results)
    # build_combination_dependence_graph()
    plt.show()
    print("Total runtime %.2f seconds.\n" % (time.time()
- start_time))


if __name__ == "__main__":
    main(sys.argv[0:])
```

Лістинг № 2 data_generation.py

```
import os
import pickle

from filters import filter_docs_from_voids
from data_preprocessor import
get_vectorizer_tfidf_transformer
from reuters_parser import ReutersParser

def get_most_important_topics():
    topics = open("data/most_popular_topics.txt",
"r").readlines()
    topics = [t.strip() for t in topics]
    for i in range(0, len(topics)):
        topics[i] = topics[i].lower()
    return topics

def obtain_topic_tags():
    """
    Open the topic list file and import all of the topic
names
    taking care to strip the trailing "\n" from each
word.
    """
    topics = open("data/all-topics-strings.lc.txt",
"r").readlines()
    topics = [t.strip() for t in topics]
    for i in range(0, len(topics)):
        topics[i] = topics[i].lower()
    return topics

def obtain_train_test_tags():
    types = open("data/all-cgisplit.txt",
"r").readlines()
    types = [t.strip() for t in types]
    for i in range(0, len(types)):
        types[i] = types[i].lower()
    return types

def load_data():
```

```
    # Create the list of Reuters data and create the
parser
    if os.path.isfile('dump/docs_parsed.pkl'):
        input_f = open('dump/docs_parsed.pkl', 'rb')
        docs = pickle.load(input_f)
        input_f.close()
    else:
        #                               joblib.dump(svm,
"saved_data/saved_trained_data.pkl")
        # files = ["data/reut2-%03d.sgm" % r for r in
range(0, 22)]
        files = ["data/reut2-%03d.sgm" % r for r in
range(9, 16)]
        parser = ReutersParser()
        # Parse the document and force all generated
docs into
        # a list so that it can be printed out to the
console
        print("Parsing training data...\n")
        docs = []
        for fn in files:
            for d in parser.parse(open(fn, 'rb')):
                docs.append((d[0],     d[1],     d[2],
int(d[3])))

        output = open('dump/docs_parsed.pkl', 'wb')
        pickle.dump(docs, output)
        output.close()
    return docs


def get_data(argv):
    # завантажує дані
    all_docs = load_data()
    # завантажує категорії
    all_topics = get_most_important_topics()
    #  завантажує мітки даних (для навчання і для
тренування)
    types = obtain_train_test_tags()
    # відфільтровуєм дані від пустих значень
    docs_filtered    =    filter_docs_from_voids(types,
all_docs)
    #    вчить    vectorizer,    tfidf_transformer    на
відфільтрованих даних
    trained_vectorizer,   trained_tfidf_transformer   =
get_vectorizer_tfidf_transformer(docs_filtered, argv)

    return    types,    all_topics,    docs_filtered    ,
trained_vectorizer , trained_tfidf_transformer
```

Лістинг № 3 data_preprocessor.py

```
from           stemmed_tfidfvectorizer           import
StemmedTfidfVectorizer
from       sklearn.feature_extraction.text       import
TfidfVectorizer
from       sklearn.feature_extraction.text       import
TfidfTransformer


def get_train_docs(docs, train_tag):
    docs_train = [doc for doc in docs if doc[2].decode()
== train_tag]
    return docs_train

def get_test_docs(docs, test_tag):
    docs_test = [doc for doc in docs if doc[2].decode()
== test_tag]
    return docs_test

def get_vectorizer_tfidf_transformer(docs, argv):
    """
    Creates a document corpus list (by stripping out the
    class labels), then applies the TF-IDF transform to
this
    list.

    The function returns both the class label vector (y)
and
    the corpus token/feature matrix (X).
    """
    # фільтрується списки за типом запису, залишаються
тільки ті, які відповідають doc_type
    # docs = [doc for doc in docs if doc[2].decode() ==
doc_type]
    # Create the training data class labels
    # створюється масив міток чи відноситься до теми чи
ні


    # Create the document corpus list
    # створюється масив документів без міток довжина
відповідає масиву y.
    corpus = [d[1] for d in docs]


    # Create the TF-IDF vectoriser and transform the
corpus
    if len(argv) == 1:
        print("\nWith remove stopwords and stemming")
        vectorizer                                      =
StemmedTfidfVectorizer(stop_words="english", \

analyzer="word", \
                                                min_df=1)
    if len(argv) > 1:
```

```
        if "--no-stemming" == argv[1]:
            print("Without stemming")
            vectorizer                                  =
TfidfVectorizer(stop_words="english", \

analyzer="word", \
                                        min_df=1)
        else:
            if "--no-stopwords" == argv[1]:
                print("Without removing stopwords")
                vectorizer                              =
TfidfVectorizer(analyzer="word", min_df=1)
            else:
                print("With   remove   stopwords   and
stemming")
                vectorizer                              =
StemmedTfidfVectorizer(stop_words="english", \

analyzer="word", \

min_df=1)

    x = vectorizer.fit_transform(corpus)
    tfidf_transformer = TfidfTransformer()
    x_train_tfidf = tfidf_transformer.fit_transform(x)

    return vectorizer, tfidf_transformer
```

Лістинг № 4 filters.py

```
def labeled_docs_by_topics(topic, docs):
    # розмічає дані на два типи label
    #  ті,  які  відносяться  до  теми  і  ті,  які  не
відносяться
    count_topic = 0
    count_non_topic = 0
    ret_docs = []
    for doc in docs:
        if len(doc) < 3:
            continue
        label = "non-" + topic
        for topic_entry in doc[0]:
            if topic_entry == topic:
                label = topic
                count_topic += 1
                break
        ret_docs.append((label,     doc[1],     doc[2],
doc[3]))
    count_non_topic = len(docs) - count_topic
    return ret_docs, count_topic, count_non_topic


def filter_docs_from_voids(types, docs):
    # відфільтровуєм від пустих значень
    # d[1] == ''
    # відфільтровує ті, які не мають незви теми (topic),
    # d[0] == ""
    #  відфільтровує  ті,  в  яких  тип  не  відповідає
заданому типу (types) (для тренування чи перевірки-
тестування)
    # d[2].decode() not in types

    ret_docs = []
    for d in docs:
        if d[0] == [] or d[0] == "":
            continue
        if d[1] == '':
            continue
        if d[2].decode() not in types:
            continue

        ret_docs.append((d[0], d[1], d[2], d[3]))

    return ret_docs
```

Лістинг № 5 kmeans_process.py

```
from sklearn.cluster import KMeans
from     models_data.models_combination_data     import
ModelsSetData


def
select_non_intersectioned_predictions(classifiers_resu
lts):
    non_intersection_predictions_dict                =
{classifier.type_classifier: list() for classifier in
classifiers_results}
    non_intersection_probabilities_dict              =
{classifier.type_classifier: list() for classifier in
classifiers_results}
    # Запис результатів класифікаторів в dictionary
    predictions_dict   =   {classifier.type_classifier:
classifier.prediction      for      classifier      in
classifiers_results}
    probabilities_dict  =  {classifier.type_classifier:
classifier.prediction_proba    for    classifier    in
classifiers_results}
    # контейнер для id документів, передбачення над
якими не збіглися у моделей
    non_intersection_document_id = []
```

```
    # контейнер істинних відповідей, містить тільки ті елементи,
    # індекси яких такі ж як у id документів, передбачення над якими не збіглися у моделей
    non_intersection_original_labels = []
    results_count = len(classifiers_results[0].result_classified)
    for i in range(0, results_count):
        one_result_of_classifiers = []
        for results in predictions_dict.values():
            one_result_of_classifiers.append(results[i])
        if len(set(one_result_of_classifiers)) == 1:
            continue
        else:
            non_intersection_document_id.append(classifiers_results[0].index_data[i])
            non_intersection_original_labels.append(classifiers_results[0].labels[i])
            for classifier_name in non_intersection_predictions_dict.keys():
                non_intersection_predictions_dict[classifier_name].append(predictions_dict.get(classifier_name)[i])
                non_intersection_probabilities_dict[classifier_name].append(probabilities_dict.get(classifier_name)[i])

    return distribute_probabilities(non_intersection_predictions_dict,non_intersection_probabilities_dict),\
           non_intersection_document_id, non_intersection_original_labels


def distribute_probabilities(non_intersection_predictions_dict, non_intersection_probabilities_dict):
    # розділяємо ймовірності на додатні та від'ємні для відповідних класів (non-topic: -;topic: +;)
    predictions_container = [predictions for predictions in non_intersection_predictions_dict.values()]
    probabilities_container = [probabilities for probabilities in non_intersection_probabilities_dict.values()]

    sub_str = "non"
    for i in range(0, len(probabilities_container[0])):
        for j in range(0, len(probabilities_container)):
            if sub_str in predictions_container[j][i]:
                probabilities_container[j][i] = -1 * probabilities_container[j][i]
    return non_intersection_probabilities_dict


def kmeans_process(classifiers_results):
    # створюємо модель для кластеризації даних
    model = KMeans(n_clusters=2, random_state=0)
    # контейнер для передбачень класифікаторів, які не пересіклись
    probabilities_container_dict, documents_id_container, original_labels_container = select_non_intersectioned_predictions(classifiers_results)
    # створюємо контейнер для центроїдів
    centroids_container = [[], []]
    probabilities_container = [probabilities_lst for probabilities_lst in probabilities_container_dict.values()]
    for i in range(0, len(probabilities_container[0])):
        tmp_prediction = []
        cluster_centers = []
        index = 0
        for prediction in probabilities_container:
            tmp_prediction.append([])
            tmp_prediction[index].append(prediction[i])
            index += 1

        model.fit(tmp_prediction)
        cluster_centers = model.cluster_centers_
        centroids_container[0].append(cluster_centers[0][0])
        centroids_container[1].append(cluster_centers[1][0])

    # контейнер, в якому зберігаються всі передбачення однієї з моделей
    # пізніше в ньому заміняться індекси тих передбачень, які були однакові
    # у результуючому наборі наборі моделей.
    # значення передбачень на замінюваних індексах будуть оптимізовані згідно кожного набору моделей
    raw_prediction = classifiers_results[0].prediction
    index_data = classifiers_results[0].index_data
    indexed_raw_prediction_data = dict()
    for i in range(0, len(index_data)):
```

```
        indexed_raw_prediction_data[index_data[i]]    =
raw_prediction[i]

    indexed_model_combination_prediction_dict        =
determine_prediction_by_centroids_average(centroids_co
ntainer,

classifiers_results[0].topic,

documents_id_container)
    optimized_predictions_dict                       =
replace_raw_prediction(indexed_raw_prediction_data,
indexed_model_combination_prediction_dict)
    models_names   =   [classifier.type_classifier   for
classifier in classifiers_results]
    models_set = ModelsSetData(
        classifiers_names= models_names,
        non_intersection_document_id=
documents_id_container,
        probabilities_container_dict                 =
probabilities_container_dict,

optimized_predictions=optimized_predictions_dict,
        original_labels=original_labels_container
    )
    return models_set


def replace_raw_prediction(indexed_raw_prediction_data,
indexed_model_combination_prediction_dict):
    optimized_prediction = dict()
    for            key,            value            in
indexed_raw_prediction_data.items():
        optimized_prediction[key] = value
        for            k,            v            in
indexed_model_combination_prediction_dict.items():
            if (k == key):
                optimized_prediction[key] = v

    return optimized_prediction



def calculate_centroids_average(centroids_container):
    centroids_average = []
    for i in range(0, len(centroids_container[0])):
        sum = 0
        for centroids_list in centroids_container:
            sum += centroids_list[i]
        centroids_average.append(sum                 /
len(centroids_container))
    return centroids_average


def      distribute_topic_non_topic(centroids_average,
topic):
    # створюється контейнер, в якому будуть записані
відповіді topic-non-topic
    # згідно середнього значення центроїдів ймовірностей
передбачень для окремої вибірки моделей
    topic_non_topic_lst = []
    for i in centroids_average:
        if (i <= 0):
            topic_non_topic_lst.append('non-' + topic)
        else:
            topic_non_topic_lst.append(topic)
    return topic_non_topic_lst



def
determine_prediction_by_centroids_average(centroids_co
ntainer, topic, documents_id_container):
    centroids_average                                =
calculate_centroids_average(centroids_container)
    model_combination_prediction                     =
distribute_topic_non_topic(centroids_average, topic)
    indexed_model_combination_prediction_dict = dict()
    for i in range(0, len(documents_id_container)):


indexed_model_combination_prediction_dict[documents_id
_container[i]] = model_combination_prediction[i]

    return indexed_model_combination_prediction_dict
```

Лістинг № 6 model_evaluator.py

```
from     sklearn.metrics     import     accuracy_score,
confusion_matrix

def     evaluate_by_important_metrics(originals_labels,
every_combination_models_predictions):
    every_combination_models_predictions_evaluated    =
dict()
    for           key,           value_obj           in
every_combination_models_predictions.items():
        tmp_predictions = []
        for                    v                    in
value_obj.optimized_predictions.values():
            tmp_predictions.append(v)
```

```
        accuracy                                    =
accuracy_score(y_true=originals_labels,         y_pred=
tmp_predictions)
        #     recall     =     recall_score(y_true     =
originals_labels,     y_pred     =     tmp_predictions,
average=None)
        confusion_m                                 =
confusion_matrix(y_true=originals_labels,       y_pred=
tmp_predictions)
        value_obj.accuracy = round(accuracy, 4)
        value_obj.recall  =  round(confusion_m[0][0]  /
(confusion_m[0][0] + confusion_m[1][0]), 4)
        value_obj.confusion_matrix = confusion_m

every_combination_models_predictions_evaluated[key]   =
value_obj

    sorted_dict     =     {k:v     for     k,v     in
sorted(every_combination_models_predictions_evaluated.
items(),
                                        key  =  lambda
item: item[1].accuracy)}
    return  sorted_dict
```

Лістинг № 7 reuters_parser.py

```
import re

from html.parser import HTMLParser




class ReutersParser(HTMLParser):
    """
    ReutersParser subclasses HTMLParser and is used to
open the SGML
    files associated with the Reuters-21578 categorised
test collection.

    The parser is a generator and will yield a single
document at a time.
    Since the data will be chunked on parsing, it is
necessary to keep
    some internal state of when tags have been "entered"
and "exited".
    Hence the in_body, in_topics and in_topic_d boolean
members.
    """
    def __init__(self, encoding='latin-1'):
        """
        Initialise the superclass (HTMLParser) and reset
the parser.
        Sets the encoding of the SGML files by default
to latin-1.
        """
        HTMLParser.__init__(self)
        self._reset()
        self.encoding = encoding

    def _reset(self):
        """
        This is called only on initialisation of the
parser class
        and when a new topic-body tuple has been
generated. It
        resets all off the state so that a new tuple can
be subsequently
        generated.
        """
        self.in_body = False
        self.in_topics = False
        self.in_topic_d = False
        self.in_reuters = False
        self.body = ""
        self.topics = []
        self.topic_d = ""
        self.reuters = ""
        self.cgisplit = ""
        self.newid = ""


    def parse(self, fd):
        """
        parse accepts a file descriptor and loads the
data in chunks
        in order to minimise memory usage. It then
yields new documents
        as they are parsed.
        """
        self.docs = []
        for chunk in fd:
            self.feed(chunk.decode(self.encoding))
            for doc in self.docs:
                yield doc
            self.docs = []
        self.close()


    def handle_starttag(self, tag, attrs):
        """
        This method is used to determine what to do when
the parser
```

```
        comes across a particular tag of type "tag". In
this instance
        we simply set the internal state booleans to
True if that particular
        tag has been found.
        """
        if tag == "reuters":
            #pass
            self.in_reuters = True
            for attribute in attrs:
                if attribute[0] == "cgisplit":
                    self.cgisplit                    =
attribute[1].encode("utf-8").lower()
                if attribute[0] == "newid":
                    self.newid = attribute[1]
                    #break
        elif tag == "body":
            self.in_body = True
        elif tag == "topics":
            self.in_topics = True
        elif tag == "d":
            self.in_topic_d = True

    def handle_endtag(self, tag):
        """
        This method is used to determine what to do when
the parser
        finishes with a particular tag of type "tag".

        If the tag is a  tag, then we remove all
        white-space with a regular expression and then
append the
        topic-body tuple.

        If the tag is a  or  tag then we simply set
        the internal state to False for these booleans,
respectively.

        If the tag is a  tag (found within a  tag), then
we
        append the particular topic to the "topics" list
and
        finally reset it.
        """
        if tag == "reuters":
            self.body = re.sub(r'\s+', r' ', self.body)
            self.in_reuters = False
            self.docs.append( (self.topics, self.body,
self.cgisplit, self.newid) )
            self._reset()
        elif tag == "body":
            self.in_body = False
        elif tag == "topics":
            self.in_topics = False
        elif tag == "d":
            self.in_topic_d = False
            self.topics.append(self.topic_d)
            self.topic_d = ""

    def handle_data(self, data):
        """
        The data is simply appended to the appropriate
member state
        for that particular tag, up until the end
closing tag appears.
        """
        if self.in_body:
            #self.body += data.encode("utf-8")
           self.body += data
        elif self.in_topic_d:
         #   self.topic_d += data.encode("utf-8")
         #   self.topic_d += data.encode(encoding="utf-
8")
            self.topic_d += data
```

Лістинг№8stemmed_tfidfvectorizer.py

```
import nltk.stem
from       sklearn.feature_extraction.text       import
TfidfVectorizer

class StemmedTfidfVectorizer(TfidfVectorizer):
    def build_analyzer(self):
        english_stemmer                                =
nltk.stem.SnowballStemmer('english')
        analyzer                                       =
super(TfidfVectorizer,self).build_analyzer()
        return lambda doc: (english_stemmer.stem(w) for
w in analyzer(doc))
```

Лістинг№9 universal_permutator.py

```
from copy import copy

def     permutate_universally(classifiers_results     ,
kmeans_process_func):
    # стоворимо словник для запису відповідей для кожної
комбінації моделей
    every_combination_models_predictions = dict()
    # створення всіх можливих універсальних комбінацій
результатів класифікаторів
```

```
    for i in range(0, len(classifiers_results)):
        tmp_lst = []
        tmp_lst.append(classifiers_results[i])
        for j in range(i + 1, len(classifiers_results)):
            tmp_lst.append(classifiers_results[j])
            for k in range(j, len(classifiers_results)):
                copy_tmp = copy(tmp_lst)
                copy_tmp[j - i] = classifiers_results[k]
                # назви моделей на поточній комбінації
                combination_set = [model.type_classifier for model in copy_tmp]
                combination_set_str = '|'.join(combination_set)

every_combination_models_predictions[combination_set_str] = kmeans_process_func(copy_tmp)



    return every_combination_models_predictions
```

Лістинг№10 utils.py

```
import os
import shutil
import pickle
from pathlib import Path



def dump_teached_data_load(path_load):
    input_f = open(path_load, 'rb')
    list_cls = pickle.load(input_f)
    input_f.close()
    return list_cls

# запис результатів навчання в dump
def dump_teached_data_save(list_cls, path_save):
    output = open(path_save, 'wb')
    pickle.dump(list_cls, output)
    output.close()

# видалення директорій з файлами в них
def del_directiry(outdir_in):
    currentdir = os.getcwd()
    outdir = 'datesaved'
    #outdir_in = 'tmp'
    if os.path.isdir(outdir) == True:
        if os.path.isdir(outdir + '/' + outdir_in) == True:
            shutil.rmtree(outdir + '/' + outdir_in, ignore_errors=True)

# запис результатів тестування в файли по категоріям
def save_indexed_data_to_file(cls, outdir_in):
    indent = 20
    print("Saved to file...")
    first_mode_dir = os.getcwd()
    currentdir = os.getcwd()
    outdir = 'datesaved'
    # outdir_in = 'test'
    parentdir = Path(currentdir).stem
    if parentdir != outdir_in:
        if os.path.isdir(outdir) == False:
            os.mkdir(outdir)
        if os.path.isdir(outdir + '/' + outdir_in) == False:
            os.mkdir(outdir + '/' + outdir_in)
        os.chdir(currentdir + '/' + outdir + '/' + outdir_in)
    dir = cls.type_classifier
    if os.path.isdir(dir) == False:
        os.mkdir(dir)
    file_name = dir + '/' + cls.topic + '.txt'
    outfile = open(file_name, 'w', encoding='utf8')
    outfile.write(cls.type_classifier + '\n')
    outfile.write(cls.topic + '\n')
    outfile.write(str(cls.index_data) + '\n')
    outfile.write(str(cls.prediction_proba) + '\n')
    outfile.write(str(cls.result_classified) + '\n')
    outfile.close()
    os.chdir(first_mode_dir)
```

Лістинг№11 classifier_data.py

```
import numpy as np
from sklearn.metrics import accuracy_score, confusion_matrix

class Classifier():
    def __init__(self, type_classifier, topic, index_data, prediction,
                 result_classified, count_pred_right, count_pred_right_topic,
                 count_pred_right_non_topic, prediction_proba, labels):
        self.type_classifier = type_classifier
        self.topic = topic
        self.index_data = index_data
        self.prediction = prediction
```

```
        self.result_classified = result_classified
        self.count_pred_right = count_pred_right
        self.count_pred_right_topic                    =
count_pred_right_topic
        self.count_pred_right_non_topic                =
count_pred_right_non_topic
        self.prediction_proba = prediction_proba
        self.labels = labels
        self.accuracy                                  =
round(accuracy_score(y_true=labels,             y_pred=
prediction), 4)
        self.confusion_matrix                          =
np.array(confusion_matrix(y_true=labels,        y_pred=
prediction)) + 0.01
        self.recall = round(self.confusion_matrix[0][0]
/             (self.confusion_matrix[0][0]             +
self.confusion_matrix[1][0]), 4)
```

Лістинг№12 models.py

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from  sklearn.ensemble  import  RandomForestClassifier,
AdaBoostClassifier
from sklearn.linear_model import LogisticRegression

classifiers = \
    {
        "svm":         SVC(C=1.0,         gamma='auto',
kernel='linear',               class_weight='balanced',
probability=True),
        "naive-bayes": MultinomialNB(alpha=0.01),
        "k-neighbors":
KNeighborsClassifier(n_neighbors=30,
weights='uniform'),
        "ada-boost": AdaBoostClassifier(),
        "random-forest":
RandomForestClassifier(n_estimators=200,   max_depth=3,
random_state=0),
        "logistic-regression": LogisticRegression()
    }
```

Лістинг№13 models_combination_data.py

```
class ModelsSetData():
  def         __init__(self,         classifiers_names,
non_intersection_document_id,probabilities_container_d
ict,
                optimized_predictions,
original_labels):
        self.classifiers_names = classifiers_names
        self.itersection_document_id                   =
non_intersection_document_id
        self.probabilities_container_dict              =
probabilities_container_dict
        self.optimized_predictions                     =
optimized_predictions
        self.original_labels = original_labels
        self.accuracy = 0
        self.recall = 0
        self.confusion_matrix = 0
        probabilities                                  =
probabilities_container_dict.values()
        sum = 0
        count = 0
        for i in probabilities:
              for j in i:
                  sum+=j
                  count+=1
        self.average_prob = round(sum / count, 3)
```

Лістинг№14 base_models_accuracy_recall_graph.py

```
import matplotlib.pyplot as plt

def
build_accuracy_recall_graph_base(classifiers_results):
    sorted_cls    =    sorted(classifiers_results    ,
key=lambda x: x.accuracy, reverse=False)
    accuracies = []
    recalls = []
    models_names = []
    accuracy_recall_averages = []
    for classifier in sorted_cls:
        accuracies.append(classifier.accuracy)
        recalls.append(classifier.recall)

models_names.append(classifier.type_classifier)

accuracy_recall_averages.append((classifier.recall    +
classifier.accuracy) / 2)
    font = {'family': 'serif',
            'color': '#233842',
            'weight': 'normal',
            'size': 12,
            }

    fig = plt.figure(figsize=(5, 3), dpi=100)
    ax = fig.add_subplot(111)
    ax.plot(models_names, accuracies, label='Точність')
```

```
    ax.plot(models_names, recalls, label='Повнота')
    ax.plot(models_names,     accuracy_recall_averages,
label='Сумарний показник')
    ax.tick_params(axis='x', rotation=90, labelsize=6)
    ax.set_xlabel('Комбінації моделей', fontdict=font)
    ax.set_ylabel('Точність та повнота', fontdict=font)
    ax.set_title('Точність та повнота базових моделей',
fontdict=font)
    ax.text(3, 0.95,
            f'Найкраща
модель:\n{models_names[accuracy_recall_averages.index(
max(accuracy_recall_averages))]}'

f'\nТочність:{accuracies[accuracy_recall_averages.inde
x(max(accuracy_recall_averages))]},            Повнота:
{recalls[accuracy_recall_averages.index(max(accuracy_r
ecall_averages))]}'
            , style='italic', fontsize=8,
            bbox={'facecolor':  'red',  'alpha':  0.5,
'pad': 5})

    ax.legend()
    ax.grid()
```

Лістинг№15 combination_dependence_graph.py

```
import matplotlib.pyplot as plt

def y_function(x):
    y = 0
    for i in range(0, x):
        for j in range(i + 1, x):
            for k in range(j, x):
                y += 1
    return y

def build_combination_dependence_graph():
    font = {'family': 'serif',
            'color': '#233842',
            'weight': 'normal',
            'size': 12,
            }
    models_count_lst = range(2, 20)
    combinations_count_lst  =  [y_function(count)  for
count in models_count_lst]
    plt.style.use('ggplot')
    fig = plt.figure(figsize=(5, 3), dpi=100)
    ax = fig.add_subplot(111)
    ax.plot(models_count_lst,   combinations_count_lst,
'bo')
    ax.set_xlabel('Кількість моделей')
    ax.set_ylabel('Кількість унікальних комбінацій')
    ax.set_title('Залежність кількості комбінацій від
кількості моделей', fontdict=font)
    ax.plot(models_count_lst,   combinations_count_lst,
'bo')
    ax.set_xticks(models_count_lst)


def main():
    build_combination_dependence_graph()

if __name__ == "__main__":
    main()
```

Лістинг№16 correlation_btw_model_and_origin_labels_graph.py

```
import random
import matplotlib.pyplot as plt
import tkinter as tk
from scipy.stats import pearsonr
from pandas import DataFrame
from tkinter import *
from      matplotlib.backends.backend_tkagg      import
FigureCanvasTkAgg
from matplotlib.patches import Rectangle


def replace_original_labels(original_labels_dict):
    numeric_original_labels = dict()
    sub_str = "non"
    for      models_names,      original_labels      in
original_labels_dict.items():
        numeric_original_labels_tmp = []
        for label in original_labels:
            n = round(random.uniform(0, 0.1), 3)
            if sub_str in label:
                numeric_original_labels_tmp.append(-1 -
n)
            else:
                numeric_original_labels_tmp.append(1  +
n)
        numeric_original_labels[models_names]        =
numeric_original_labels_tmp
    return numeric_original_labels

def
calculate_correlation(every_combination_models_predict
ions_evaluated):
    original_labels                                  =
{models_name:models_set_obj.original_labels
```

```
                    for models_name, models_set_obj in
every_combination_models_predictions_evaluated.items()
}
    numeric_original_labels                          =
replace_original_labels(original_labels)
    probabilities_of_each_model_in_set               =
[models_set_obj.probabilities_container_dict       for
models_set_obj in

every_combination_models_predictions_evaluated.values(
)]

correlation_coef_btw_each_model_in_set_and_original_la
bels = dict()
    index = 0
    for     models_names,     numeric_labels     in
numeric_original_labels.items():
        correlation_coef_dict_tmp = dict()
        if(len(numeric_labels) < 2):
            index += 1
            continue
        for   model_name   ,   model_probabilities   in
probabilities_of_each_model_in_set[index].items():
            coef      =      pearsonr(numeric_labels,
model_probabilities)[0]
            correlation_coef_dict_tmp[model_name]    =
round(coef, 3)

correlation_coef_btw_each_model_in_set_and_original_la
bels[models_names] = correlation_coef_dict_tmp
        index += 1
    return
correlation_coef_btw_each_model_in_set_and_original_la
bels

def
build_correlation_graphs2d(every_combination_models_pr
edictions_evaluated):

correlation_coef_btw_each_model_in_set_and_original_la
bels                                                 =
calculate_correlation(every_combination_models_predict
ions_evaluated)
    font = {'family': 'serif',
            'color': '#233842',
            'weight': 'bold',
            'size': 8,
            }
    colors  =  ['green',  'blue',  'purple',  'brown',
'teal', 'orange']
    root = tk.Tk()
    root.title('Кореляція    моделей    з    істинними
значеннями')
    main_frame = Frame(root)
    main_frame.pack(fill=BOTH, expand=1)
    my_canvas = Canvas(main_frame)
    my_canvas.pack(side=LEFT, fill=BOTH, expand=1)
    my_scrollbar        =        Scrollbar(main_frame,
orient=VERTICAL, command=my_canvas.yview)
    my_scrollbar.pack(side=RIGHT, fill=Y)

my_canvas.configure(yscrollcommand=my_scrollbar.set)
    my_canvas.bind('<Configure>',       lambda       e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")
))
    second_frame = Frame(my_canvas)
    my_canvas.create_window((0,                      0),
window=second_frame, anchor='nw')
    def _on_mouse_wheel(event):
        my_canvas.yview_scroll(-1 * int((event.delta
                                       /       120)),
"units")
    my_canvas.bind_all("<MouseWheel>", _on_mouse_wheel)


    for       models_names,       correlation       in
correlation_coef_btw_each_model_in_set_and_original_la
bels.items():
        names_and_corr_dict        =        {'models':
list(correlation.keys()),                        'corr':
list(correlation.values())}
        df       =       DataFrame(names_and_corr_dict,
columns=['models', 'corr'])

        figure = plt.figure(figsize=(5, 3), dpi=100)
        ax = figure.add_subplot(111)
        ax.set_title(models_names, fontdict=font)
        ax.set_xlabel('Моделі',          fontdict=font,
fontsize=12)
        ax.set_ylabel('Коеф.   кореляції   між   кожною
моделлю в наборі і оригіналом', fontdict=font)

        bar = FigureCanvasTkAgg(figure, second_frame, )
        bar.get_tk_widget().pack()

        df               =               df[['models',
'corr']].groupby('models').sum()
        df        =        df.sort_values(by=['corr'],
ascending=True)
        df.plot(kind='bar',      ax=ax,      grid=True,
legend=False, rot=0, fontsize=8, width=0.4)
```

```
        index = 0
        for ch in ax.get_children():
            if isinstance(ch, Rectangle) and ch.xy !=
(0, 0):
                ch.set_color(colors[index])
                index += 1

    root.mainloop()

def
build_correlation_graphs3d(every_combination_models_pr
edictions_evaluated):

correlation_coef_btw_each_model_in_set_and_original_la
bels = calculate_correlation(

every_combination_models_predictions_evaluated)
    font = {'family': 'serif',
            'color': '#233842',
            'weight': 'bold',
            'size': 12,
            }
    colors = ['green', 'blue', 'purple', 'brown',
'teal', 'orange', 'red', 'yellow', 'pink']
    colors3d = []
    fig = plt.figure()
    ax = fig.add_subplot(111, projection = '3d')
    ax.figure.set_size_inches(7, 6)
    ax.set_xlabel('Комбінація', fontdict=font)
    ax.set_zlabel('Коеф.кореляції', fontdict=font)
    ax.set_title('Коеф. кореляції між кожною моделлю в
наборі і оригіналом', fontdict=font)
    x = []
    y = []
    y_labels = []
    z = []
    dx = []
    dy = []
    dz = []

    y_pos = 0
    color_index = 0
    for models_names, correlation in
correlation_coef_btw_each_model_in_set_and_original_la
bels.items():
        y_labels.append(models_names)
        x_pos = 0
        if(color_index == len(colors)):
            color_index = 0
        for corr in correlation.values():
            colors3d.append(colors[color_index])
            x.append(x_pos)
            y.append(y_pos)
            z.append(0)
            dx.append(0.4)
            dy.append(0.8)
            dz.append(corr)
            x_pos+=1
        y_pos+=5
        color_index+=1

    ax.set_yticks(range(0, len(y_labels) * 5, 5))
    ax.set_yticklabels(y_labels, fontsize=5, weight =
'bold', rotation = 90, color = '#233842')
    ax.bar3d(x, y, z, dx, dy, dz, color = colors3d)
```

Лістинг№17 models_set_accuracy_recall_graph.py

```
import matplotlib.pyplot as plt


def
build_accuracy_recall_graph(every_combination_models_p
redictions_evaluated):
    accuracies = []
    recalls = []
    models_set_names = []
    accuracy_recall_averages = []
    for set_obj in
every_combination_models_predictions_evaluated.values(
):
        accuracies.append(set_obj.accuracy)
        recalls.append(set_obj.recall)

models_set_names.append(str(set_obj.classifiers_names)
)

accuracy_recall_averages.append((set_obj.accuracy +
set_obj.recall) / 2)
    font = {'family': 'serif',
            'color': '#233842',
            'weight': 'normal',
            'size': 12,
            }

    fig = plt.figure(figsize=(5, 3), dpi=100)
    ax = fig.add_subplot(111)
    ax.plot(models_set_names, accuracies, label =
'Точність')
    ax.plot(models_set_names, recalls, label =
'Повнота')
    ax.plot(models_set_names,
accuracy_recall_averages, label='Сумарний показник')
```

```
    ax.tick_params(axis='x', rotation=90, labelsize =
6)
    ax.set_xlabel('Комбінації моделей' , fontdict=font)
    ax.set_ylabel('Точність та повнота' ,
fontdict=font)
    ax.set_title('Точність та повнота кожного набору
моделей', fontdict=font)
    ax.text(19, 0.98, f'Найкращий
набір:\n{models_set_names[accuracy_recall_averages.ind
ex(max(accuracy_recall_averages))]}'

f'\nТочність:{accuracies[accuracy_recall_averages.inde
x(max(accuracy_recall_averages))]}, Повнота:
{recalls[accuracy_recall_averages.index(max(accuracy_r
ecall_averages))]}'
            , style='italic',fontsize = 8,
            bbox={'facecolor': 'red', 'alpha': 0.5,
'pad': 5})
    ax.legend()
    ax.grid()
```

Додаток Г – Довідка антиплагіату

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 3.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 8%**

	Документ		Сумарний збіг по Базі Даних	
ID: 109054 Назва: КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА на тему Метод класифікації текстових даних на основі групування рішень моделей ансамблю Додано в БД: 2022-12-07 Автора: М.М. Стебелецький Керівники: Е.А. Манзюк Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	67911	1062	3091 (5%)	51 (5%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

Додаток Д – Довідка «Unicheck»

UNICHECK

Ім'я користувача:
Кафедра КН

ID перевірки:
1013230895

Дата перевірки:
07.12.2022 15:04:40 EET

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
07.12.2022 15:05:46 EET

ID користувача:
100005671

Назва документа: **КНм-21-1_Стебелецький**

Кількість сторінок: **73** Кількість слів: **11611** Кількість символів: **86605** Розмір файлу: **2.37 MB** ID файлу: **1012991016**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

11.9%
Схожість

Найбільша схожість: **10.2%** з джерелом з Бібліотеки (ID файлу: **1008302844**)

2.11% Джерела з Інтернету	25	Сторінка 75
10.9% Джерела з Бібліотеки	114	Сторінка 75

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	7
Підозріле форматування	17 сторінок

Додаток Е – Рішення експертної комісії

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНИХ НАУК

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ МАГІСТРА ДО ЗАХИСТУ ЗА РЕЗУЛЬТАТАМИ АНАЛІЗУ ЗВІТУ ПОДІБНОСТІ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *Метод класифікації текстових даних на основі групування рішень моделей ансамблю*

Автор: Стебелецький М.М., група КНм-21-1

Спеціальність: 122 – Комп'ютерні науки

Освітня програма: освітньо-професійна

Науковий керівник: к.т.н., доц. кафедри КН Манзюк Е.А.

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) за програмою Anti-Plagiarism виявлені 3% запозичень, що підтверджує авторство дослідження.

2) За програмою UNICHECK виявлені 11,9% запозичень є фрагментарними – містять поширені конструкції, загальновідомі терміни, скорочення та визначення.

Обсяг запозичень, визначений системами виявлення збігів/ідентичності/ схожості, складає 3% і 11,9% відповідно, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Е. А. Манзюк

Гарант ОП _____ Р. О. Багрій

Завідувач кафедри КН _____ О. В. Бармак

Додаток Є – Відгук опонента

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

кафедра комп'ютерних наук

ВІДГУК
ОПОНЕНТА

на кваліфікаційну роботу магістра

гр. КНм-21-1 Стебелецький Мирослав Миронович за темою: *Метод класифікації текстовихданих на основі групування рішень моделей ансамблю*

1. Актуальність обраної теми

Тема кваліфікаційної робота є актуальною та відповідає сучасному рівню досліджень предметної області. В роботі на належному рівні представлено обґрунтування та проведений огляд досліджень в напрямку обраної теми.

2. Відповідність роботи предметній області спеціальності 122 Комп'ютерні науки та загальним вимогам до наукових робіт

Тема кваліфікаційної роботи та її реалізація відповідає предметній області спеціальності 122 Комп'ютерні науки а також відповідає вимогам до наукових робіт освітньо-кваліфікаційного рівня магістри.

3. Повнота розкриття мети та завдань дослідження

Завдання досліджень розкривають поставлену мету кваліфікаційної роботи та повною мірою представлені в роботі.

4. Наявність наукової новизни

Запропоновані в роботі модель формування ансамблю прийняття рішень та метод виведення колективного рішення на основі групування передбачень моделей ансамблю мають наукову новизну та відповідають кваліфікаційному рівню магістра. Результати дослідження оприлюднені на науковій конференції.

5. Зміст кожного розділу роботи

Робота містить чотири розділи. В першому розділі подано обґрунтування актуальності вибраної теми, проведено дослідження сучасних близьких до теми наукових робіт, поставлено завдання дослідження. Наступний розділ присвячений розробці методу обробки природної мови. У третьому розділі розробляється система класифікації даних із використанням кластерного аналізу в якості агрегатора рішень моделей, зокрема, ведеться розробка моделі формування ансамблю прийняття рішень та методу виведення

колективногорішення. Четвертий розділ містить дослідження ефективності запропонованих методів. Робота також містить висновки до кожного розділу та загальні висновки, список використаний джерел.

6. Ступінь розкриття теми роботи

Тема наукового дослідження належним чином розкрита в логічній та послідовній структурі представлення. Тема в достатній мірі обґрунтована та досліджено сучасний рівень наукових робіт. Поставлені завдання реалізовані та проведено дослідження ефективності запропонованих методів.

7. Якість оформлення кваліфікаційної роботи

Оформлення кваліфікаційної роботи здійснено у відповідності до необхідних нормта правил.

8. Недоліки кваліфікаційної роботи

Доцільно було б розширити область даних для порівняння ефективності. Виявлені недоліки стосуються аспектів оформлення та не впливають на зміст роботи.

9. Загальний висновок (допускається чи не допускається до захисту), якої оцінки заслуговує кваліфікаційна робота.

Враховуючи рівень виконання та забезпечення усіх необхідних вимог робота може бути допущена до захисту. Рекомендована оцінка відмінно.

Опонент  *Зав. каф. ІПЗ, доктор фіз.-мат. наук,*

Бедратюк Л. П.

Додаток Ж – Відгук наукового керівника

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу магістра

гр. КНм-21-1 Стебелецького Мирослава Мироновича за темою: *Метод класифікації текстових даних на основі групування рішень моделей ансамблю*

1. **Актуальність теми**

В магістерській роботі розроблено розробка методу класифікації текстових даних на основі групування рішень моделей ансамблю. Використано агрегаційних механізм на основі неієрархічного методу кластеризації у завданнях бінарної класифікації текстового контенту. На основі чого визначено напрямок дослідження та поставлено задачі.

2. **Відповідність роботи предметній області спеціальності 122 Комп'ютерні науки та загальним вимогам до наукових робіт**

За ступенем розкриття мети дослідження, рівнем запропонованих методів, логічною послідовністю, проведеними експериментальними дослідженнями робота відповідає вимога до наукових робіт освітньо-кваліфікаційного рівня магістр. Мета, завдання, об'єкт та предмет дослідження відповідають предметній області спеціальності 122 Комп'ютерні науки та вимогам до кваліфікаційної роботи магістра.

3. **Професійні та особистісні якості магістранта**

Рівень набутих компетенцій зі спеціальності комп'ютерні науки, який продемонстрований при роботі над кваліфікаційною роботою при вирішенні наукових задач за сукупністю ознак відповідає вимогам що ставляться до професійних якостей магістрантів.

4. **Ступінь самостійності під час виконання кваліфікаційної роботи**

Кваліфікаційна робота виконана студентом самостійно, сформульовані завдання роботи, розроблено методи та проведені експериментальні дослідження для встановленняефективності запропонованих методів.

5. **Наукова новизна та оригінальність запропонованих підходів**

В роботі наявна наукова новизна, яка відповідає рівню магістра. Отримані

наступні результати:

– запропоновано інноваційний метод групування моделей в унікальні набори та подальше «відсіювання» результатів моделей задля створення ансамблів та агрегування результату в середині кожного ансамблю;

– розроблено метод агрегації моделей у задачах бінарної класифікації на основі кластерного аналізу;

– розроблено метод оцінювання ансамблів з використанням кореляційних показників щодо передбачень базових класифікаторів та оригінальних значень;

– реалізовано систему бінарної класифікації, яка містить модуль візуалізації основних метрик у дво та тривимірному просторах для аналізу отриманих результатів та подальшого дослідження.

6. Ступінь оволодіння методами дослідження

Під час виконання наукових досліджень автором продемонстровано належний рівень володіння методами наукових досліджень та належний рівень застосування набутих компетенції.

7. Повнота та якість розкриття теми роботи

Тема робота в повній мірі розкрита проведеними дослідженнями на належному науковому рівні, який відповідає освітньо-кваліфікаційному рівню магістра.

8. Логічність, послідовність, аргументованість, літературна грамотність викладу матеріалу

Кваліфікаційна робота магістра проведенна з дотриманням та у відповідності до вимог щодо наукової складової, послідовності викладення матеріалу, рівню аргументованості, наукового обґрунтування, доведення та перевірки пропонованих методів. За стилістичним викладенням робота відповідає науковому рівню.

9. Можливість практичного застосування кваліфікаційної роботи, окремих її частин

Практичне значення роботи полягає в розроблені прикладних методів класифікації текстових даних на основі групування рішень моделей ансамблю.

10. Висновок про можливість допуску кваліфікаційної роботи до захисту,на яку оцінку заслуговує робота

Кваліфікаційна робота виконана на належному рівні і може бути допущена до захисту та заслуговує на оцінку *відмінно*.

Науковий керівник _____ к.т.н. Манзюк Е.А.