МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н. Каразіна

Факультет: **ННІ Каразінський банківський інститут**
Кафедра: **Інформаційних технологій та математичного моделювання**
Спеціальність: **122 Комп'ютерні науки**
Освітня програма: **Комп'ютерні науки та інформаційні технології в бізнесі**

Група: АК–41б **денна форма навчання**

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему:

«ДОСЛІДЖЕННЯ ТА АНАЛІЗ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ»

ЗА НАКАЗОМ № 4601–5/335 ВІД 07 ЛЮТОГО 2025 РОКУ

здобувача вищої освіти **Броннікова Кирила Ігоровича**

Робота допущена до захисту в ЕК
протокол кафедри ІТММ №13 від 31.05.2025р.

Завідувач кафедри ІТММ
к.п.н., доцент
_____ Н.І. Стяглик

Науковий керівник
к.ф.-м.н., доцент
_____ Н.М. Чеканова

м. Харків 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В. Н. Каразіна

Факультет навчально–науковий інститут "Каразінський банківський інститут"
Кафедра інформаційних технологій та математичного моделювання
Рівень вищої освіти перший (бакалаврський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки та інформаційні технології в бізнесі

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ **Н. І. Стяглик**
Підпис ініціали, прізвище

"08" лютого 2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЄКТ)

Броннікова Кирила Ігоровича
(прізвище, ім'я, по батькові студента)

1. Тема роботи Дослідження та аналіз алгоритмів машинного навчання

керівник роботи к.ф.-м.н., доцент Чеканова Н.М.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "08" лютого 2025 року № 4601–5/335

2. Строк подання студентом роботи 15 травня 2025 року

3. Перелік питань, які потрібно розробити:
У розділі 1: розглянути теоретичні основи машинного навчання. Провести класифікацію типів задач машинного навчання. Сформулювати постановку задачі машиного навчання.

У розділі 2: дослідити основні алгоритми машинного навчання

У розділі 3: реалізувати комп'ютерну модель класифікації листів, здатну відрізняти спам-повідомлення від звичайних на основі їх змісту. Порівняти ефективність застосування різних алгоритмів машиного навчання для задачі класифікації.

4. План роботи

№ з/п	Назви етапів роботи
1	Вибір здобувачем теми кваліфікаційної бакалаврської роботи
2	Затвердження плану і завдання кваліфікаційної бакалаврської роботи
3	Здача кваліфікаційної бакалаврської роботи керівнику
4	Підпис кваліфікаційної бакалаврської роботи керівника
5	Підпис кваліфікаційної бакалаврської роботи у нормоконтролера
6	Допуск завідувачем кафедри до захисту кваліфікаційної бакалаврської роботи
7	Захист кваліфікаційної бакалаврської роботи

5. Дата видачі завдання 08 лютого 2025 року

Студент _____ К.І. Бронніков
підпис ініціали, прізвище

Керівник роботи _____ Н.М. Чеканова
підпис ініціали, прізвище

РЕФЕРАТ
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ
«ДОСЛІДЖЕННЯ ТА АНАЛІЗ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ»
Броннікова Кирила Ігоровича

Кваліфікаційна бакалаврська робота містить 63 сторінки, 1 таблицю, 14 рисунків, список літератури з 18 найменувань.

Об'єктом дослідження є процес застосування машинного навчання до аналізу великих масивів даних.

Предметом дослідження є алгоритми та методи машинного навчання, що використовуються для класифікації й обробки даних.

Мета кваліфікаційної бакалаврської роботи полягає у дослідженні, порівняльному аналізі та практичному застосуванні алгоритмів машинного навчання для аналізу великих масивів даних.

Завданнями кваліфікаційної бакалаврської роботи є:

- дослідити базові поняття та типи задач машинного навчання;
- сформулювати постановку задачі машинного навчання;
- проаналізувати основні алгоритми класифікації, зокрема: простий байєсівський класифікатор, метод опорних векторів, метод k-ближчих сусідів, дерево рішень та метод випадкового лісу;
- реалізувати комп'ютерну модель класифікації SMS-повідомлень на основі алгоритмів ML та провести експериментальне порівняння їх результатів.

Актуальність дослідження зумовлена необхідністю глибокого аналізу сучасних алгоритмів машинного навчання, порівняння їх характеристик, виявлення сильних та слабких сторін з метою вибору найбільш ефективних підходів для вирішення задач класифікації. Особливої актуальності набуває завдання автоматичного виявлення спаму у текстових повідомленнях, що є критичним для підвищення безпеки комунікацій у цифровому середовищі.

За результатами дослідження порівняно ефективність обраних алгоритмів класифікації, визначено найдоцільніші моделі для задачі виявлення спаму, створено та апробовано програмну реалізацію моделі.

Практична новизна роботи полягає у створенні комп'ютерної моделі класифікації SMS-повідомлень з використанням алгоритмів машинного навчання та проведенні їх експериментального аналізу, що дозволяє визначити оптимальний підхід для задачі виявлення спаму.

Одержані результати можуть бути використані у системах автоматичної обробки повідомлень, інформаційної безпеки, а також як основа для подальших наукових досліджень у сфері текстової аналітики та застосування глибокого навчання в задачах класифікації.

КЛЮЧОВІ СЛОВА: МАШИННЕ НАВЧАННЯ, ЗАДАЧА КЛАСИФІКАЦІЇ, БАЙЄСІВСЬКИЙ КЛАСИФІКАТОР, МЕТОД ОПОРНИХ ВЕКТОРІВ, ДЕРЕВО РІШЕНЬ, ВИПАДКОВИЙ ЛІС, K-БЛИЖЧИХ СУСІДІВ, АНАЛІЗ ТЕКСТУ, ВИЯВЛЕННЯ СПАМУ.

ABSTRACT
AT QUALIFICATION BACHELOR WORK
«RESEARCH AND ANALYSIS OF MACHINE LEARNING ALGORITHMS»
Kyrylo Bronnikov

The bachelor's thesis contains 63 pages, 1 table, 18 figures, and a list of references with 17 titles.

The object of the research is the process of applying machine learning to the analysis of large datasets.

The subject of the research is the algorithms and methods of machine learning used for data classification and processing.

The purpose of the bachelor's thesis is to study, compare, and practically apply machine learning algorithms for the analysis of large volumes of data.

The tasks of a bachelor's degree are:

- to study the basic concepts and types of machine learning tasks;
- to formulate the problem statement of machine learning;
- to analyze key classification algorithms, namely: Naive Bayes classifier, support vector machine, k-nearest neighbors, decision tree, and random forest;
- to implement a computer model for SMS message classification based on ML algorithms and conduct experimental comparison of their performance.

The relevance of the research is driven by the need for in-depth analysis of modern machine learning algorithms, comparison of their characteristics, identification of strengths and weaknesses to select the most effective approaches for solving classification problems. Of particular importance is the task of automatic spam detection in text messages, which is critical for improving communication security in the digital environment.

According to the results of the research: the effectiveness of the selected classification algorithms was compared, the most suitable models for spam detection tasks were identified, and a software implementation of the model was developed and tested.

Main theoretical provisions on the topic of the practical relevance of the work lie in the development and testing of a computer model for SMS message classification using machine learning algorithms and conducting experimental analysis to determine the optimal approach to spam detection.

The results obtained can be used in automatic message processing systems, information security, and also as a basis for further research in the field of text analytics and the application of deep learning to classification tasks.

KEYWORDS: MACHINE LEARNING, CLASSIFICATION PROBLEM, NAIVE BAYES CLASSIFIER, SUPPORT VECTOR MACHINE, DECISION TREE, RANDOM FOREST, K-NEAREST NEIGHBORS, TEXT ANALYSIS, SPAM DETECTION.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧОК, СИМВОЛІВ І ТЕРМІНІВ

ML – Machine Learning (машинне навчання)

AI – Artificial Intelligence (штучний інтелект)

SMS – Short Message Service (служба коротких повідомлень)

SVM – Support Vector Machine (метод опорних векторів)

KNN – K-Nearest Neighbors (метод k-ближчих сусідів)

NB – Naive Bayes (простий байєсівський класифікатор)

DT – Decision Tree (дерево рішень)

RF – Random Forest (випадковий ліс)

NLP – Natural Language Processing (обробка природної мови)

TF-IDF – Term Frequency – Inverse Document Frequency (частота терміну – обернена частота документів)

ROC – Receiver Operating Characteristic (крива робочих характеристик приймача)

AUC – Area Under the Curve (площа під кривою)

TP – True Positive (істинно позитивне спрацювання)

TN – True Negative (істинно негативне спрацювання)

FP – False Positive (хибнопозитивне спрацювання)

FN – False Negative (хибнонегативне спрацювання)

ВСТУП

У сучасних інформаційних системах зростає потреба в ефективному аналізі великих обсягів даних та прийнятті рішень на їх основі. Одним із найбільш перспективних напрямів вирішення цих задач є машинне навчання – підгалузь штучного інтелекту, що дозволяє системам самостійно навчатися з даних та прогнозувати майбутні події без явного програмування. Розвиток методів машинного навчання відкриває широкі можливості для їх практичного застосування, зокрема у задачах класифікації текстових повідомлень, виявлення спаму, аналізу поведінки користувачів тощо.

Актуальність дослідження зумовлена необхідністю глибокого аналізу сучасних алгоритмів машинного навчання, порівняння їх характеристик, виявлення сильних та слабких сторін з метою вибору найбільш ефективних підходів для вирішення задач класифікації. Особливої актуальності набуває завдання автоматичного виявлення спаму у текстових повідомленнях, що є критичним для підвищення безпеки комунікацій у цифровому середовищі.

Об’єктом дослідження є процес застосування машинного навчання до аналізу великих масивів даних.

Предметом дослідження є алгоритми та методи машинного навчання, що використовуються для класифікації й обробки даних.

Мета кваліфікаційної бакалаврської роботи полягає у дослідженні, порівняльному аналізі та практичному застосуванні алгоритмів машинного навчання для аналізу великих масивів даних.

Завданнями кваліфікаційної бакалаврської роботи є:

- дослідити базові поняття та типи задач машинного навчання;

- сформулювати постановку задачі машинного навчання;

- проаналізувати основні алгоритми класифікації, зокрема: простий байєсівський класифікатор, метод опорних векторів, метод k-ближчих сусідів, дерево рішень та метод випадкового лісу;

- реалізувати комп’ютерну модель класифікації SMS-повідомлень на

основі алгоритмів ML та провести експериментальне порівняння їх результатів.

Практична новизна роботи полягає у створенні комп’ютерної моделі класифікації SMS-повідомлень з використанням алгоритмів машинного навчання та проведенні їх експериментального аналізу, що дозволяє визначити оптимальний підхід для задачі виявлення спаму.

Одержані результати можуть бути використані у системах автоматичної обробки повідомлень, інформаційної безпеки, а також як основа для подальших наукових досліджень у сфері текстової аналітики та застосування глибокого навчання в задачах класифікації.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНОГО СТАНУ РОЗВИТКУ МАШИННОГО НАВЧАННЯ

1.1. Основні поняття машинного навчання

Сучасний світ неможливо уявити без інтенсивного використання комп’ютерних технологій, зокрема таких, які здатні не просто виконувати запрограмовані команди, а й приймати рішення на основі аналізу великих обсягів даних. Одним із напрямків, що активно розвивається у цій сфері, є машинне навчання – підгалузь штучного інтелекту. Для кращого розуміння машинного навчання важливо спочатку розглянемо поняття штучного інтелекту. Штучний інтелект – це область інформатики, що досліджує створення комп’ютерних систем, здатних виконувати завдання, які зазвичай вимагають людського інтелекту. Це включає розпізнавання мови, розуміння природної мови, здатність до навчання, планування, прийняття рішень, візуальне сприйняття та інші складні когнітивні процеси. Ідея створення штучного інтелекту базується на припущенні, що людський інтелект можна формалізувати у вигляді алгоритмів і моделей, які комп’ютер може реалізувати [1-4, 7, 9].

Історично зародження штучного інтелекту датується серединою XX століття, коли 1956 року на конференції в Дартмутському коледжі вперше був офіційно використаний термін «штучний інтелект». Ця конференція вважається початком формування напрямку як самостійної наукової дисципліни. На початковому етапі дослідники намагалися створити системи, які могли б імітувати логічне мислення та розв’язувати задачі за допомогою правил і алгоритмів. Однак швидко виявилось, що жорстко запрограмовані правила не здатні охопити складність реального світу. Перші успіхи пов’язані з розробкою експертних систем – програм, які відтворювали знання фахівців у вузьких сферах. Проте такі системи були обмежені за своєю гнучкістю та не вміли навчатися на нових даних. Важливим етапом розвитку

стало формування ідей машинного навчання – підходу, що передбачає автоматичне покращення роботи системи через аналіз досвіду. Відповідно, машинне навчання розглядається як одна із ключових технологій реалізації штучного інтелекту [2-4, 6].

Історія машинного навчання тісно пов’язана з розвитком теоретичної статистики, теорії інформації, обчислювальної математики та нейробіології. У 1957 році Френк Розенблатт запропонував модель персептрона – простий штучний нейрон, що міг класифікувати двовимірні дані. Ця модель стала фундаментом для подальших розробок штучних нейронних мереж. Проте у 1969 році було опубліковано працю Марвіна Мінського і Сеймура Паперта «Персептрон», де було доведено обмеженість персептронів у розв’язанні задач, що не є лінійно роздільними, що на кілька років уповільнило розвиток напрямку. Відродження стало можливим у 1980-х роках із впровадженням алгоритму зворотного розповсюдження помилки, який дозволив ефективно навчати багатошарові нейронні мережі. Цей прорив дав змогу створювати більш складні й потужні моделі. У 1990-х роках машинне навчання розширилося за рахунок методів статистичного навчання: методів опорних векторів, дерев рішень, байєсівських мереж. Паралельно розвивалися алгоритми кластеризації та зниження розмірності даних. З початком XXI століття збільшення обсягів даних (big data) та розвиток апаратного забезпечення, зокрема графічних процесорів (GPU), дали можливість глибинному навчанню (deep learning) стати основним трендом. Складні багатошарові нейронні мережі демонструють виняткову ефективність у розпізнаванні зображень, обробці мови, грі в складні ігри (наприклад, Go, шахи) та автономному керуванні [2-4, 8].

Штучний інтелект і машинне навчання знайшли застосування в численних сферах: у медицині для автоматичного діагностування, аналізу медичних зображень, прогнозування розвитку хвороб та персоналізації лікування; у фінансах – алгоритмічна торгівля, виявлення шахрайства, кредитний скоринг; у промисловості – передбачення відмов обладнання,

оптимізація виробничих процесів, управління якістю; у транспорті – автономні транспортні засоби, оптимізація маршрутів; у розвагах – рекомендаційні системи у потокових сервісах, ігри; у безпеці – розпізнавання облич, аналіз відеопотоків, кібербезпека; в обробці природної мови – голосові асистенти, автоматичний переклад, чат-боти. Машинне навчання є не просто окремою технологією, а невід'ємною складовою сучасних інформаційних систем і інноваційних технологій, що трансформують різні галузі та створюють нові можливості для розвитку суспільства [5, 7, 8].

Машинне навчання – це напрямок штучного інтелекту, який зосереджується на розробці алгоритмів і статистичних моделей, що дозволяють комп'ютерним системам автоматично покращувати свою продуктивність у виконанні певних завдань на основі досвіду, без явного програмування цих правил. Іншими словами, машинне навчання – це метод створення програмного забезпечення, яке вчиться із прикладів і вміє робити прогнози або приймати рішення, аналізуючи вхідні дані. Основні терміни машинного навчання включають дані – інформацію у вигляді чисел, текстів, зображень або інших форматів, які використовуються для навчання моделей; об'єкт – окремий приклад із набору даних, що описується набором ознак; ознаки – характеристики або властивості об'єкта, які описують його атрибути; модель – формалізований опис закономірностей у даних, який дає змогу робити прогнози або класифікації; навчання – процес побудови моделі на основі набору даних; тестування – оцінка якості моделі на нових даних, які не використовувалися під час навчання; цільова змінна – значення, яке модель має передбачити у задачах з учителем; задача з учителем – навчання на основі прикладів, у яких є правильні відповіді; задача без учителя – пошук структури або закономірностей у даних без відомих правильних відповідей; підкріплене навчання – навчання через взаємодію агента з середовищем, де агент отримує винагороди за правильні дії [4-8].

Основні види машинного навчання включають навчання з учителем, де модель навчається на розмічених даних, тобто кожен приклад містить вхідні

ознаки та відповідну правильну відповідь. Завдання моделі – навчитися прогнозувати правильні відповіді для нових, невідомих даних. Прикладами таких завдань є класифікація (розпізнавання об’єктів за класами, наприклад, спам чи не спам) та регресія (прогнозування числових значень, наприклад ціна будинку або температура). Навчання без учителя передбачає роботу з даними без розмітки, і модель повинна знайти приховані закономірності, групи або структури у даних. Прикладами таких завдань є кластеризація (групування клієнтів за схожістю поведінки) та зниження розмірності (побудова компактного представлення даних). Підкріплене навчання – це метод, у якому агент навчається через взаємодію з оточенням, отримуючи винагороду або штраф за дії, з метою максимізації сумарної винагороди. Цей метод широко застосовується в робототехніці, іграх та автономних системах.

Різноманіття алгоритмів машинного навчання охоплює прості лінійні моделі, такі як лінійна регресія та логістична регресія, які добре інтерпретуються та застосовуються для базових задач. Рішення дерев і ансамблеві методи (наприклад, випадкові ліси, градієнтний бустинг) забезпечують високу точність і здатність працювати з різними типами даних. Метод опорних векторів є ефективним у задачах класифікації з високою розмірністю. Нейронні мережі, зокрема глибокі, здатні моделювати дуже складні залежності та структуровані дані, такі як зображення, мова та текст. Для кожного класу задач існують спеціальні підходи та методи оцінки якості моделей, що допомагають вибрати оптимальну модель для конкретної задачі.

Одним із ключових аспектів машинного навчання є підготовка даних, яка включає очищення, нормалізацію, відбір ознак і трансформації. Якість та кількість даних суттєво впливають на результати навчання моделей. Для оцінки моделей використовуються різні метрики, залежно від типу задачі: точність, повнота, F1-мера для класифікації, середньоквадратична помилка для регресії, коефіцієнт силового показника та інші. Вибір правильної метрики та правильна оцінка моделей дозволяють забезпечити високу якість і надійність прогнозів [4-9, 12, 15].

Інструменти та середовища для реалізації машинного навчання постійно розвиваються. Серед найпопулярніших бібліотек та фреймворків варто відзначити Python-бібліотеки: Scikit-learn, TensorFlow, PyTorch, Keras, які надають широкі можливості для розробки, навчання та впровадження моделей машинного навчання. Окрім того, існують платформи, що дозволяють працювати з машинним навчанням у хмарі, інтегруючи обчислювальні ресурси і масштабування. Ефективне застосування машинного навчання вимагає розуміння як теоретичних основ, так і практичних аспектів – вибору алгоритмів, налаштування параметрів, роботи з даними, а також оцінки результатів [5, 8].

Отже, машинне навчання є потужним інструментом сучасної науки і технологій, що дозволяє автоматизувати аналіз і прийняття рішень на основі даних. Його розвиток тісно пов’язаний із розвитком комп’ютерних технологій, статистики та штучного інтелекту. Впровадження машинного навчання має великий вплив на різні галузі економіки, науки, медицини, техніки, що робить його однією з найбільш перспективних і динамічних сфер досліджень і практичного застосування.

1.2. Типи задач машинного навчання

Типи задач машинного навчання охоплюють широкий спектр підходів і методів, які використовуються для розв’язання різноманітних проблем у галузі аналізу даних та штучного інтелекту. Основним критерієм класифікації задач машинного навчання є наявність або відсутність розмічених даних, а також спосіб взаємодії алгоритму з навчальним середовищем. Розглянемо три основні типи задач: навчання з учителем, навчання без учителя та підкріплене навчання. Кожен із цих типів відрізняється підходами до побудови моделей, способом навчання та застосуванням [1-4, 7, 10].

Навчання з учителем є найбільш поширеним і добре дослідженим напрямом машинного навчання. У цій парадигмі модель навчається на основі навчального набору даних $\{(x_i, y_i)\}_{i=1}^{N}$, де $x_i \in \mathbb{R}^d$ – вектор вхідних ознак, а y_i – цільова змінна, що може бути числовою (у задачах регресії) або дискретною (у задачах класифікації).

Мета навчання полягає у побудові функції $f : \mathbb{R}^d \to Y$, яка мінімізує деякий функціонал втрат $L(y, f(x))$ на нових, раніше невідомих даних, де Y – множина можливих виходів (наприклад, множина класів або множина дійсних чисел) [7, 8].

У задачах класифікації цільова змінна є категоріальною: $y_i \in \{1, 2, ..., K\}$, де K – кількість класів. Класична формалізація класифікації полягає у знаходженні розбиття простору ознак на області, відповідні кожному класу, за допомогою функції f, що призначає кожному вектору x один із класів. Приклади застосувань класифікації включають розпізнавання образів (наприклад, класифікація зображень на об'єкти: людина, автомобіль, тварина), медичну діагностику (визначення хвороби за набором симптомів), розпізнавання мови, фільтрацію спаму у електронній пошті тощо (рис. 1.1).

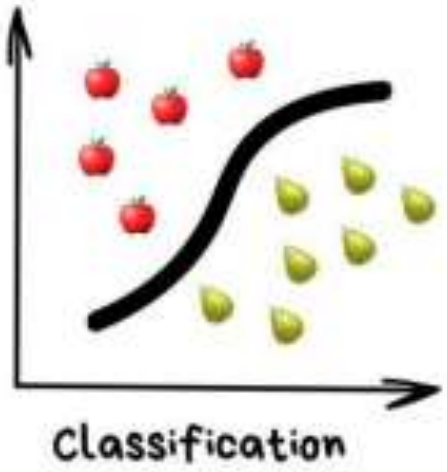

Рис. 1.1. Задачі класифікації

Одним із базових алгоритмів класифікації є логістична регресія, яка моделює ймовірність належності до класу за допомогою сигмоїдальної функції. Формально для двокласової задачі ймовірність належності обчислюється як $P(y=1 \mid x)=\frac{1}{1+e^{-(w^{\top}x+b)}}$, де w – вектор ваг, b – зміщення. Модель навчається шляхом максимізації логарифмічної ймовірності (логарифмічної функції правдоподібності) [7, 9].

Задачі регресії, навпаки, передбачають прогнозування неперервної цільової змінної $y \in \mathbb{R}$ (рис. 1.2). Прикладами є прогнозування вартості нерухомості за характеристиками будинку, оцінка попиту на товари, прогнозування фінансових показників або температурних значень [8].

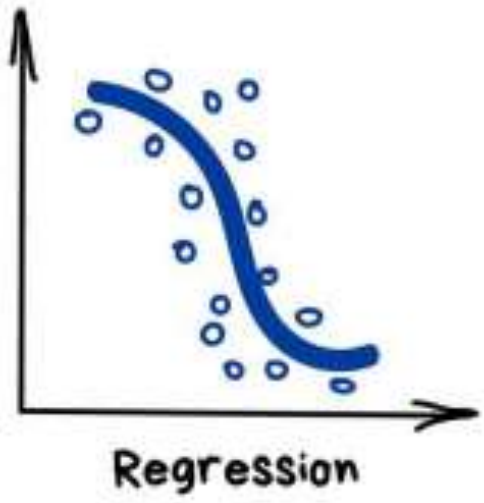

Рис. 1.2. Задачі регресії

Одним із найбільш інтуїтивних алгоритмів регресії є лінійна регресія (рис. 1.3), що передбачає наближення цільової змінної лінійною функцією ознак: $y \approx f(x)=-w^{\top}x+b,$ де параметри $w,\ b$ знаходять мінімізацією середньоквадратичної помилки [3-5]:

$$MSE=\frac{1}{N}\sum_{i=1}^{N}(y_i-w^{\top}x_i-b)^2.$$

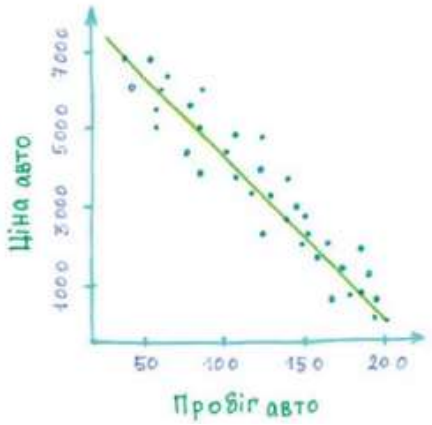

Рис. 1.3. Лінійна регресія

Окремим випадком регресії є прогнозування, що акцентує увагу на часових залежностях у даних. У задачах прогнозування важливим є використання історичних даних для прогнозу майбутніх значень (рис. 1.4). Наприклад, у прогнозуванні попиту на товари використовується модель, яка враховує сезонність, тренди і випадкові коливання [5- 9]. Математично задачі прогнозування можна формалізувати як пошук функції f, що для даної історії спостережень $\{x_{t-n}, \ldots, x_t\}$ передбачає майбутнє значення x_{t+h}:

$$x_{t+h} = f(x_{t-n}, \ldots, x_t) + \varepsilon,$$

де ε – випадкова похибка.

Популярні методи прогнозування включають авторегресивні моделі (AR), модель ковзного середнього (MA), а також більш складні нейронні мережі з рекурентною архітектурою (RNN, LSTM).

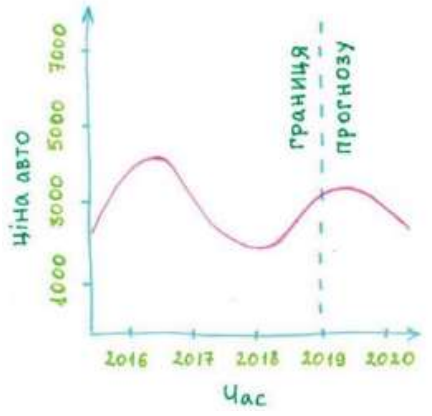

Рис. 1.4. Задача прогнозування

Навчання без учителя розглядається як задачі, у яких відсутня цільова змінна або мітки. Вхідні дані представлені у вигляді набору векторів ознак $\{x_i\}_{i=1}^{N}$, без відомої відповіді. Головна мета навчання без учителя полягає у виявленні структури або закономірностей у даних, що можуть бути корисними для подальшого аналізу або підготовки до навчання з учителем. Найбільш поширеними задачами навчання без учителя є кластеризація, зниження розмірності, пошук асоціативних правил і виявлення аномалій.

Кластеризація полягає у розбитті набору даних на K підгруп (кластерів), в яких об'єкти є більш схожими між собою, ніж із об'єктами з інших кластерів (рис. 1.5). Формально це можна представити як пошук розбиття $S = \{S_1, S_2, ..., S_K\}$, де $S_j \subseteq \{1, ..., N\}$, яке мінімізує внутрішньокластерну дисперсію [10-12]. Наприклад, у методі - середніх (k-means) мінімізується функціонал:

$$J = \sum_{j=1}^{K} \sum_{i \in S_j} || x_i - \mu_j ||^2,$$

де μ_j – центр кластера S_j, зазвичай середнє арифметичне векторів у кластері.

Кластеризація знаходить застосування у сегментації клієнтів у маркетингу, виявленні аномалій у мережевій безпеці, аналізі генетичних даних тощо.

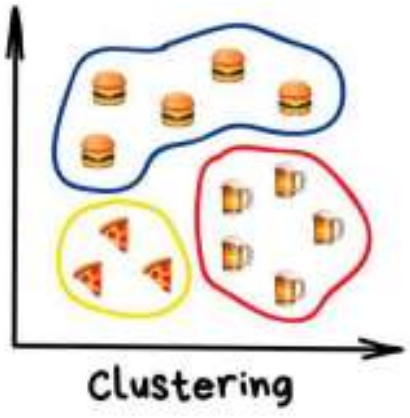

Рис. 1.5. Задача кластеризації

Зниження розмірності є важливою задачею для візуалізації, прискорення обчислень і поліпшення якості моделей. Зниження розмірності полягає у відборі або трансформації початкових ознак у компактніший набір, що зберігає основну інформацію, що покращує візуалізацію і зменшує складність моделей (рис. 1.6). Одним із класичних методів є метод головних компонент (PCA), який знаходить ортогональну проекцію даних у простір меншої розмірності, що максимізує дисперсію проекції. Формально PCA вирішує задачу пошуку матриці проекції $W \in \mathbb{R}^{d \times m}$, (де $m<d$) для максимізації дисперсії [10-12]:

$$\max_{W} Var(W^{\top} X),$$

де X – матриця вхідних даних.

Зниження розмірності допомагає виявляти основні тренди і структуру даних, спрощує інтерпретацію та подальший аналіз.

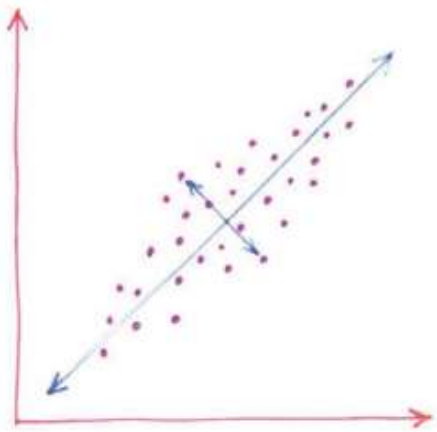

Рис. 1.6. Задача зменшення розмірності

Пошук асоціативних правил – це задача виявлення закономірностей у великих наборах транзакцій, що описуються у вигляді множин предметів. Асоціативне правило має вигляд $A \Rightarrow B$, де A та B – множини предметів. Основні характеристики правила – підтримка (support) і довіра (confidence):

$$support(A \Rightarrow B) = \frac{|\{T_i : A \cup B \subseteq T_i\}|}{N},$$

$$confidence(A \Rightarrow B) = \frac{|\{T_i: A \cup B \subseteq T_i\}|}{|\{T_i: A \subseteq T_i\}|},$$

де T_i – транзакції.

Пошук асоціативних правил широко використовується для виявлення залежностей між подіями, наприклад, у маркетингових дослідженнях для аналізу поведінки споживачів [11-14].

Виявлення аномалій (аналітика виявлення викидів) – це задача виявлення вхідних даних, які суттєво відрізняються від більшості (рис. 1.7). Аномалії можуть свідчити про помилки, шахрайство, технічні несправності. Формально аномалією вважається об’єкт x, для якого міра аномальності $a(x)$ перевищує деякий поріг:

$$a(x) = \frac{1}{N}\sum_{i=1}^{N} d(x, x_i),$$

де $d(x, x_i)$ – метрика схожості (наприклад, евклідова відстань), і об’єкти, що значно віддалені від більшості (вважаються аномальними).

Методи включають локальний вихідний фактор (LOF), метод опорних векторів для аномалій (One-Class SVM), ізоляційний ліс (Isolation Forest). Наприклад, у банківській сфері виявлення аномалій допомагає ідентифікувати шахрайські транзакції.

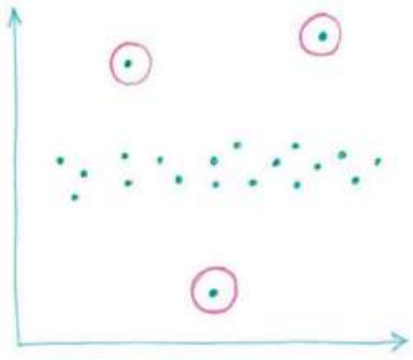

Рис. 1.7 Задача виявлення аномалій

Навчання без учителя дозволяє досліджувати великі масиви даних і виявляти приховані зв’язки без попередніх знань.

Підкріплене навчання є відносно новим і специфічним типом машинного навчання, в якому агент взаємодіє з середовищем, роблячи дії і отримуючи за це винагороди або покарання. Мета агента – навчитися вибирати послідовність дій, яка максимізує сумарну винагороду у довгостроковій перспективі. На відміну від навчання з учителем, де є безпосередня вказівка правильних відповідей, у підкріпленому навчанні агент має експериментувати і вчитися на основі досвіду, що може бути частково або повністю невизначеним. Формально таке навчання моделюється у вигляді задачі прийняття рішень у марковських процесах прийняття рішень (MDP). MDP описується п'ятіркою (S, A, P, R, γ) де S – простір станів, A – множина дій, $P : S \times A \times S \rightarrow [0, 1]$ – ймовірність переходу до нового стану, $R : S \times A \rightarrow R$ – функція винагороди, $\gamma \in [0, 1]$ – коефіцієнт дисконтованої суми винагород [8-10]. Мета агента – знайти політику $\pi : S \rightarrow A$, яка максимізує очікувану дисконтовану суму винагород:

$$G_t = \mathrm{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right].$$

Задачі підкріпленого навчання широко застосовується у робототехніці, автономних транспортних засобах, іграх, де алгоритми вчяться приймати рішення в складних і змінних умовах. Підкріплене навчання часто комбінується з глибинними нейронними мережами, що дозволяє ефективно опрацьовувати складні простори станів і дій [9-10].

Окрім основних типів, існують також гібридні підходи, які поєднують елементи різних типів навчання для покращення результатів. Наприклад, напівкероване навчання використовує як розмічені, так і нерозмічені дані, що особливо корисно у випадках, коли отримання міток є дорогим або трудомістким. Трансферне навчання (transfer learning) дозволяє переносити знання, отримані при розв'язанні однієї задачі, на іншу, часто споріднену, завдяки чому значно скорочується час і ресурси на навчання нової моделі. Самонавчання (self-supervised learning) передбачає створення штучних міток із неструктурованих даних і навчання моделі на таких завданнях, що

останнім часом стало особливо актуальним у обробці природної мови та комп'ютерному зорі [12-13].

1.3. Постановка задачі машинного навчання

Загалом, формальна постановка задачі машинного навчання передбачає наявність простору вхідних даних $X \subseteq \mathbb{R}^d$ та простору вихідних або цільових значень Y, які модель повинна передбачити. У задачах з учителем передбачається наявність пари (x, y), де $x \in X$, $y \in Y$, і мета навчання полягає в побудові функції $f : X \to Y$, яка узагальнює залежність між вхідними і вихідними даними. За умови, що пари (x_i, y_i) утворюють навчальну вибірку $D_{train} = \{(x_i, y_i)\}_{i=1}^{N}$, задача формулюється як мінімізація функції втрат (або ризику) [11-13]:

$$R(f) = E_{(x,y)\sim P(x,y)}[L(f(x), y)],$$

де $L(f(x), y)$ – функція втрат, що визначає помилку між передбаченим і реальним значенням;

$P(x, y)$ – невідомий розподіл, з якого беруться дані.

Оскільки P є невідомим, на практиці мінімізується так званий емпіричний ризик:

$$R_{emp}(f) = \frac{1}{N}\sum_{i=1}^{N} L(f(x_i), y_i).$$

Це є основою для більшості алгоритмів машинного навчання [11-13].

Процес постановки задачі машинного навчання зазвичай починається з попереднього аналізу даних. На цьому етапі проводиться вивчення структури вхідної інформації: перевіряється наявність пропущених значень, вимірюється статистика розподілів ознак, виявляються можливі аномалії або шум, а також проводиться перетворення ознак до форматів, які можуть бути оброблені машинними алгоритмами. Можливим є також створення нових ознак або зменшення розмірності простору за допомогою відповідних

методів (наприклад, головних компонент – PCA). У випадку наявності мішаних типів даних (числові, категоріальні, текстові, часові ряди) проводиться відповідна нормалізація, кодування або токенізація [12].

Після попередньої обробки даних необхідно підібрати або реалізувати відповідний метод навчання та аналізу. Це передбачає вибір моделі, яка буде використовуватися, зокрема її класу (лінійна модель, дерево рішень, нейронна мережа тощо) та алгоритму оптимізації параметрів (градієнтний спуск, стохастичний градієнтний спуск, метод Ньютона, еволюційні алгоритми).

Вибір моделі залежить від типу задачі:

- для задач регресії частіше застосовуються лінійні регресії, дерева регресії, методи ансамблю (наприклад, випадковий ліс);
- для задач класифікації – логістична регресія, метод опорних векторів (SVM), нейронні мережі;
- для задач кластеризації – *k*-середніх, ієрархічна кластеризація, DBSCAN тощо [12, 13].

На наступному етапі необхідно поділити доступні дані на дві частини: навчальну вибірку та тестувальну вибірку. Навчальна вибірка (як правило, 70–80% усіх даних) використовується безпосередньо для побудови моделі, тоді як тестувальна вибірка (20–30%) застосовується для незалежного оцінювання якості моделі на нових, ще не бачених прикладах. Такий поділ дозволяє уникнути перенавчання (overfitting), коли модель ідеально відтворює навчальні приклади, але не може узагальнювати нові ситуації.

Наступним ключовим етапом є навчання машини на основі наявної бази даних. Це означає побудову алгоритму прийняття рішень – функції f, яка наближає залежність між x та y на основі емпіричних даних. У більшості випадків навчання зводиться до розв’язання задачі оптимізації [12, 13]:

$$f^* = arg\ \min_{f \in F}\left[\frac{1}{N}\sum_{i=1}^{N} L(f(x_i), y_i) + \lambda\Omega(f)\right],$$

де $\Omega(f)$ – регуляризатор, який контролює складність моделі;

λ – коефіцієнт регуляризації.

Регуляризація зменшує ризик перенавчання, особливо коли кількість ознак перевищує кількість спостережень.

Після навчання моделі необхідно здійснити перевірку побудованого алгоритму на тестувальній вибірці. Цей етап передбачає обчислення метрик якості, які дозволяють оцінити, наскільки добре модель узагальнює дані. Для задач класифікації типовими метриками є точність (accuracy), повнота (recall), точність (precision), F1-мірка, ROC-AUC. Для задач регресії застосовуються середньоквадратична помилка (MSE), середня абсолютна помилка (MAE), коефіцієнт детермінації R^2. У випадку незадовільних результатів модель може бути змінена або доопрацьована шляхом повторного вибору ознак, модифікації структури моделі, корекції гіперпараметрів тощо [11-13].

Останній етап – це перевірка побудованого алгоритму у реальному світі. На цьому етапі модель інтегрується в прикладну систему, де вона повинна опрацьовувати нові дані в режимі реального часу або пакетної обробки (batch mode). Надзвичайно важливо забезпечити можливість постійного моніторингу її продуктивності, автоматичного виявлення змін у розподілах вхідних даних (data drift, concept drift), а також періодичне перенавчання моделі в міру накопичення нових даних. У більш складних сценаріях, наприклад у фінансовому аналізі або медичній діагностиці, перевірка моделі в реальному середовищі проводиться за участю експертів та з дотриманням етичних норм [12, 13].

Таким чином, постановка задачі машинного навчання включає послідовне проходження таких етапів: попередній аналіз даних, вибір і налаштування моделі, поділ вибірки на навчальну та тестувальну, навчання алгоритму на навчальних даних, перевірка якості моделі на тестувальній вибірці, а також остаточна перевірка ефективності алгоритму у реальних умовах. Формальний опис задачі у вигляді функціоналу помилок, простору гіпотез та оптимізаційної задачі дозволяє не лише зрозуміти принцип роботи

моделі, а й забезпечити об'єктивність її оцінювання та можливість узагальнення результатів. Саме тому постановка задачі машинного навчання вважається одним із найважливіших етапів у всьому циклі побудови інтелектуальних систем.

1.4. Висновки до розділу 1

У першому розділі проведено аналіз теоретичних і прикладних основ машинного навчання як складової штучного інтелекту. Розглянуто базові поняття: модель, ознаки, вибірка, функція втрат, навчання з учителем і без учителя. Висвітлено мету побудови моделей – узагальнення закономірностей у даних і автоматизація прийняття рішень. Наведено огляд етапів розвитку галузі – від статистичних методів до глибоких нейронних мереж.

Систематизовано основні типи задач машинного навчання: регресія, класифікація, кластеризація, зменшення розмірності, виявлення аномалій, прогнозування та пошук правил. Для кожного типу задачі наведено математичну постановку, приклади застосування та функції втрат. Наголошено на ролі оптимізації як уніфікованого підходу до побудови моделей.

Описано типову послідовність дій у процесі машинного навчання: аналіз даних, вибір методу, поділ вибірки, навчання, тестування, впровадження. Розглянуто формальне представлення задачі як мінімізації емпіричного ризику з регуляризацією, а також важливість запобігання перенавчанню, вибору метрик оцінки та моніторингу моделі у реальному середовищі.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

У сучасних умовах стрімкого розвитку інформаційних технологій машинне навчання стало одним із ключових напрямів у галузі штучного інтелекту. Його алгоритми дозволяють комп'ютерним системам самостійно навчатися на основі даних, виявляти приховані закономірності та приймати рішення без явного програмування. Такий підхід відкриває широкі можливості для створення інтелектуальних систем у різних сферах – від медицини й фінансів до промисловості, безпеки та комунікацій [11-14].

Дослідження алгоритмів машинного навчання є актуальним завданням, оскільки ефективність побудованих моделей значною мірою залежить від правильного вибору алгоритму, якості вхідних даних, а також особливостей предметної області. Існує велика кількість методів, що реалізують різні підходи до навчання – як з учителем, так і без учителя – серед яких найпоширенішими є наївний байєсівський класифікатор, метод опорних векторів, дерева рішень, логістична регресія, метод k-ближчих сусідів та інші. У цьому розділі розглянемо теоретичні основи найуживаніших алгоритмів машинного навчання, їх характеристики, переваги та недоліки.

2.1. Простий байєсівський класифікатор

Простий байєсівський класифікатор (Naive Bayes) є одним із найбільш поширених алгоритмів машинного навчання, який використовується для задач класифікації. Цей алгоритм базується на застосуванні теореми Байєса у поєднанні з припущенням умовної незалежності ознак, що дозволяє ефективно і швидко обробляти дані, навіть якщо кількість ознак значна. Основна ідея методу полягає в оцінюванні ймовірності належності об'єкта до певного класу на основі значень його ознак [12-15].

Теорема Байєса формулюється таким чином: для події C_k (класу) і спостереження x (вектор ознак) апостеріорна ймовірність належності до класу C_k визначається як:

$$P(C_k | x) = \frac{P(x | C_k) \cdot P(C_k)}{P(x)},$$

де $P(C_k | x)$ – ймовірність того, що об'єкт з ознаками $x = (x_1, x_2, ..., x_n)$ належить до класу C_k;

$P(x | C_k)$ – ймовірність спостереження ознак x за умови, що об'єкт належить до класу C_k;

$P(C_k)$ – апріорна ймовірність класу C_k;

$P(x)$ – загальна ймовірність появи ознак x [11, 13-16].

Для задачі класифікації важливо знайти клас, який максимізує апостеріорну ймовірність, тобто:

$$\hat{C} = arg \max_{C_k} P(C_k | x).$$

Оскільки знаменник $P(x)$ однаковий для всіх класів, при порівнянні класів він не впливає на вибір, і формула спрощується до пошуку максимума добутку [5-8, 11, 13-16]

$$\hat{C} = arg \max_{C_k} P(C_k) \cdot P(x | C_k).$$

Основна складність полягає в обчисленні ймовірності $P(x | C_k)$ – спільної ймовірності спостереження вектору ознак, що зазвичай є складним, особливо при великій кількості ознак і їх взаємозалежності. Для спрощення обчислень вводиться ключове припущення про умовну незалежність ознак за умови класу C_k, тобто вважається, що кожна ознака x_i не залежить від інших ознак, якщо відомо, що об'єкт належить до класу C_k. Це означає, що:

$$P(x | C_k) = \prod_{i=1}^{n} P(x_i | C_k).$$

Завдяки цьому припущенню складна спільна ймовірність розкладається у добуток простих одномірних ймовірностей, що значно спрощує оцінювання моделі. На практиці це припущення часто не виконується повністю, але

навіть приблизна умовна незалежність забезпечує досить добрі результати класифікації [8, 11-14].

Процес навчання простого байєсівського класифікатора зводиться до оцінювання апріорних ймовірностей $P(C_k)$ і умовних ймовірностей $P(x_i| C_k)$ на основі навчальної вибірки. Апріорні ймовірності класів розраховуються як частка об'єктів відповідного класу у навчальних даних, тобто^

$$P(C_k) = \frac{N_k}{N},$$

де N_k – кількість об'єктів класу C_k;

N – загальна кількість об'єктів.

Оцінка умовних ймовірностей залежить від типу ознак: дискретні ознаки можуть оцінюватися за допомогою частот, а неперервні – за допомогою припущення про нормальний розподіл [8, 11-14].

Існують основні варіанти простого байєсівського класифікатора. Перший – це Multinomial Naive Bayes, який часто використовується для задач текстової класифікації, де ознаки відповідають частотам слів у документі. У цьому випадку умова ймовірність ознаки x_i у класі C_k оцінюється за формулою [8, 11-14]:

$$P(x_i| C_k) = \frac{число появ\, x_i \ у\ C_k + \alpha}{\sum_j число появ\, x_i \ у\ C_k + \alpha \cdot d},$$

де α – параметр згладжування Лапласа, що дозволяє уникнути проблеми нульових ймовірностей;

d – кількість різних ознак.

Другий варіант – Bernoulli Naive Bayes, застосовний для бінарних ознак, які позначають присутність або відсутність певної властивості.

Третій – Gaussian Naive Bayes, який припускає, що неперервні ознаки розподілені за нормальним законом у кожному класі. В цьому випадку оцінка ймовірності ознаки задається функцією густини нормального розподілу [12, 14]:

$$P(x_i|\ C_k) = \frac{1}{\sqrt{2\pi\sigma_{k,i}^2}} exp\left(-\frac{(x_i - \mu_{k,i})^2}{2\sigma_{k,i}^2}\right),$$

де $\mu_{k,i}$ – середнє значення та дисперсія ознаки x_i у класі C_k, обчислені на навчальних даних.

Основні переваги простого байєсівського класифікатора полягають у його простоті, швидкості та ефективності. Навчання моделі виконується швидко, оскільки полягає лише у підрахунку частот і параметрів розподілів. Завдяки припущенню незалежності ознак модель добре справляється з задачами високої розмірності, наприклад, з текстовою класифікацією, де кількість ознак (слів) може бути дуже великою. Крім того, Naive Bayes показує високу стійкість навіть при невеликому обсязі навчальних даних, оскільки не вимагає складного оптимізаційного процесу. Важливою перевагою є також можливість швидкого оновлення моделі, коли надходять нові дані, без необхідності повного перенавчання [8, 9, 13].

Однак цей метод має і певні недоліки. Головним є сильне припущення про умовну незалежність ознак, яке в багатьох практичних задачах не виконується, оскільки ознаки часто мають кореляції. Це може призводити до зниження точності класифікації в ситуаціях із суттєвими залежностями між ознаками. Ще одним недоліком є проблема нульових ймовірностей: якщо у навчальних даних для деякого класу не зустрічається певна ознака, оцінка її умовної ймовірності дорівнює нулю, що робить нульовою ймовірність всього класу. Цю проблему вирішують за допомогою згладжування, наприклад, Лапласового, яке додає невелике додатне число до кожного підрахунку. Також Naive Bayes не здатен моделювати складні нелінійні взаємозв’язки між ознаками, що обмежує його застосування в деяких сферах, де важлива більш точна модель поведінки ознак [13].

Простий байєсівський класифікатор знаходить широке застосування у різних сферах. Найпопулярнішою областю є фільтрація електронної пошти для виявлення спаму, де алгоритм ефективно класифікує повідомлення на "спам" та "не спам" за допомогою аналізу частот вживання слів. Крім того,

Naive Bayes використовується в системах рекомендацій, аналізі тональності текстів, класифікації новин, у медичній діагностиці для визначення ризику певних захворювань, а також у багатьох інших прикладних задачах [12-13].

2.2. Метод опорних векторів

Метод опорних векторів (Support Vector Machine, SVM) є одним із найпотужніших і найбільш широко використовуваних алгоритмів машинного навчання, особливо для задач класифікації, регресії та виявлення аномалій. Основна ідея методу полягає в побудові гіперплощини, яка оптимально розділяє точки даних різних класів у багатовимірному просторі ознак. У випадку двокласової класифікації алгоритм прагне знайти таку гіперплощину, яка буде максимально віддаленою від найближчих точок кожного класу, що й отримали назву опорні вектори. Відстань від гіперплощини до найближчих точок визначає зазор (margin), і метою SVM є максимізація цього зазору [8, 11-14].

Формально, нехай задано навчальну вибірку з n об’єктів: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, де $x_i \in \mathbb{R}^d$ – вектор ознак а $y_i \in \{-1, 1\}$ – мітка класу. Метою є побудова гіперплощини, що розділяє класи та задається рівнянням $w^T x + b = 0$, де w – вектор нормалі, а b – зсув. Для того щоб класи були розділені правильно, має виконуватись нерівність $y_i(w^T x_i + b) \geq 1$ для всіх i. Тоді задача оптимізації зводиться до знаходження такого w і b, які мінімізують функціонал $\frac{1}{2}||w||^2$, тобто максимізують зазор між класами за умови правильного класифікування всіх об'єктів [8, 9, 11-14].

Однак у реальних задачах часто дані не є лінійно роздільними. У такому випадку використовується підхід м’якого зазору (soft margin), який дозволяє деяким об’єктам порушувати умову розділення. До моделі додаються змінні $\xi_i \geq 0$, які відповідають за допуск до помилок, і задача

формулюється як: мінімізувати $\frac{1}{2}\| w\|^2 + C\sum_{i=1}^{n}\xi_i$, де $C>0$ – параметр регуляризації, що визначає ступінь штрафу за помилки [8, 9, 11-14].

У випадках, коли дані не можна лінійно розділити у вихідному просторі ознак, застосовується трюк із ядром (kernel trick). Ідея полягає в тому, щоб за допомогою деякого нелінійного відображення $\phi(x)$ перенести дані в простір вищої розмірності, в якому вони вже можуть бути лінійно роздільними. При цьому не потрібно явно обчислювати нові координати – достатньо використовувати ядрову функцію $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$, яка обчислює скалярний добуток у новому просторі. Найбільш поширеними є такі ядра: лінійне $K(x, x') = x^T x'$, поліноміальне $K(x, x') = (x^T x' + c)^d$, радіально-базисна функція (RBF) $K(x, x') = exp(-\gamma \| x - x' \|^2)$ та сигмоїдне $K(x, x') = tanh(\kappa x^T x' + c)$.

Ключовим аспектом методу є те, що побудова гіперплощини залежить лише від тих точок, які розміщені найближче до неї – тобто опорних векторів. Ці точки мають найбільше значення для навчання моделі, тоді як решта даних не впливають на положення гіперплощини. Це забезпечує ефективність та узагальнюючу здатність моделі [8, 9, 11-14].

Метод опорних векторів має низку переваг. По-перше, він ефективний у високорозмірних просторах, що робить його придатним для задач, пов’язаних із текстовими або біоінформатичними даними. По-друге, завдяки максимізації зазору, SVM має високу здатність до узагальнення та зменшену ймовірність перенавчання. По-третє, метод підтримує нелінійне розділення класів через використання ядер, що робить його гнучким у роботі з різноманітними типами даних. Крім того, SVM потребує зберігання лише опорних векторів, що дозволяє зменшити обсяг пам’яті для збереження моделі. Також існує велика кількість бібліотек з ефективними реалізаціями, зокрема в Python (scikit-learn, libsvm) [11-14].

Однак метод має й недоліки. По-перше, він чутливий до вибору параметрів регуляризації C та параметрів ядра, зокрема параметра γ у RBF. Неправильне налаштування може суттєво знизити якість класифікації. По-друге, при великому обсязі даних (сотні тисяч або мільйони об'єктів) SVM може працювати повільно, оскільки обчислення включають розв’язання квадратичної задачі оптимізації. По-третє, модель складно інтерпретувати – на відміну від дерев рішень або логістичної регресії, результати SVM важко пояснити людині. Також метод погано масштабується для задач, де класи значно перетинаються або мають багато шуму [11-14].

Метод опорних векторів широко використовується у практиці. Він показав високу ефективність у задачах розпізнавання образів, зокрема класифікації рукописних цифр у наборі даних MNIST. У сфері обробки природної мови SVM застосовується для класифікації текстів, виявлення спаму, тематичного аналізу. В біоінформатиці метод використовують для класифікації білків, аналізу геномних даних. Також він застосовується у медицині – для діагностики хвороб за результатами аналізів або зображень. У виробничих системах SVM використовується для контролю якості та виявлення дефектів [11-14].

Отже, метод опорних векторів є одним із ключових інструментів машинного навчання. Його здатність ефективно працювати з малими обсягами даних, висока точність класифікації, гнучкість завдяки ядерним функціям та теоретично обґрунтована геометрична основа роблять його незамінним у багатьох прикладних задачах. Незважаючи на деякі обмеження, пов’язані з продуктивністю та складністю інтерпретації, SVM залишається золотим стандартом у задачах двокласової класифікації і продовжує відігравати важливу роль у розвитку сучасних систем інтелектуального аналізу даних.

2.3. Метод k-ближчих сусідів

Метод k-ближчих сусідів (k-Nearest Neighbors, k-NN) є одним із найпростіших і водночас ефективних алгоритмів машинного навчання, який використовується як для задач класифікації, так і для регресії. Його ключова ідея полягає в тому, що передбачення для нового об'єкта здійснюється на основі інформації про k найближчих до нього об’єктів у навчальній вибірці. Метод не має явної фази навчання, оскільки весь процес прийняття рішень відбувається під час класифікації нового прикладу, через що k-NN вважається методом «лінивого навчання» (lazy learning) [12-14].

Під час роботи алгоритму, для кожного нового вхідного прикладу обчислюється відстань до всіх наявних об'єктів навчальної множини. Найчастіше використовується евклідова відстань, яка визначається як [12]:

$$d(x, x') = \sqrt{\sum_{i=1}^{n} (x_i - x'_i)^2},$$

де x та x' – вектори ознак двох об’єктів.

Альтернативні метрики відстані включають мангеттенську, косинусну, або відстань Мінковського. Після обчислення відстаней вибираються k об'єктів, які є найближчими до нового прикладу. Клас нового об’єкта визначається голосуванням – найчастіше обирається клас, що найчастіше зустрічається серед k сусідів. У випадку рівної кількості голосів може застосовуватись додаткове правило, наприклад, зменшення k або випадковий вибір. Формально, клас нового об'єкта $\hat{y}$ визначається як:

$$\hat{y} = arg \max_{y \in Y} \sum_{i \in N_k(x)} \delta(y_i, y),$$

де $N_k(x)$ – множина індексів k найближчих сусідів;

y_i – відома мітка класу для кожного сусіда;

$\delta(y_i, y)$ – функція Кронекера, яка дорівнює 1, якщо $y_i = y$, і 0 – інакше.

Існує також модифікація методу – зважений k-NN, де внесок кожного сусіда у голосування залежить від відстані: ближчі сусіди мають більшу вагу.

Наприклад, вага може визначатися як обернена величина відстані: $w_i = 1/(d(x, x_i) + \varepsilon)$, де ε – мале число для уникнення ділення на нуль [8, 11-14].

Ключовим параметром методу є значення *k*. Занадто мале значення (наприклад, $k = 1$) робить модель надмірно чутливою до шуму, що призводить до перенавчання (overfitting). Навпаки, надто велике значення може спричинити зниження точності через включення в голосування віддалених і менш релевантних сусідів (underfitting). Тому вибір оптимального k є критично важливим і зазвичай здійснюється за допомогою перехресної валідації. Ще одним важливим аспектом є попередня обробка даних. Оскільки метод базується на обчисленні відстаней, різні масштаби ознак можуть спотворити результати. Наприклад, якщо одна з ознак варіюється в межах 0–1, а інша – 0–1000, остання домінуватиме у розрахунку відстаней. Тому перед застосуванням k-NN часто виконують нормалізацію або стандартизацію даних [8, 11-14].

Метод k-ближчих сусідів має як переваги, так і недоліки. До основних переваг належать простота реалізації, відсутність фази навчання, гнучкість щодо типу задачі (класифікація або регресія), а також здатність працювати з багатокласовими проблемами. Алгоритм легко реалізується, добре інтерпретується та часто слугує еталоном для оцінки складніших моделей. Проте існують і суттєві недоліки. Перш за все – обчислювальна складність. Для кожного нового прикладу необхідно порівнювати його з усіма об'єктами навчальної вибірки, що робить метод непридатним для роботи з великими наборами даних без застосування спеціалізованих структур (наприклад, kd-дерев, ball-дерев або пошуку за допомогою індексів). Крім того, k-NN погано працює в задачах з великою кількістю ознак (curse of dimensionality), оскільки відстані між об’єктами в багатовимірному просторі втрачають інформативність. Ще один недолік – чутливість до вибору метрики та масштабу ознак. Важливо також враховувати, що k-NN не забезпечує

інтерпретації моделі у вигляді формул чи коефіцієнтів, що може бути обмеженням у прикладних задачах, де потрібна пояснюваність [11-14].

Попри зазначені обмеження, метод k-ближчих сусідів широко використовується в прикладних задачах аналізу даних. У розпізнаванні образів він застосовується для класифікації рукописних цифр, наприклад, у відомому наборі MNIST. В обробці текстів – для класифікації документів і виявлення спаму. В системах рекомендацій – для передбачення вподобань користувачів на основі їхньої схожості з іншими. У медицині – для класифікації пацієнтів за симптомами або результатами аналізів. Завдяки простоті та ефективності, k-NN залишається важливою частиною інструментарію спеціаліста з машинного навчання, особливо на етапах первинного прототипування моделей або як базовий метод для порівняння з більш складними алгоритмами.

2.4. Дерево рішень

Дерево рішень (Decision Tree) – це інтерпретований і широко використовуваний алгоритм машинного навчання, який застосовується для задач класифікації та регресії. Його основна ідея полягає у побудові моделі у вигляді дерева, де кожна внутрішня вершина (вузол) відповідає певній умові (правилу розбиття), кожна гілка – результату перевірки цієї умови, а кожен лист – передбачуваному значенню (класу або числу). Така структура дозволяє приймати рішення шляхом послідовного переходу від кореня до листа, слідуючи умовам у вузлах. Дерева рішень є інтуїтивно зрозумілими, легко інтерпретуються і не вимагають масштабування ознак, що робить їх зручним інструментом у багатьох практичних задачах [8, 11-14].

Побудова дерева рішень здійснюється шляхом рекурсивного поділу простору ознак на підмножини з максимальною «чистотою» щодо цільової змінної. Алгоритм починає з усього навчального набору і на кожному кроці вибирає ознаку та поріг поділу, які найкраще розділяють дані. Критерієм

якості розбиття для задач класифікації зазвичай виступає ентропія або індекс Джині. Ентропія визначає рівень невпорядкованості у множині і обчислюється як [8, 13]:

$$H(S) = -\sum_{i=1}^{k} p_i log_2 p_i,$$

де p_i – частка об’єктів класу i у множині S.

Іншим популярним критерієм є індекс Джині:

$$G(S) = 1 - \sum_{i=1}^{k} p_i^2,$$

який також вимірює ступінь нечистоти вузла. Найкраще розбиття – те, яке дає найбільше зменшення цих метрик, тобто інформаційний приріст. Для задач регресії частіше використовують критерій середньоквадратичної помилки або середнього абсолютного відхилення [8, 11-14].

Розглянемо приклад: маючи набір даних про клієнтів банку, де потрібно передбачити, чи отримає особа кредит (клас «так» або «ні»), алгоритм спершу оцінює всі можливі ознаки, наприклад, дохід, вік, кредитну історію тощо, та вибирає ту, яка найкраще ділить клієнтів на підмножини, де більшість мають однаковий клас. Наприклад, перший розподіл може бути «дохід > 50 000», далі – «вік < 35», і так далі, поки кожен лист дерева не стане максимально однорідним, або не буде досягнуто глибину, яка запобігає перенавчанню [8, 12].

Однією з головних переваг дерева рішень є простота інтерпретації. Кожне правило в дереві відповідає послідовності логічних умов, що легко пояснити кінцевим користувачам. Наприклад, у медичній діагностиці таке дерево може виглядати як набір простих запитань, що ведуть до діагнозу, який ґрунтується на симптомах. Крім того, дерева рішень не потребують попередньої нормалізації або стандартизації даних і можуть працювати з як числовими, так і категоріальними ознаками [11, 12].

До недоліків методу слід віднести схильність до перенавчання (overfitting), особливо при побудові глибоких дерев. Такі дерева надто точно

відображають навчальні дані, але демонструють погану узагальненість на нових прикладах. Щоб зменшити цю проблему, застосовують обрізання дерева (pruning) – процес видалення гілок, які несуттєво покращують точність. Існують два типи обрізання: попереднє (обмеження глибини дерева, мінімальної кількості об'єктів у листі тощо) і післяпобудовне (видалення гілок після створення повного дерева за допомогою перевірки на валідаційній вибірці) [12, 13].

Ще одна проблема дерев рішень – нестабільність. Невелика зміна у даних може призвести до побудови абсолютно іншого дерева. Це обмеження частково вирішується методами ансамблю, наприклад, випадковими лісами (Random Forest), де поєднуються багато дерев, кожне з яких побудоване на випадкових підмножинах даних та ознак, а рішення приймається голосуванням або усередненням [13, 14].

Алгоритми побудови дерев варіюються залежно від реалізації. Найвідомішими є ID3 (Iterative Dichotomiser 3), C4.5 і CART (Classification and Regression Tree). ID3 використовує ентропію як критерій розбиття, C4.5 розширює його, дозволяючи працювати з неповними даними та обрізанням дерева, а CART застосовується як для класифікації, так і для регресії, використовуючи індекс Джині або середньоквадратичну помилку.

У практичних задачах дерева рішень ефективні у багатьох сферах:

- у кредитному скорингу, де вони дозволяють приймати рішення щодо кредитоспроможності клієнта;
- у медицині – для підтримки діагностичних рішень;
- в аналізі ризиків, сегментації ринку, обробці текстів та інших галузях.

Їх легка візуалізація та інтерпретованість роблять дерева рішень популярними серед аналітиків, які працюють із бізнесом або непрофільними користувачами, для яких прозорість моделі має вирішальне значення [14].

2.5. Метод випадкового лісу

Random Forest є ансамблевим методом, що базується на об’єднанні великої кількості дерев рішень, які будуються на випадкових підмножинах даних та ознак. Цей метод поєднує простоту та інтерпретованість окремих дерев із високою прогностичною здатністю ансамблевих моделей, забезпечуючи високу стабільність і надійність [8, 11-14].

Основна ідея методу полягає в побудові великої кількості дерев рішень під час тренування і виведенні результату шляхом агрегації відповідей усіх дерев. У задачах класифікації прогноз визначається голосуванням більшості (majority vote), а в задачах регресії – усередненням (average) прогнозів усіх дерев. Таким чином, випадковий ліс зменшує варіативність моделі (variance), яка властива окремим деревам, не збільшуючи при цьому значно упередженість (bias) [8, 11-14].

Кожне дерево у випадковому лісі тренується на випадковій підмножині навчального набору, яка формується методом бутстрепінгу (bootstrap sampling), тобто вибором з поверненням. Крім того, при кожному розгалуженні дерева розглядається лише випадкова підмножина ознак, що додає ще одну ступінь випадковості. Це дозволяє деревам бути менш корельованими між собою, що в свою чергу підвищує ефективність ансамблю. Якщо в навчальному наборі $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, де $x_i \in \mathbb{R}^d$ – вектор ознак, а y_i – цільова змінна, тоді для кожного дерева T_j вибирається випадкова підмножина $D_j \subset D$, а також випадковий набір ознак $F_j \subset \{1,...,d\}$, які розглядаються для кожного розбиття у вузлах.

Для класифікації фінальний прогноз моделі можна подати як:

$$\hat{y} = mode(T_1(x), T_2(x), \ldots, T_k(x)),$$

де $T_j(x)$ – прогноз j-го дерева.

У випадку регресії:

$$\hat{y} = \frac{1}{k}\sum_{j=1}^{k} T_j(x) y.$$

Однією з важливих переваг випадкового лісу є можливість оцінки важливості ознак (feature importance). Під час побудови дерев обчислюється, наскільки сильно кожна ознака зменшує критерій нечистоти (наприклад, індекс Джині або ентропію), і це зменшення агрегується по всіх деревах. Отримані значення можуть бути використані для ранжування або відбору ознак, що особливо корисно в задачах з великою кількістю змінних [13, 14].

Крім того, метод випадкового лісу має вбудовану оцінку помилки узагальнення через механізм Out-Of-Bag (OOB). Оскільки кожне дерево будується на бутстреп-підмножині, приблизно третина об'єктів залишається поза нею. Ці об'єкти можуть бути використані для оцінки якості моделі без потреби у виділенні окремої валідаційної вибірки. OOB-помилка визначається як середня помилка передбачення для всіх об'єктів, які не входили в навчальну вибірку при побудові відповідного дерева:

$$OOB\,Error = \frac{1}{n}\sum_{i=1}^{n} 1(\hat{y}_i^{OOB} \neq y_i),$$

де $\hat{y}_i^{OOB}$ – передбачення, отримане на основі лише тих дерев, які не бачили об'єкт x_i під час навчання.

Серед основних переваг випадкового лісу:

1) Висока точність: завдяки агрегуванню багатьох слабких моделей, випадковий ліс досягає високої точності навіть на складних задачах.

2) Стійкість до перенавчання: об'єднання декількох дерев з випадковими підмножинами ознак і даних зменшує варіацію моделі.

3) Універсальність: працює як з задачами класифікації, так і регресії.

4) Робота з пропущеними значеннями: метод досить нечутливий до окремих пропущених або шумових даних.

5) Визначення важливості ознак: дає змогу з’ясувати, які ознаки найбільше впливають на результат моделі.

6) Вбудована валідація: OOB-оцінка дозволяє контролювати якість моделі без окремого тестового набору.

Однак метод має і свої недоліки:

1) Мала інтерпретованість: на відміну від окремого дерева рішень, випадковий ліс складається з великої кількості дерев, тому його важко візуалізувати та пояснити кінцевим користувачам.

2) Високі обчислювальні витрати: побудова сотень або тисяч дерев вимагає значних ресурсів пам’яті та часу, особливо для великих наборів даних.

3) Неефективність для надто великих ознакових просторів без попереднього відбору: при великій кількості неінформативних змінних можливе зниження точності моделі.

Метод випадкового лісу використовується у багатьох прикладних сферах, включаючи біоінформатику, фінансовий аналіз, медичну діагностику, обробку природної мови, виявлення шахрайства, прогнозування ризиків і багато інших. Завдяки своїй простоті у використанні та стабільним результатам, він входить до числа найбільш популярних алгоритмів машинного навчання [12-14].

Підсумовуючи, випадковий ліс є ефективним та універсальним алгоритмом, що забезпечує високу точність та стійкість моделей, при цьому залишаючись гнучким і здатним працювати з різноманітними типами даних. Його здатність до автоматичної оцінки важливості ознак і внутрішньої перевірки якості робить його незамінним інструментом у сучасній аналітиці даних.

2.6. Висновки до розділу 2

У ході дослідження алгоритмів машинного навчання було розглянуто п’ять найбільш поширених методів, які широко використовуються у задачах класифікації та прогнозування: простий байєсівський класифікатор, метод опорних векторів (SVM), метод k-ближчих сусідів (k-NN), дерево рішень та метод випадкового лісу. Кожен із них має свої особливості, переваги й обмеження, що визначає доцільність їх використання в залежності від конкретної задачі та типу даних.

Простий байєсівський класифікатор характеризується високою швидкістю обробки та ефективністю на великих обсягах даних із незалежними ознаками. Його головною перевагою є простота реалізації та інтерпретації результатів, однак сильне припущення про незалежність ознак обмежує його застосування у випадках, коли між ознаками існують складні залежності.

Метод опорних векторів (SVM) демонструє високу точність на задачах з високою розмірністю ознак, зокрема завдяки використанню ядрових функцій. SVM добре працює з даними, що мають чіткий розподіл класів, і є стійким до перенавчання. Водночас його недоліком є складність налаштування параметрів та обчислювальна витратність при роботі з великими наборами даних.

Метод k-ближчих сусідів (k-NN) є інтуїтивно зрозумілим і не потребує навчання моделі, проте чутливий до вибору метрики відстані, значення параметра k, а також до масштабу ознак. Його перевагою є здатність адаптуватися до локальної структури даних, але при великій кількості ознак або зростанні обсягу вибірки ефективність значно знижується через високі обчислювальні витрати.

Дерева рішень є зручними для інтерпретації і візуалізації моделей. Вони дозволяють легко виявляти взаємозв’язки між ознаками та результатами, але схильні до перенавчання, особливо при великій глибині

дерев. Метод є чутливим до змін у даних і не завжди забезпечує високу точність класифікації у порівнянні з ансамблевими підходами.

Метод випадкового лісу є прикладом ансамблевої моделі, яка значно підвищує точність та стійкість у порівнянні з окремими деревами рішень. Завдяки використанню бутстрепінгу та випадкового вибору ознак під час побудови кожного дерева, випадковий ліс демонструє високу здатність до узагальнення, стійкість до шуму та меншу схильність до перенавчання. Проте ця методика поступається у швидкості обчислень та інтерпретованості моделі.

Загалом, проведене дослідження підтвердило, що вибір алгоритму машинного навчання повинен залежати від специфіки задачі, властивостей даних (кількість ознак, розмір вибірки, тип розподілу), вимог до точності, швидкості роботи та інтерпретованості моделі. У більшості реальних сценаріїв доцільним є експериментальне порівняння кількох методів для вибору оптимального рішення.

РОЗДІЛ 3
РОЗРОБКА КОМП’ЮТЕРНОЇ МОДЕЛІ КЛАСИФІКАЦІЇ

У сучасному цифровому середовищі проблема фільтрації небажаних повідомлень набуває особливої актуальності. Зі зростанням обсягів інформації, що циркулює в електронних комунікаціях, зокрема в системах обміну текстовими повідомленнями, значно підвищується ймовірність розповсюдження спаму – небажаного контенту, що має рекламний, шахрайський або інший шкідливий характер. Це створює загрози не лише для зручності користувача, а й для безпеки інформаційних систем загалом.

З метою підвищення ефективності виявлення спаму та зменшення навантаження на користувача виникає необхідність впровадження автоматизованих систем класифікації, заснованих на алгоритмах машинного навчання. Одним із найперспективніших напрямів у цьому контексті є розробка моделей класифікації текстів, здатних відрізняти спам-повідомлення від звичайних ("ham") на основі їх змісту [11-15].

У межах цього дослідження було поставлено завдання розробити комп’ютерну модель класифікації SMS-повідомлень із метою автоматичного виявлення спаму. Для цього було використано п’ять базових, але ефективних алгоритмів машинного навчання, що відрізняються підходами до навчання, побудови гіпотез та узагальнення: наївний байєсівський класифікатор, дерево рішень, метод k-найближчих сусідів, метод опорних векторів та логістична регресія.

3.1. Вибір програмного забезпечення для реалізації моделі

У контексті побудови комп’ютерної моделі класифікації для виявлення спаму в SMS-повідомленнях виникає потреба у ретельному виборі програмного забезпечення, яке забезпечить необхідні функціональні можливості, ефективність обробки даних, масштабованість, доступ до

сучасних алгоритмів машинного навчання та зручність для подальшого аналізу результатів. Правильний вибір програмного забезпечення безпосередньо впливає на точність, надійність і продуктивність розроблюваної моделі.

У процесі вибору програмного забезпечення було враховано низку ключових критеріїв. По-перше, важливим є рівень функціональності інструменту: він має підтримувати реалізацію широкого спектра алгоритмів машинного навчання, зокрема моделей класифікації, таких як наївний байєсівський класифікатор, дерева рішень, метод k-найближчих сусідів, метод опорних векторів, логістична регресія тощо. Крім того, програмне середовище повинно забезпечувати підтримку для роботи з текстовими даними, зокрема засобами попередньої обробки, векторизації, нормалізації, виділення ознак та візуалізації результатів [11-15].

Другим критичним критерієм є ефективність роботи з великими обсягами даних. Сучасні набори даних часто містять тисячі або навіть мільйони записів, тому важливо, щоб обране середовище дозволяло ефективно керувати пам’яттю, мати інструменти для паралельних обчислень та підтримку для роботи з матрицями великої розмірності. Такі можливості є необхідною умовою для оптимізації часу виконання обчислювальних задач та забезпечення масштабованості обраних рішень [13-16].

Особливу увагу слід приділяти зручності використання, доступності документації та навчальних матеріалів. Інструменти, які мають інтуїтивно зрозумілий інтерфейс та зрозумілу структуру, дозволяють скоротити час на освоєння й зосередитися на дослідницьких та інженерних аспектах проєкту. У цьому контексті важливо оцінювати можливості середовища програмування в частині інтеграції з популярними бібліотеками, наявності широкої спільноти користувачів та активної підтримки [13-16].

Розглядаючи потенційні мови програмування, доцільно провести порівняльний аналіз таких мов як Python, C/C++, Java та C#. Python є найпоширенішою мовою в сфері аналізу даних та машинного навчання

завдяки своїй простоті, читабельності, активній спільноті та великому набору спеціалізованих бібліотек, таких як scikit-learn, pandas, NumPy, TensorFlow, Keras, PyTorch. Його інтерпретована природа дозволяє швидко тестувати гіпотези, а широке використання у сфері науки й освіти робить його зручним для дослідницької роботи [13-16].

Мови C і C++ мають значно вищу продуктивність, ніж Python, особливо в задачах низькорівневого доступу до ресурсів пам'яті або обчислення великого обсягу даних. Вони часто використовуються у системах, де критична швидкодія, наприклад у вбудованих системах, реальному часі, або для реалізації високопродуктивних бібліотек машинного навчання, таких як XGBoost чи LightGBM. Однак через складність синтаксису, потребу в ручному управлінні пам'яттю та вищий поріг входження ці мови менш зручні для початкових досліджень у сфері машинного навчання [13-16].

Java пропонує баланс між продуктивністю й простотою. Вона має стабільну об'єктно-орієнтовану структуру, активну екосистему, кросплатформеність та широкий набір бібліотек для обробки даних і машинного навчання (наприклад, Weka, Deeplearning4j, MOA). Java добре підходить для розгортання моделей у великих системах, але не така гнучка для швидкого прототипування, як Python [13-16].

Мова C# активно використовується в екосистемі .NET і має вбудовану підтримку засобів для розробки графічних інтерфейсів, роботи з базами даних і побудови серверних додатків. Для задач машинного навчання C# пропонує бібліотеки ML.NET, Accord.NET та CNTK (раніше Microsoft Cognitive Toolkit). Хоча екосистема C# у сфері машинного навчання менш розвинена, ніж у Python, вона підходить для задач, де важлива інтеграція з корпоративними застосунками Microsoft [13-16].

З огляду на викладені критерії, вибір мови програмування доцільно здійснювати залежно від цільової задачі, обсягу даних, очікуваної продуктивності та вимог до інтеграції з іншими системами. В рамках

побудови прототипу моделі класифікації перевага часто надається Python, оскільки ця мова має найбільш повноцінну підтримку як базових, так і просунутих алгоритмів машинного навчання, гнучкі засоби попередньої обробки текстових даних, а також інструменти для візуалізації результатів.

Наступним компонентом вибору є огляд наявних бібліотек у межах обраної мови. У Python набір бібліотек scikit-learn забезпечує реалізацію широкого спектра моделей машинного навчання, включаючи класифікацію, кластеризацію, регресію, моделі ансамблю, а також метрики якості, крос-валідацію та побудову пайплайнів. Pandas і NumPy забезпечують ефективну обробку структурованих даних і числових масивів, а Matplotlib та Seaborn – засоби візуалізації результатів [16-16].

Для текстової обробки важливу роль відіграє TfidfVectorizer – інструмент для перетворення текстових повідомлень у числове представлення на основі частоти термінів. Такий підхід дозволяє застосовувати класичні алгоритми машинного навчання до задач класифікації текстів. Крім того, використання бібліотеки Hugging Face datasets дозволяє зручно завантажувати стандартні датасети для тренування моделей, зокрема датасет SMS Spam Collection.

Візуалізація результатів є невід’ємною частиною дослідження. Для цього доцільно використовувати побудову матриць плутанини (confusion matrix), графіків ROC-кривих, а також діаграм розсіювання ознак. Це дозволяє не лише оцінити якість класифікації, а й дослідити структуру даних, виявити кореляції між ознаками, а також зрозуміти межі прийняття рішень моделями [15].

Отже, з огляду на широку функціональність, зручність у використанні, активну підтримку спільноти, а також наявність потужних бібліотек для прискорених математичних обчислень, обробки даних і реалізації алгоритмів машинного навчання, мова програмування Python вважається одним із найоптимальніших варіантів для реалізації реалізація комп’ютерної моделі класифікації для виявлення спаму в повідомленнях.

Також, вибір інтегрованого середовища розробки є важливим етапом при реалізації комп’ютерної моделі класифікації, оскільки воно має забезпечувати ефективне програмування, налагодження та тестування коду, а також підтримувати роботу з необхідними бібліотеками для обробки даних і машинного навчання. Серед численних IDE, які підтримують мови програмування для машинного навчання, особливу увагу варто приділити середовищам, які мають глибоку інтеграцію з Python, оскільки ця мова є найбільш підходящою для реалізації моделей завдяки своїй функціональності, простоті використання, великій кількості бібліотек та активній спільноті.

PyCharm виділяється як оптимальне рішення завдяки низці переваг. По-перше, це потужна підтримка Python, що включає інтелектуальне автозавершення коду, навігацію, рефакторинг та динамічну перевірку помилок, що значно підвищує продуктивність розробника. По-друге, PyCharm має вбудовану підтримку основних наукових та машиннонавчальних бібліотек, таких як NumPy, pandas, matplotlib, scikit-learn, TensorFlow, що є критично важливим для створення та оптимізації моделей [14, 15].

Додатково, PyCharm підтримує інтеграцію з системами контролю версій, наприклад Git, що полегшує командну розробку та управління змінами коду. Наявність професійних інструментів для налагодження та профілювання допомагає оптимізувати програмний код, підвищуючи якість і надійність розробленої моделі. Враховуючи всі ці фактори, PyCharm є найбільш доцільним середовищем для комплексної розробки, тестування та підтримки моделей машинного навчання, особливо у випадках, коли проєкт має складну структуру та потребує подальшого розвитку й інтеграції з іншими системами.

3.2. Реалізація комп’ютерної моделі класифікації

З розвитком цифрових технологій проблема спаму в електронних повідомленнях набула великого значення. Спам знижує ефективність електронного спілкування, завдає фінансових збитків і створює ризики безпеки. Одним з ефективних підходів до автоматичної фільтрації спаму є використання алгоритмів машинного навчання. Розглянемо процес створення та оцінки моделей класифікації спаму на основі векторизації тексту повідомлень та подальшого застосування різних методів навчання з учителем.

Для вирішення цієї задачі класифікації було обрано п’ять поширених та ефективних методів машинного навчання: простий байєсівський класифікатор (Naive Bayes), дерево рішень (Decision Tree), метод k-найближчих сусідів (K-Nearest Neighbors), метод опорних векторів (Support Vector Machine) та логістична регресія (Logistic Regression). Реалізація була виконана мовою Python із використанням бібліотек scikit-learn, pandas, seaborn та matplotlib. Для забезпечення репрезентативності вибірки використовувався публічний датасет SMS Spam Collection, що містить 5574 англомовні текстові повідомлення, кожне з яких має мітку: ham (не спам) або spam. Набір завантажено з відкритого репозиторію datasets від Hugging Face.

Першим кроком стала підготовка даних. Текстові повідомлення з датасету було розділено на навчальну та тестову вибірки у співвідношенні 70/30 з фіксацією випадковості для відтворюваності результатів (random_state=42). Перед застосуванням моделей машинного навчання, текст було векторизовано за допомогою методу TF-IDF (Term Frequency - Inverse Document Frequency), що дозволило перетворити текстові повідомлення у числові вектори, які можна подавати на вхід алгоритмів класифікації [11].

Наступним етапом стало навчання обраних моделей на навчальній вибірці та їх тестування на валідаційній. Для кожного алгоритму було розраховано основні метрики якості класифікації: точність (accuracy),

повнота (recall), точність (precision) та f1-міра (f1-score). Крім того, було згенеровано матриці плутанини (confusion matrices), побудовано ROC-криві та діаграми розсіювання для візуалізації поведінки моделей. Повний код комп'ютерної моделі наведено в додатку А, а результати виконання на рис. 3.1.

```
--- Naive Bayes ---
Accuracy: 0.9731022115959355
              precision    recall  f1-score   support

         ham       0.97      1.00      0.98      1447
        spam       1.00      0.80      0.89       226

    accuracy                           0.97      1673
   macro avg       0.98      0.90      0.94      1673
weighted avg       0.97      0.97      0.97      1673

--- Decision Tree ---
Accuracy: 0.9689181111775254
              precision    recall  f1-score   support

         ham       0.98      0.99      0.98      1447
        spam       0.91      0.85      0.88       226

    accuracy                           0.97      1673
   macro avg       0.95      0.92      0.93      1673
weighted avg       0.97      0.97      0.97      1673

--- K-Nearest Neighbors ---
Accuracy: 0.9007770472205618
              precision    recall  f1-score   support

         ham       0.90      1.00      0.95      1447
        spam       1.00      0.27      0.42       226

    accuracy                           0.90      1673
   macro avg       0.95      0.63      0.68      1673
weighted avg       0.91      0.90      0.87      1673

--- SVM (SVC linear) ---
Accuracy: 0.9832635983263598
              precision    recall  f1-score   support

         ham       0.99      0.99      0.99      1447
        spam       0.96      0.92      0.94       226

    accuracy                           0.98      1673
   macro avg       0.97      0.95      0.96      1673
weighted avg       0.98      0.98      0.98      1673

--- Logistic Regression ---
Accuracy: 0.9617453676031081
              precision    recall  f1-score   support

         ham       0.96      1.00      0.98      1447
        spam       0.97      0.74      0.84       226

    accuracy                           0.96      1673
   macro avg       0.97      0.87      0.91      1673
weighted avg       0.96      0.96      0.96      1673
```

Рис. 3.1. Порівняння алгоритмів машинного навчання для задачі класифікації повідомлень

Найвищу точність продемонстрував метод SVM із лінійним ядром, який досяг значення accuracy на рівні 98,3%. Він показав високі значення precision і recall для обох класів, що свідчить про його здатність однаково добре розпізнавати як спам, так і не спам повідомлення. ROC-крива для цього методу виявилася найбільш наближеною до верхнього лівого кута координатної площини, що підтверджує високу дискримінаційну здатність моделі (рис. 3.2).

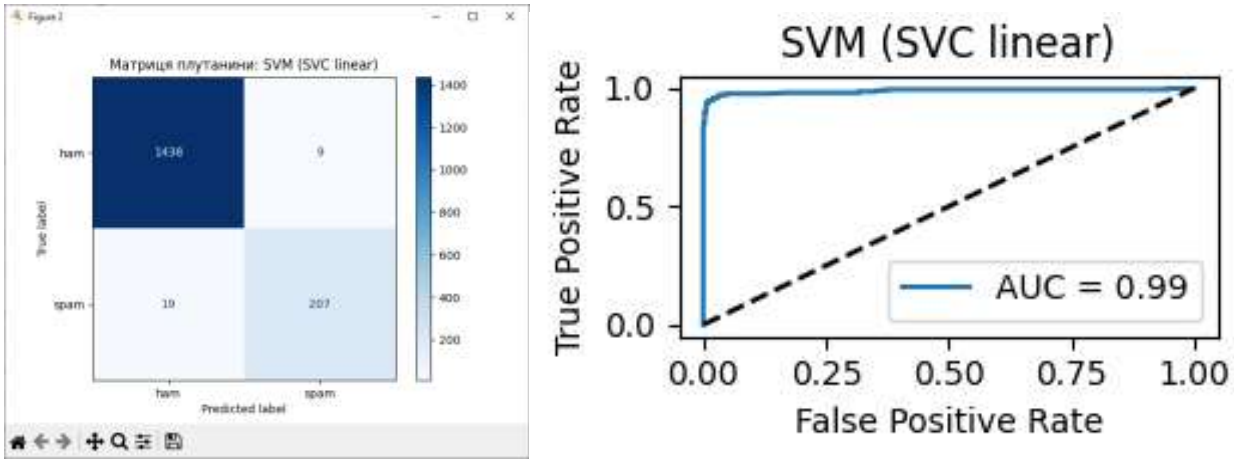

Рис. 3.2. Алгоритм опорних векторів

Простий байєсівський класифікатор (рис. 3.3) досяг точності 97,3%, показавши надзвичайно високу чутливість (recall) для класу «ham» (1,00), однак при цьому мав дещо знижену recall для класу «spam» (0,80). Цей метод виявився надзвичайно ефективним і простим у реалізації, особливо для задач текстової класифікації, де припущення про незалежність ознак виявляється прийнятним. З практичної точки зору, ця модель є найпридатнішою у випадках, коли потрібна швидка і досить точна оцінка без надмірних обчислювальних витрат.

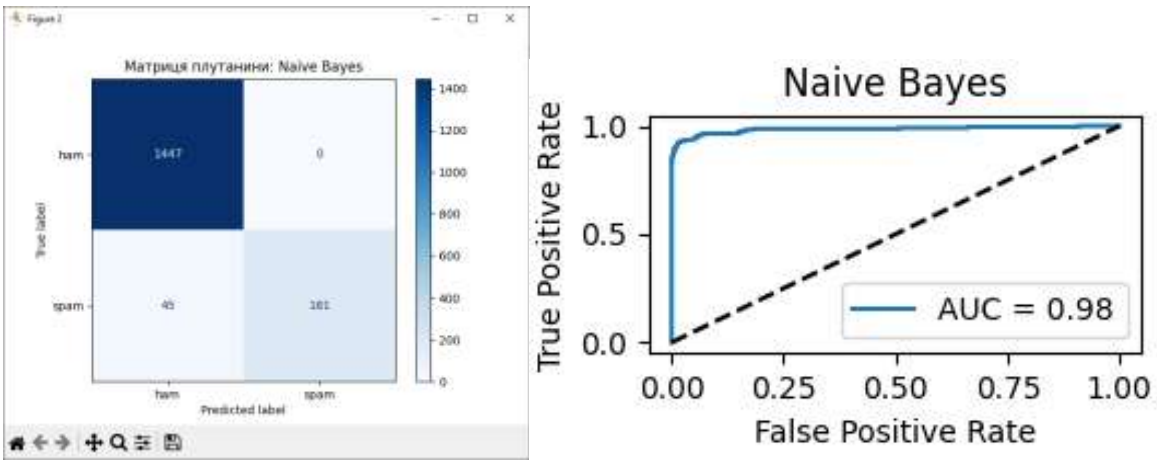

Рис. 3.3. Простий байєсівський класифікатор

Дерево рішень (Decision Tree) також досягло високих результатів, з точністю 96,9%. Його сильна сторона – інтерпретованість, оскільки побудоване дерево можна візуалізувати, а правила класифікації – легко пояснити. Проте модель показала меншу recall для класу «spam» (0,85), що може бути критично в умовах, коли пропуск спам повідомлень неприпустимий (рис. 3.4).

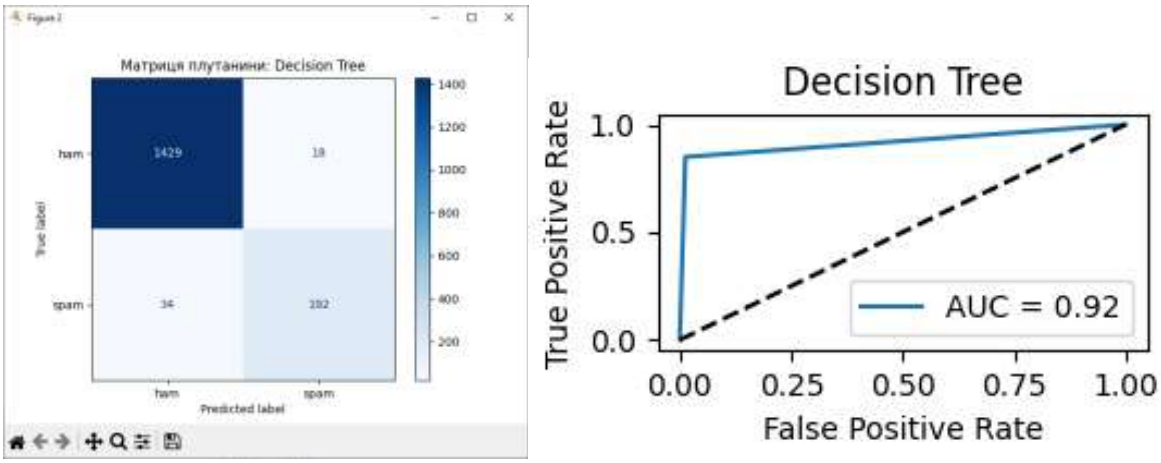

Рис. 3.4. Алгоритм дерева пошуку рішень

Метод k-найближчих сусідів виявився найменш ефективним серед розглянутих – із точністю лише 90,1% (рис. 3.5). Хоча модель показала ідеальну recall для класу «ham» (1,00), вона виявила значні труднощі з класифікацією спаму, маючи recall лише 0,27. Це може бути пояснено

високою чутливістю методу до розподілу даних у векторному просторі та недостатньою здатністю до узагальнення, особливо в умовах нерівномірного розподілу класів або за наявності великої кількості нерелевантних ознак.

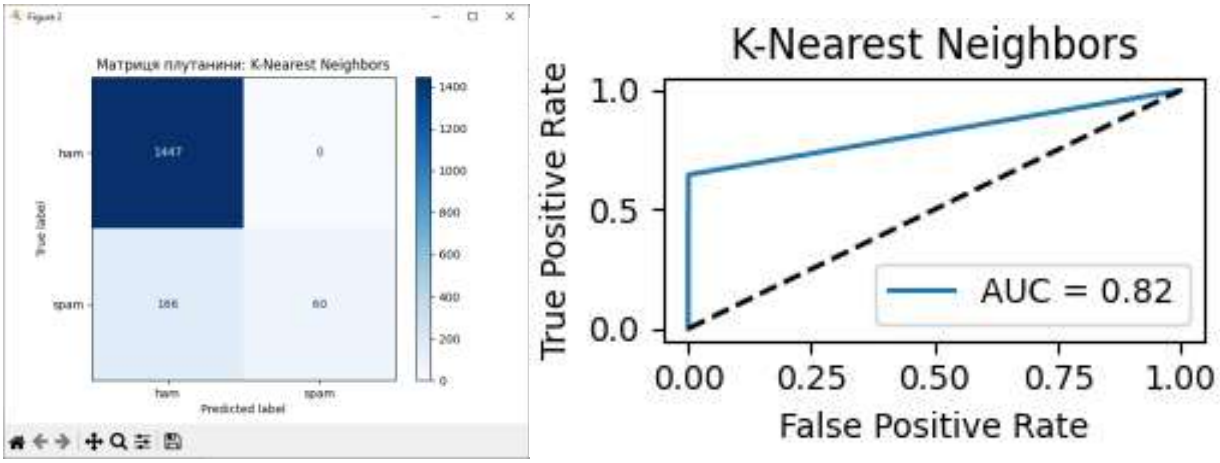

Рис. 3.5. Метод k-найближчих сусідів

Логістична регресія показала точність 96,2% і непогану збалансованість показників precision та recall для обох класів (рис. 3.6). Хоча ця модель поступається SVM у точності, її простота та швидкість навчання робить її конкурентоспроможною альтернативою для реальних додатків.

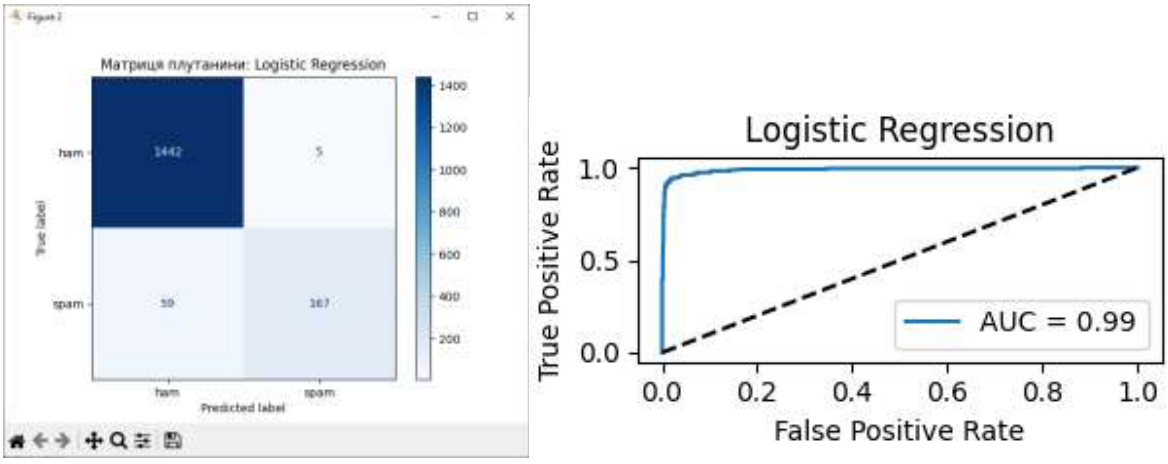

Рис. 3.6. Метод k-найближчих сусідів

Найбільшу площу під кривою (AUC) продемонстрували моделі SVM та Naive Bayes, що додатково підтверджує їхню ефективність.

Крім класифікаційних показників, було проведено аналіз кореляційної матриці на основі найчастотніших ознак TF-IDF. Було обрано 10 найбільш інформативних ознак, за якими обчислено коефіцієнти кореляції з мітками класів. Це дозволило краще зрозуміти вплив окремих термінів на ймовірність класифікації повідомлення як спаму. Побудована теплова карта (heatmap) дозволила візуалізувати ці взаємозв'язки (рис. 3.7).

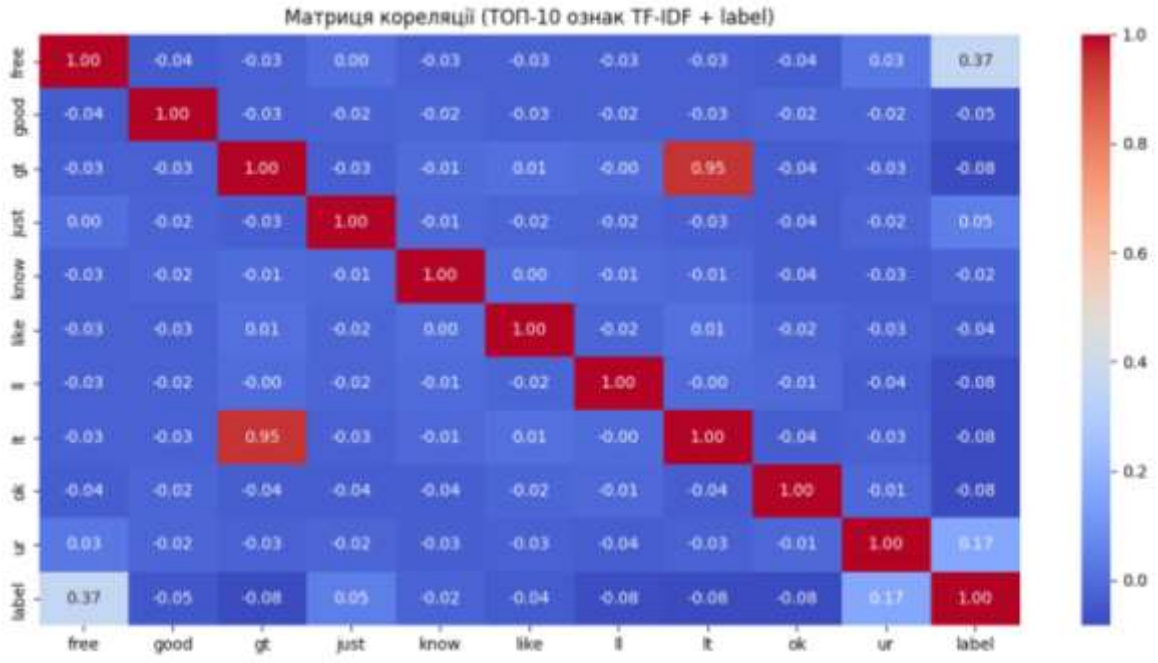

Рис. 3.7. Матриця кореляції

Узагальнюючи результати, можна стверджувати, що найкращим методом класифікації для задачі фільтрації SMS-спаму є метод опорних векторів (SVM) із лінійним ядром, що забезпечує найвищу точність, збалансованість метрик та стабільність при зміні параметрів. Альтернативою за простотою та швидкістю є простий байєсівський класифікатор, який також демонструє високу продуктивність. Інші методи, такі як дерева рішень та логістична регресія, мають прийнятні характеристики і можуть використовуватись у випадках, де важлива інтерпретованість або швидкість роботи моделі. Метод k-найближчих сусідів, попри простоту реалізації,

виявився неефективним для даної задачі через низьку чутливість до класу спаму. Зведена інформація наведена у таблиці 3.1.

Таблиця 3.1

Порівняння алгоритмів машинного навчання для задачі класифікації повідомлень

Алгоритм	Точність (Accuracy)	Precision (spam)	Recall (spam)	F1-Score (spam)
Naive Bayes	0.9731	1.00	0.80	0.89
Decision Tree	0.9689	0.91	0.85	0.88
K-Nearest Neighbors	0.9008	1.00	0.27	0.42
Support Vector Machine (SVM)	0.9833	0.96	0.92	0.94
Logistic Regression	0.9617	0.97	0.74	0.84

Таким чином, розроблена комп’ютерна модель класифікації є результатом комплексного підходу до обробки текстових даних, що поєднує попередню обробку, векторизацію, використання кількох моделей, візуалізацію результатів та аналіз кореляції. Запропонований підхід може бути використаний як основа для побудови реальних систем фільтрації електронних повідомлень, чат-ботів з виявлення токсичного контенту, а також у задачах класифікації відгуків клієнтів або аналізу тональності текстів.

3.3. Висновки до розділу 3

У розділі було проведено всебічний аналіз критеріїв вибору програмного забезпечення для реалізації комп’ютерної моделі класифікації SMS-повідомлень із метою автоматичного виявлення спаму. Враховуючи особливості поставленого завдання, було обґрунтовано пріоритетність використання мови програмування Python, яка відзначається високою функціональністю, доступністю широкого спектру бібліотек для обробки

даних та машинного навчання, простотою синтаксису та значною підтримкою з боку спільноти розробників.

Також детально розглянуто вибір інтегрованого середовища розробки, серед яких PyCharm визначено як найбільш оптимальне рішення. Це пов’язано з його розвиненими інструментами для роботи з Python, підтримкою наукових бібліотек, зручним управлінням віртуальними середовищами, а також можливостями для налагодження, тестування та контролю версій, що є критично важливими для ефективної розробки складних моделей машинного навчання.

У другій частині розділу було наведено приклади реалізації класифікаційних моделей із застосуванням різних алгоритмів машинного навчання, а також проведено їх порівняльний аналіз за показниками точності, precision, recall та F1-метриками. Отримані результати демонструють, що модель на основі Support Vector Machine (SVM) має найвищі показники, що свідчить про доцільність подальшого використання та вдосконалення саме цього алгоритму в контексті автоматичної класифікації текстових повідомлень.

Таким чином, розглянуті методи вибору інструментів розробки та впровадження моделей машинного навчання створюють міцну основу для подальшого проектування, оптимізації та масштабування комп’ютерної моделі класифікації, що є ключовим етапом у розробці систем автоматичного виявлення спаму.

ВИСНОВКИ

У кваліфікаційній бакалаврській роботі було проведено комплексне дослідження алгоритмів машинного навчання. Результати виконаної роботи дозволяють сформулювати низку узагальнюючих висновків.

У першому розділі розглянуто фундаментальні поняття машинного навчання, класифіковано основні типи задач, описано принципи побудови моделей та окреслено типову послідовність розв’язання задачі навчання. Особливу увагу приділено математичному формулюванню задачі класифікації, функціям втрат, ролі регуляризації та методам уникнення перенавчання. Важливим аспектом є також вибір метрик оцінювання та забезпечення якості моделі після її впровадження.

У другому розділі здійснено порівняльний аналіз найбільш поширених алгоритмів класифікації: простого байєсівського класифікатора, методу опорних векторів (SVM), методу k-ближчих сусідів (k-NN), дерева рішень та випадкового лісу. Для кожного з методів визначено переваги, недоліки та сфери застосування. Аналіз показав, що вибір моделі має базуватись на характеристиках вхідних даних, обмеженнях обчислювальних ресурсів і вимогах до інтерпретованості моделі. Зазначено доцільність експериментального порівняння моделей на практичних задачах для пошуку оптимального підходу.

У третьому розділі обґрунтовано вибір програмного забезпечення для побудови комп’ютерної моделі класифікації SMS-повідомлень. Обрано мову програмування Python та інтегроване середовище розробки PyCharm як оптимальні інструменти завдяки їхній функціональності, підтримці бібліотек для машинного навчання та зручності налагодження. Реалізовано класифікаційні моделі за допомогою п’яти алгоритмів, проведено експериментальне дослідження та оцінено результати за точністю, precision, recall і F1-метрикою. Найкращі результати показала модель на основі Support Vector Machine, що свідчить про її ефективність у задачах текстової

класифікації.

Перспективи подальших досліджень алгоритмів машинного навчання насамперед пов’язані з використанням методів глибокого навчання для підвищення ефективності класифікації текстових повідомлень. Особливу увагу доцільно приділити нейронним мережам сучасної архітектури, таким як рекурентні (RNN), згорткові (CNN) та трансформери (зокрема BERT), які здатні враховувати семантичні зв’язки між словами та контекст повідомлень. Застосування таких моделей може суттєво покращити точність і надійність систем автоматичного виявлення спаму, а також забезпечити адаптацію до нових шаблонів шкідливих повідомлень завдяки високій здатності до узагальнення.

Узагальнюючи результати, можна стверджувати, що ефективність застосування алгоритмів машинного навчання значною мірою визначається особливостями даних та специфікою задачі. Поєднання теоретичних знань, практичного досвіду та інструментальних засобів дозволяє створювати ефективні моделі для реальних прикладних задач, зокрема для автоматичної класифікації повідомлень з метою виявлення спаму. Отримані результати створюють підґрунтя для подальших досліджень у сфері інтелектуальних систем обробки текстової інформації, оптимізації моделей та розробки масштабованих програмних рішень.

ПЕРЕЛІК ПОСИЛАНЬ

1. Гавриленко С. Ю. Машинне навчання [Електронний ресурс] : конспект лекцій / Гавриленко С. Ю. ; Нац. техн. ун-т "Харків. політехн. ін-т". – Харків : НТУ “ХПІ”, 2024. – 232 с.

2. Кононова К. Ю. Машинне навчання – методи та моделі. Базовий підручник. – Запоріжжя : ЗНУ, 2023.

3. Звенігородський О. С., Зінченко О. В., Чичкарьов Є. А., Кисіль Т. М. Штучний інтелект. Вступний курс. – Київ : ДУІКТ, 2022. – 193 с.

4. Кисіль Т. М., Звенігородський О. С., Фесенко М. А. Основи штучного інтелекту. Методичні рекомендації. – Київ : ДУІКТ, 2022.

5. Григоров О. В., Аніщенко Г. О., Стрижак В. В. та ін. Штучний інтелект. Машинне навчання // Автомобіль і електроніка. Сучасні технології : зб. наук. пр. – Харків : ХНАДУ, 2019. – Вип. 15.

6. Google Machine Learning Crash Course. – [Електронний ресурс]. – Режим доступу: https://developers.google.com/machine-learning/crash-course

7. GeeksforGeeks: Підручник з машинного навчання. – [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/machine-learning/

8. Мокін Б. І., Мокін О. Б. Машинне навчання та інтелектуальний аналіз даних: навч. посібник. Вінниця: ВНТУ, 2024. 263 с.

9. Шевченко А. І., Барановський С. В., Білокобильський О. В., Бодянський Є. В., Бомба А. Я. Стратегія розвитку штучного інтелекту в Україні: монографія. Київ: Інститут проблем штучного інтелекту НАН України, 2023. 150 с.

10. Застосування технологій штучного інтелекту у фізиці високих енергій: навч. посібник. Київ: КНУ, 2023. 120 с.

11. Штучний інтелект і нейромережі: збірник самарі. Київ: Моноліт-Bizz, 2024. 216 с.

12. W3Schools: Підручник з машинного навчання. – [Електронний

ресурс]. – Режим доступу: https://www.w3schools.com/python/python_ml_getting_started.asp

13. TutorialsPoint: Підручник з машинного навчання. – [Електронний ресурс]. – Режим доступу: https://www.tutorialspoint.com/machine_learning/index.htm

14. Kaggle: Вступ до машинного навчання. – [Електронний ресурс]. – Режим доступу: https://www.kaggle.com/learn/intro-to-machine-learning

15. Machine Learning for Everybody – Повний курс. – [Електронний ресурс]. – Режим доступу: https://www.youtube.com/watch?v=i_LwzRVP7bg

16. Python Machine Learning Tutorial (Data Science). – [Електронний ресурс]. – Режим доступу: https://www.youtube.com/watch?v=7eh4d6sabA0

17. Apple Machine Learning Research. – [Електронний ресурс]. – Режим доступу: https://machinelearning.apple.com/

18. MathWorks: Підручники та приклади з машинного навчання. – [Електронний ресурс]. – Режим доступу: https://www.mathworks.com/solutions/machine-learning/tutorials-examples.html

ДОДАТКИ

Додаток А

Комп’ютерна модель класифікації повідомлень

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd

from datasets import load_dataset
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import roc_curve, auc

# --- Завантаження даних ---
dataset = load_dataset("sms_spam")
texts = dataset['train']['sms']
labels = dataset['train']['label']

# --- Розбиття на train/test ---
X_train, X_test, y_train, y_test = train_test_split(texts, labels, test_size=0.3, random_state=42)

# --- Векторизація для класифікації ---
vectorizer = TfidfVectorizer(stop_words='english', max_df=0.9)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# --- Векторизація для кореляції (обмежено 10 ознак для компактності) ---
vectorizer_corr = TfidfVectorizer(stop_words='english', max_features=10)
X_vec_corr = vectorizer_corr.fit_transform(texts)
feature_names = vectorizer_corr.get_feature_names_out()
df_features = pd.DataFrame(X_vec_corr.toarray(), columns=feature_names)
df_features['label'] = labels

# Обчислення кореляційної матриці
corr = df_features.corr()

# --- Ініціалізація моделей ---
models = {
    "Naive Bayes": MultinomialNB(),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(n_neighbors=5),
```

```
    "SVM (SVC linear)": SVC(kernel='linear', probability=True, random_state=42),
    "Logistic Regression": LogisticRegression(max_iter=1000, random_state=42)
}

# --- Навчання моделей і оцінка ---
plt.figure(figsize=(12, 10))
plt.suptitle('ROC-криві моделей класифікації спаму', fontsize=16)

for i, (name, model) in enumerate(models.items(), 1):
    model.fit(X_train_vec, y_train)
    y_pred = model.predict(X_test_vec)
    acc = accuracy_score(y_test, y_pred)

    print(f"--- {name} ---")
    print("Accuracy:", acc)
    print(classification_report(y_test, y_pred, target_names=['ham', 'spam']))

    # Матриця плутанини
    cm = confusion_matrix(y_test, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['ham', 'spam'])
    disp.plot(cmap=plt.cm.Blues)
    plt.title(f'Матриця плутанини: {name}')
    plt.show()

    # ROC-крива
    if hasattr(model, "predict_proba"):
        y_scores = model.predict_proba(X_test_vec)[:, 1]
    elif hasattr(model, "decision_function"):
        y_scores = model.decision_function(X_test_vec)
    else:
        print(f"Пропущено ROC для {name} (немає predict_proba або decision_function)")
        continue

    if len(np.unique(y_scores)) == 1:
        print(f"Не можна побудувати ROC для {name}: всі оцінки однакові.")
        continue

    fpr, tpr, thresholds = roc_curve(y_test, y_scores)
    roc_auc = auc(fpr, tpr)

    plt.subplot(3, 2, i)
    plt.plot(fpr, tpr, label=f'AUC = {roc_auc:.2f}')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.title(name)
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.legend(loc='lower right')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

# --- Візуалізація матриці кореляції ---
```

```
plt.figure(figsize=(12, 10))
sns.heatmap(corr, annot=True, fmt=".2f", cmap='coolwarm', cbar=True)
plt.title("Матриця кореляції (ТОП-10 ознак TF-IDF + label)")
plt.show()

# --- Діаграма розсіювання для двох найчастотніших ознак ---
plt.figure(figsize=(10, 8))
if len(feature_names) >= 2:
    sns.scatterplot(
        data=df_features,
        x=feature_names[0],
        y=feature_names[1],
        hue='label',
        palette=['green', 'red'],
        alpha=0.7
    )
    plt.title(f"Діаграма розсіювання за ознаками '{feature_names[0]}' і '{feature_names[1]}'")
    plt.xlabel(feature_names[0])
    plt.ylabel(feature_names[1])
    plt.legend(title='label', labels=['ham', 'spam'])
    plt.show()
else:
    print("Замало ознак для побудови діаграми розсіювання.")
```