



Машинне навчання

21 лютого 2024 р.



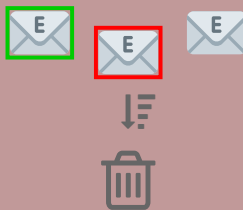
Штучний інтелект

Будь-яка техніка, яка дозволяє комп'ютерам імітувати поведінку людини



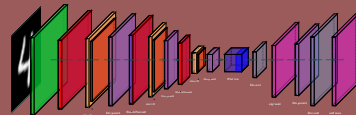
Машинне навчання

Можливість комп'ютера учитися не будучи явно запрограмованим



Глибинне навчання

Пошук шаблону в даних за допомогою нейронних мереж



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Кочура Ю. П., Гордієнко Ю. Г.

Машинне навчання

Занурення в машинне навчання

Навчальний посібник

Рекомендовано для здобувачів ступеня “магістр”, які навчаються за освітніми програмами «Комп’ютерні системи та мережі» спеціальність 123 «Комп’ютерна інженерія» та «Інженерія програмного забезпечення комп’ютерних систем» спеціальність 121 «Інженерія програмного забезпечення»

Електронне видання

Навчальний посібник перебуває на етапі написання. Остання доступна версія цього документа за QR-кодом:



Ми були б дуже вдячні за надану допомогу та пропозиції щодо наповнення та покращення цього документа. Повідомлення про проблеми надсилайте через github: <https://github.com/dml-book/dml/issues>

Чернетка. *Версія 0.1.1*

Зміст

1	Вступ	3
1.1	Що таке машинне навчання?	3
1.1.1	Навчання з учителем	4
1.1.2	Навчання без учителя	6
1.1.3	Навчання з частковим залученням учителя	7
1.1.4	Навчання з підкріпленням	7
1.2	Мотивація та інтуїція	8
1.2.1	Інжиніринг ознак	9

Розділ 1

Вступ

“Теоретично між теорією та практикою нема ніякої різниці. Але на практиці вона є.”

– Бенджамін Брюстер

1.1 Що таке машинне навчання?

Машинне навчання (Machine Learning, ML) – це галузь штучного інтелекту, що вивчає розробку та застосування алгоритмів, які дозволяють комп’ютеру вчитися робити прогнози чи приймати рішення на основі вхідних даних не будучи при цьому явно запрограмованим. Ці дані можуть бути отримані з реального середовища за допомогою сенсорів, створені вручну або згенеровані іншим алгоритмом.

Машинне навчання також можна розглядати як процес вирішення деякого практичного завдання шляхом алгоритмічного навчання **статистичної моделі** на наборі даних, що характеризує шаблон поставленого завдання.

Комп’ютери та обчислення допомагають нам досягати більш складних цілей і кращих результатів у вирішенні проблем, ніж ми могли б досягти самі. Однак, багато сучасних завдань вийшли за рамки обчислень через один основний обмежуючий фактор: традиційно, комп’ютери можуть дотримуватися лише конкретних вказівок/інструкцій, які їм дають.

Вирішення проблем з програмування вимагає написання конкретних покрокових інструкцій, які має виконувати комп’ютер. Ми називаємо ці кроки алгоритмами. У цьому випадку, комп’ютери можуть допомогти нам там, де ми:

1. Розуміємо як вирішити проблему.
2. Можемо описати проблему за допомогою чітких покрокових інструкцій, які комп’ютер може зрозуміти.

Методи контрольованого машинного навчання дозволяють комп’ютерам “учитися” на прикладах. Вирішення проблем із застосуванням машинного навчання вимагає виявлення деякого шаблону¹, а потім, коли такий шаблон готовий, дозволяють, наприклад, нейронній мережі вивчити карту переходів між вхідними та вихідними даними. Ця особливість відкриває нові типи проблем, де комп’ютери можуть допомогти нам у їх розв’язанні, за умови, коли ми:

1. Визначили шаблон проблеми.
2. Маємо достатньо даних, що ілюструють даний шаблон.

¹Пошук прикладів, що висвітлюють обидві сторони шаблону: вхід і вихід.

Принципова відмінність парадигм: класичного програмування (символьний штучний інтелект) та машинного навчання, показана на рисунку 1.1. Таким чином, для вирішення задачі за допомогою класичного програмування нам потрібно прописати чіткі покрокові інструкції нашого алгоритму. Далі, коли алгоритм реалізовано, ми можемо подати йому на вхід деякі дані, що характеризують окремий стан поставленої задачі, а на виході він згенерує нам деякі вихідні дані (відповіді). Коли мова йде про контрольоване машинне навчання, тут на вхід комп'ютерній програмі подаються приклади, що висвітлюють обидві сторони шаблону поставленої задачі: вхідні та очікувані вихідні дані (відповіді), а на виході отримуємо правило (алгоритм) за яким буде здійснюватись перехід між вхідними та вихідними даними.



Рис. 1.1: Відмінність класичного програмування від машинного навчання.

Для стислості, далі будуть використовуватись терміни **навчання** та **машинне навчання** як синоніми. З тієї ж причини часто буде використовуватись термін **модель**, під яким будемо мати на увазі **статистичну модель**.

За характером навчальних даних навчання буває з учителем (контрольоване), без учителя (неконтрольоване), з частковим залученням учителя (напівконтрольоване) та з підкріпленням (див. рис. 1.2).

1.1.1 Навчання з учителем

У цьому випадку ми працюємо з множиною даних, яку представлено наступним чином:

$$d = \{(\mathbf{X}^{(1)}, y^{(1)}), (\mathbf{X}^{(2)}, y^{(2)}), \dots, (\mathbf{X}^{(n)}, y^{(n)})\}, \quad (1.1)$$

де n – загальна кількість прикладів у наборі, $y^{(i)}$ – мітка i -го прикладу, як правило, $y^{(i)} \in \mathbb{R}$ або $y^{(i)} \in \mathbb{N}$.

Кожен елемент $\mathbf{X}^{(i)}$ множини (1.1) називається вектором вхідних ознак або прикладом. Вектор – це одновимірний масив. Одновимірний масив, у свою чергу, є упорядкованою та проіндексованою послідовністю значень. Довжина цієї послідовності, m , називається розмірністю вектора: $\mathbf{X}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \dots, x_m^{(i)})$.

Вектор ознак – це вектор, у якому кожен елемент $j = 1, \dots, m$ містить значення, що характеризує приклад. Кожне таке значення називається ознакою та позначається x_j . Запис $x_j^{(i)}$ означає, що ми розглядаємо ознаку j для i -го прикладу.





Контрольоване навчання Supervised learning	Напівконтрольоване навчання Semi-supervised learning	Неконтрольоване навчання Unsupervised learning	Навчання з підкріплення Reinforcement learning
Дані: (X, y) X – приклад, y – мітка	Дані: (X, y) та $X, \{ (X, y) \} < X $ X – приклад, y – мітка	Дані: X X – приклад, немає міток!	Дані: пари стан-дія
Мета – знайти функцію відображення $X \rightarrow y$	Мета – знайти функцію відображення або категорію $X \rightarrow y$	Мета – знайти правильну категорію.	Мета – максимізація загальної винагороди, отриманої агентом при взаємодії з навколишнім середовищем.
Приклад  Це є яблуко.	Приклад  Це є яблуко.	Приклад  Цей об'єкт схожий на інший.	Приклад  Їжте це, бо це зробить вас сильнішим.

Рис. 1.2: Види навчання за характером навчальних даних.

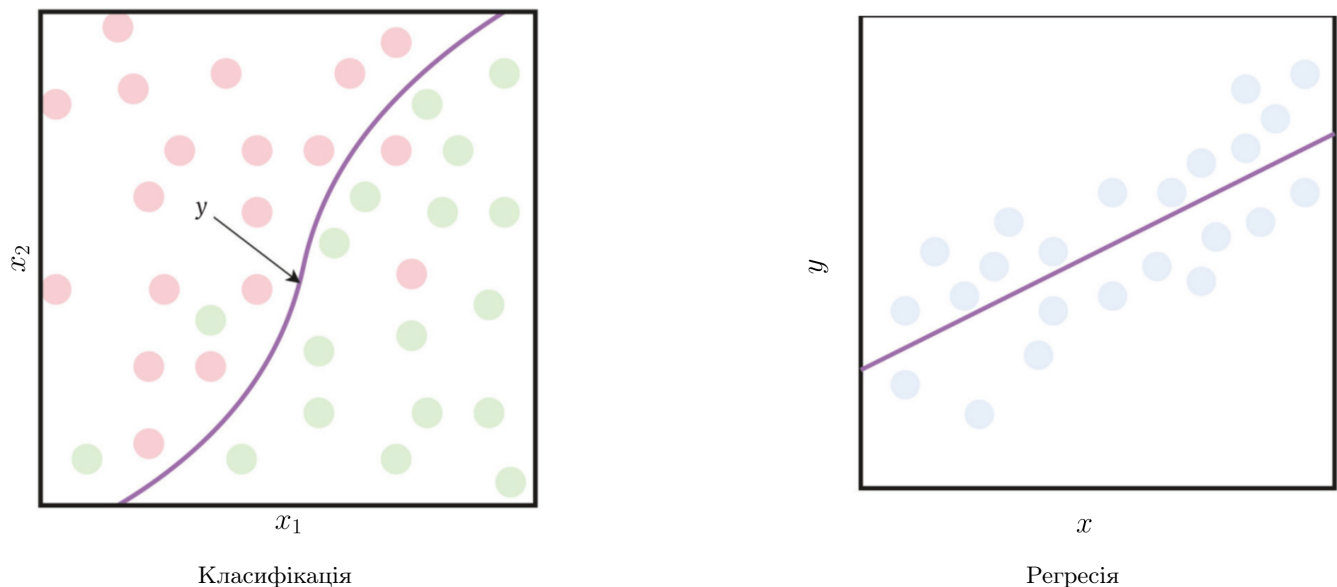


Рис. 1.3: Відмінність між класифікацією та регресією.

Мітка $y^{(i)}$ може бути або елементом кінцевої множини класів $\{1, 2, \dots, C\}$ або дійсним числом, або складнішою структурою, такою як вектор, матриця, дерево чи граф. У задачах класифікації, клас – категорія даних, до якої належить розглянутий приклад. Скажімо, якщо прикладами є повідомлення електронної пошти, а наше завдання полягає у виявленні спаму, тоді усі повідомлення потрібно розділити на два класи: спам, не спам.

Завдання передбачення класу називається класифікацією, а завдання передбачення дійсного числа – регресією. Числове значення мітки, яке має бути передбачене моделлю, навченою з учителем, називається цільовим показником або метою. Прикладом регресії є завдання передбачення заробітної плати співробітника на основі його досвіду роботи та знань. Прикладом класифікації є завдання, коли модель повертає висновок (здоровий, хворий) на основі введених лікарем показників, які характеризують стан пацієнта.

Відмінність між класифікацією та регресією показано на рис. 1.3. У разі класифікації алгоритм навчання шукає лінію або, у загальному випадку, гіперповерхню, яка розділяє приклади різних класів. З іншого боку, у разі регресії алгоритм навчання прагне знайти лінію чи гіперповерхню, яка добре відповідає навчальним прикладам.

Мета алгоритму навчання з учителем – знайти екстремум цільової функції² на навчальному наборі даних і скоротити розрив між прогнозами моделі та мітками реальних спостережень. Модель на вхід приймає вектор ознак, а на виході видає прогноз, який дозволяє визначити мітку для цього вектора ознак. Наприклад, модель, створена з використанням набору даних про пацієнтів, може на вхід приймати вектор ознак, що характеризує стан пацієнта, а на виході видавати прогноз у вигляді ймовірності про наявності у пацієнта ознак деякого патологічного стану.

Те, що знаходиться всередині методів машинного навчання, зокрема, глибоких нейронних мереж, може бути складним, але за своєю суттю це просто функції. Вони беруть деякі вхідні дані (вектор ознак \mathbf{X}) і генерують деякі вихідні дані $f(\mathbf{X})$ – прогноз. Графічно це можна відобразити так як показано на рис. 1.4. Іншими словами, ми можемо говорити, що модель “дивиться” на вектор вхідних ознак і на основі досвіду роботи з аналогічними прикладами, виводить прогноз.

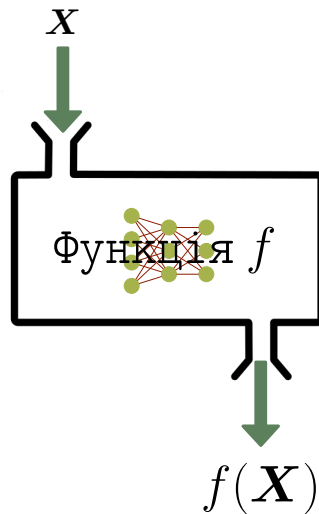


Рис. 1.4: Представлення моделі як функції.

1.1.2 Навчання без учителя

У цьому випадку ми працюємо з множиною нерозмічених прикладів³, яку представлено наступним чином:

$$d = \{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \dots, \mathbf{X}^{(n)}\}, \quad (1.2)$$

де n – загальна кількість прикладів у наборі.

Як і раніше, $\mathbf{X}^{(i)}$ – вектор вхідних ознак. Мета алгоритму навчання без учителя – створити модель, яка приймає на вхід вектор ознак \mathbf{X} і перетворює його на інший вектор або значення, яке можна використати для вирішення прикладного завдання. Наприклад, у задачах кластеризації, модель повертає ідентифікатор кластера для кожного вхідного вектора ознак з набору даних. Кластеризація використовується для пошуку груп схожих екземплярів у великому наборі даних, наприклад серед зображень або текстових документів.

У задачах зменшення розмірності модель повертає новий вектор ознак, який має меншу кількість елементів порівняно з вхідним вектором \mathbf{X} . Наприклад, дослідник має вектор ознак, який надто складний для його візуалізації та розуміння, оскільки число вимірів може бути дуже великим. Модель зменшення розмірності перетворює цей вектор великої розмірності на інший вектор меншої розмірності, який буде зберігати частину найбільш важливої інформації вхідного вектора і буде простішим для візуалізації та розуміння.

²Функція загальних втрат або емпіричний ризик.

³Відсутні мітки.

У задачах виявлення аномалій виходом є дійсне число, яке вказує, наскільки \mathbf{X} відрізняється від типового приклада з набору даних, який використовувався для навчання моделі. Виявлення аномалій може використовуватися для розпізнавання підозрілих або несанкціонованих дій у системах безпеки, таких як виявлення атак на комп'ютерні мережі або виявлення зловживань кредитними картками. Крім того, виявлення аномалій може бути корисним для визначення якості даних. Аналіз викидів може допомогти виявити та виправити помилки у даних, які можуть виникнути через неправильну розмітку, помилки вимірювань або інші фактори.

1.1.3 Навчання з частковим залученням учителя

У цьому випадку набір даних представлено множиною з розмічених та нерозмічених прикладів:

$$d = \{(\mathbf{X}^{(1)}, y^{(1)}), \mathbf{X}^{(2)}, \dots, \mathbf{X}^{(n)}\}, \quad (1.3)$$

де n – загальна кількість прикладів у наборі, $y^{(i)}$ – мітка i -го прикладу.

Як і у попередніх випадках, $\mathbf{X}^{(i)}$ – вектор вхідних ознак. Зазвичай кількість нерозмічених прикладів набагато більша ніж кількість розмічених. Навчання з частковим залученням учителя поєднує елементи навчання з учителем та навчання без учителя. Основна мета цього підходу полягає в тому, щоб використати невелику кількість прикладів з мітками для навчання моделі, а велику кількість нерозмічених прикладів – для отримання додаткової інформації та поліпшення універсальності моделі. У випадку, коли кількість прикладів для деяких класів обмежена, а для інших – значно більша, навчання з частковим залученням учителя може допомогти боротися з дисбалансом класів та покращити прогнози для екземплярів класів, які зустрічаються у наборі даних значно рідше за інші.

1.1.4 Навчання з підкріпленням

Навчання з підкріпленням – це розділ машинного навчання, де агент ⁴ вивчає оптимальну стратегію для своїх дій з метою максимізації загальної винагороди, отриманої внаслідок переходів між дозволеними станами середовища ⁵. Стан середовища, який доступний агенту для спостережень – вектор вхідних ознак. Оптимальна стратегія – це функція, яка на вхід приймає вектор ознак стану, а на виході видає оптимальну дію, яку слід виконати агенту у цьому стані. Дія є оптимальною, якщо вона максимізує середню загальну винагороду агента.

Базується навчання з підкріпленням на гіпотезі винагороди. Гіпотеза винагороди звучить наступним чином: будь-яка мета може бути формалізована як результат максимізації сукупної винагороди. Головна ідея гіпотези винагороди полягає у тому, що агент спрямований на вивчення оптимальної стратегії, тобто такої послідовностей дій, яка дозволить агенту отримати у кінцевому підсумку максимальну загальну винагороду.

Навчання з підкріпленням фундаментально відрізняється від контрольованого (з учителем) та неконтрольованого (без учителя) / напівконтрольованого навчання, оскільки ми тренуємо модель управляти діями нашого агента з метою знаходження ним оптимальної послідовності дій, які приведуть його до вирішення поставленого завдання з максимальним кінцевим значенням загальної винагороди. Іншими словами метою навчання агента є визначення найкращого наступного стану у який має перейти агент для отримання найбільшої кінцевої винагороди.

Задачі, де необхідно приймати рішення або запровадити певну поведінку (виконати певні дії), прийнято називати задачами керування. Навчання з підкріпленням займається вирішенням задач в яких прийняття рішень часто є послідовним, а мета – довгостроковою. Такі завдання виникають в іграх, робототехніці, управлінні ресурсами чи логістикою.

⁴Агент – сутність (програма чи об'єкт), що існує та діє окремо в деякому середовищі.

⁵Стохастичний та невизначений світ у якому існує та діє агент.

1.2 Мотивація та інтуїція

Для більшості задач машинного навчання вибір хороших ознак⁶ має першочергове значення. Навіть найкращий алгоритм не зможе продемонструвати хороших результатів, якщо для його навчання було використано погані ознаки. З іншого боку, вибір ознак часто є вкрай нетривіальним завданням. Наприклад, розглянемо кольорове зображення обличчя людини. Нехай у цьому випадку зображення обличчя людини буде представлено інтенсивністю пікселя для трьох кольорів: червоного, зеленого та синього (якщо використовується RGB-кодування). Тому, 1М піксельне кольорове зображення буде мати 3М ознак за якими ми можемо навчати нашу модель. З іншого боку, вираз обличчя людини, ймовірно, можна охарактеризувати ≤ 56 ознаками (на обличчі людини є ~ 56 м'язів). Отже, для ідентифікації конкретних виразів обличчя, дані великої розмірності можуть бути представлені відповідними ознаками меншої розмірності.

Таким чином, ідеальний екстрактор (видобувач) ознак буде приймати на вхід зображення обличчя, а на виході видавати видобуті ознаки, що характеризують вираз обличчя людини. Однак, на теперішній час, не існує ідеальних способів як це зробити. У традиційних методах розпізнавання образів, які були розроблені починаючи з 50-х років, ці екстрактори ознак були жорстко закодовані на основі суб'єктивної інтуїції дослідників. Основна ідея, яка прийшла до нас разом з нейронними мережами, полягає в тому, що хороші ознаки можуть бути вивчені мережею безпосередньо з даних, таким чином більше непотрібно досліднику видобувати їх вручну.

З іншого боку, базова модель слабо розвинулася з часу її створення починаючи з 1950-х років. Перша машина («Марк-1», Френк Розенблат в 1957 р.), яка реалізовувала алгоритм перцептрона (булева модель нейрона Мак-Калок-Пітса) була лінійним класифікатором, побудованим поверх простого жорстко прописаного екстрактора ознак. До сьогодення на практиці для вирішення прикладних задач машинного навчання використовують ручне видобування ознак.

Звичайно, якби нейронна мережа була представлена лише лінійними шарами (нейронами), сукупний ефект також був би лінійним і ми могли б згорнути всю архітектуру мережі лише в один шар (нейрон). Це пояснюється тим, що результат комбінації лінійних перетворень залишається лінійним перетворенням. З іншого боку, введення нелінійності відкриває можливість для побудови глибоких мереж з кількох шарів, оскільки їх неможливо лінійно об'єднати, таким чином кожен окремих нейрон та шар буде вивчати різні ознаки.

Загалом, у цьому підручнику ми будемо переважно розглядати методи контрольованого навчання на основі градієнта, де результат прогнозу для деякого вхідного вектора ознак $\mathbf{X}^{(i)}$ представлений функцією:

$$\hat{y}^{(i)} = f(\mathbf{W}, b, \mathbf{X}^{(i)}), \quad (1.4)$$

де \mathbf{W} – деякий вектор/матриця вагових коефіцієнтів (навчальні параметри), b – сзув моделі (скаляр/вектор, також навчальний параметр), f – деяка функція аргументів (може бути нелінійною). Якість моделі визначається на основі деякої цільової функції втрат J :

$$, \quad (1.5)$$

$$L(\mathbf{W}, b, \mathbf{X}^{(i)}, y^{(i)}) = L(f(\mathbf{W}, b, \mathbf{X}^{(i)}), y^{(i)}) = L(\hat{y}^{(i)}, y^{(i)}) \quad (1.6a)$$

$$J(\hat{y}, y) = \frac{1}{n} \sum_{i=1}^n L(\hat{y}^{(i)}, y^{(i)}), \quad (1.6b)$$

⁶Ознака (фіча) – це окрема властивість чи характеристика у даних, від якої безпосередньо залежить вихідний результат передбачення моделі.

де $L(\hat{y}^{(i)}, y^{(i)})$ – деяка міра невідповідності (функція втрат) між істинною величиною y та оціночною величиною \hat{y} для одного навчального прикладу, тут розглянуто деякий i -й приклад з набору даних. Наша мета – знайти таку конфігурацію навчальних параметрів моделі, які будуть відповідати глобальному мінімуму цільову функцію втрат ⁷ на валідаційній/тестовій вибірці даних. Після того як модель зробить передбачення, ми розраховуємо втрати і оновлюємо ваги та зсув моделі в найкращому напрямку, для цього обчислюємо градієнти цільової функції втрат відносно цих навчальних параметрів:

$$\frac{\partial J(\mathbf{W}, b, \mathbf{X}, y)}{\partial \mathbf{W}} = \frac{\partial J(f(\mathbf{W}, b, \mathbf{X}), y)}{\partial \hat{y}} \frac{\partial f(\mathbf{W}, b, \mathbf{X})}{\partial \mathbf{W}} \quad (1.7a)$$

$$\frac{\partial J(\mathbf{W}, b, \mathbf{X}, y)}{\partial b} = \frac{\partial J(f(\mathbf{W}, b, \mathbf{X}), y)}{\partial \hat{y}} \frac{\partial f(\mathbf{W}, b, \mathbf{X})}{\partial b} \quad (1.7b)$$

Градієнт показує нам напрям найкрутішого зростання цільової функції у заданій точці кривої. Так як нам потрібно мінімізувати втрати моделі, маємо взяти градієнт з від’ємним знаком – це буде напрям найкрутішого спадання нашої цільової функції. Далі виконуємо оновлення навчальних параметрів:

$$\mathbf{W} = \mathbf{W} - \alpha \frac{\partial J(\mathbf{W}, b, \mathbf{X}, y)}{\partial \mathbf{W}} \quad (1.8a)$$

$$b = b - \alpha \frac{\partial J(\mathbf{W}, b, \mathbf{X}, y)}{\partial b}, \quad (1.8b)$$

де α – крок навчання (швидкість навчання). Залежно від того, скільки навчальних прикладів використовується для цього оновлення, ми отримуємо різні алгоритми оптимізації: стохастичний, пакетний та міні-пакетний градієнтний спуск (до цього ми повернемося пізніше).

З поданого вище шаблону виникають три запитання:

1. Яку архітектуру, наприклад, нейронної мережі, ми повині використовувати для вивчення $f(\mathbf{W}, b, \mathbf{X})$?
2. Яку функцію втрат $L(\hat{y}, y)$ слід використовувати?
3. Чи варто використовувати простий градієнтний спуск (1.8) чи якийсь більш досконалий алгоритм оптимізації?

1.2.1 Інжиніринг ознак

Інжиніринг (конструювання) ознак є дуже важливим етапом для створення моделі. Він передбачає видобування та вибір ознак. Під час видобування ознак витягуються з даних усі ознаки, які характеризують поставлену задачу. Під час вибору – визначаються усі найбільш важливі ознаки з метою покращення продуктивності моделі.

На рисунку 1.5 розглянуто приклад класифікації зображень. Ручне вилучення ознак з даних вимагає глибоких знань як задачі, яка вирішується, так і предметної галузі. Крім того, цей спосіб є трудомістким. Ми можемо автоматизувати процес конструювання ознак за допомогою глибинного навчання!

⁷Усереднені втрати моделі для всієї вибірки.

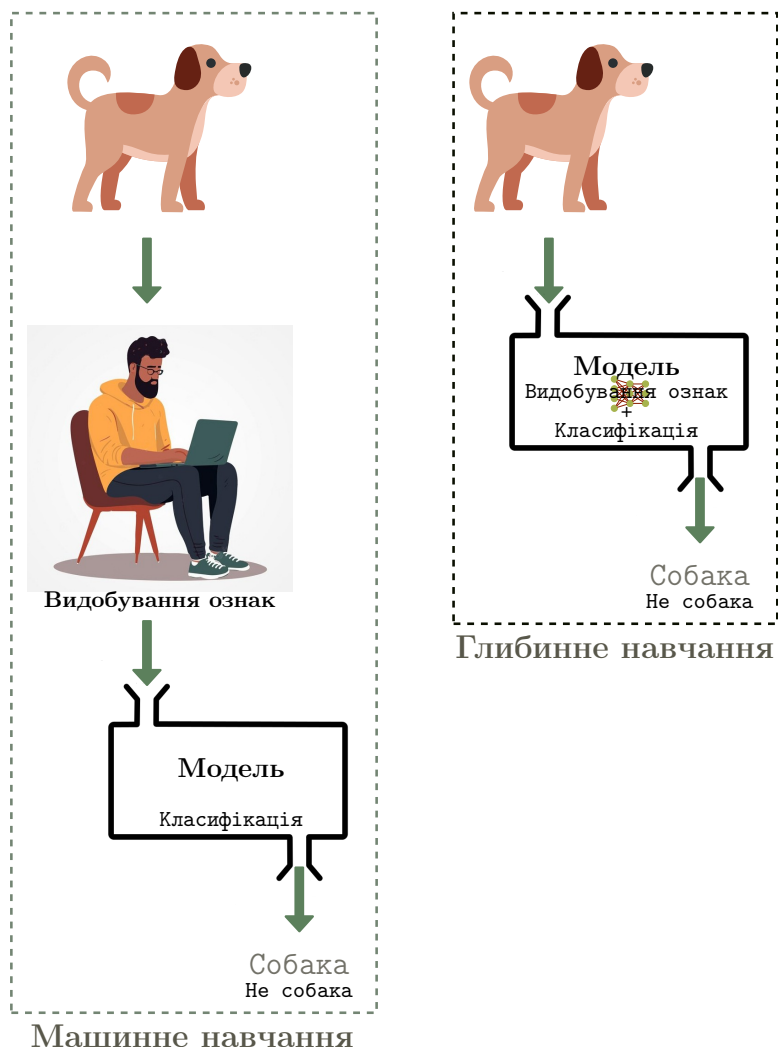


Рис. 1.5: Інжиніринг ознак в машинному навчанні та глибинному навчанні