ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Алгоритми класифікації

Зміст лекції

Що таке класифікація?

Класифікація - задача машинного навчання з учителем, де модель навчається відображати вхідні дані (ознаки) на дискретні класи (мітки). Мета - передбачити клас нових, раніше не бачених прикладів.

Вхідні дані (X)	Модель (f)	Вихід (y)
Числові ознаки	Навчання на прикладах	Бінарний клас (0/1)
Категоріальні змінні	Пошук закономірностей	Мультикласовий (A,B,C)
Текст, зображення	Побудова межі рішення	Ймовірності класів
Вектори ознак	Параметри моделі	Передбачена мітка

Приклади: спам/не спам • хвороба/здоровий • кішка/собака/птах • тональність тексту

k-Найближчих сусідів (kNN)

Принцип роботи

Клас нового об'єкта визначається голосуванням k найближчих сусідів у просторі ознак за метрикою відстані.

Параметри

k — кількість сусідів (зазвичай 1–20). Метрика: Евклідова, Манхеттенська, Мінковського.

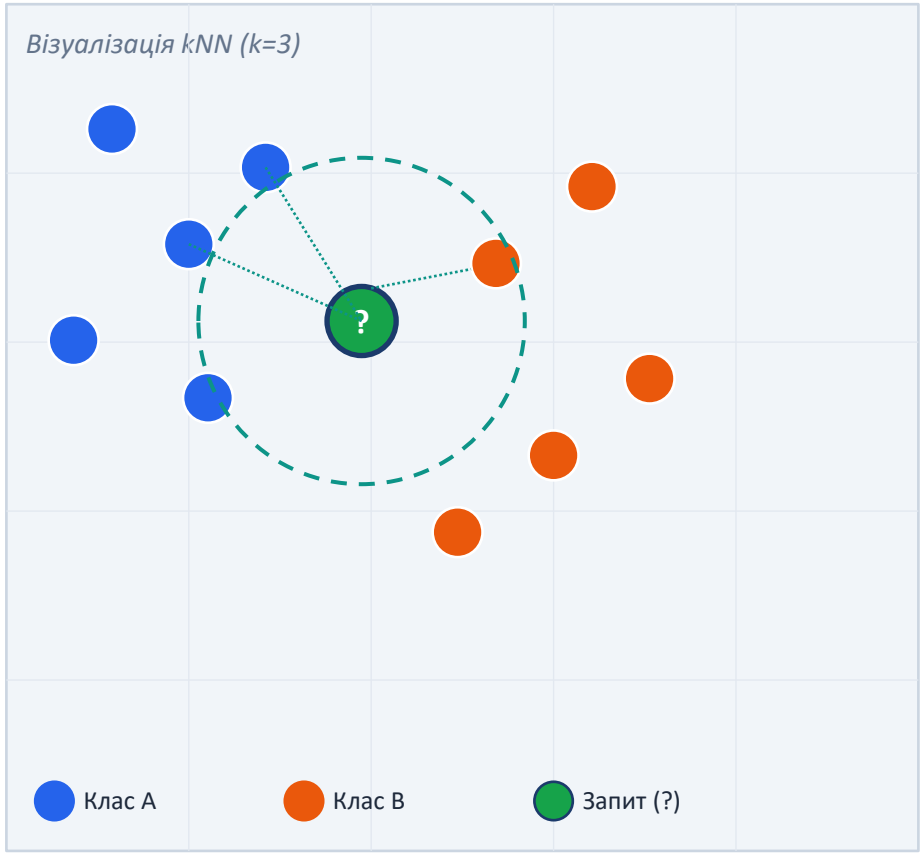
Переваги

Проста реалізація. Відсутність фази навчання. Добре для нелінійних меж рішення.

Недоліки

Повільний при великих даних. Чутливий до масштабу та шуму. Прокляття розмірності.

Візуалізація kNN (k=3)



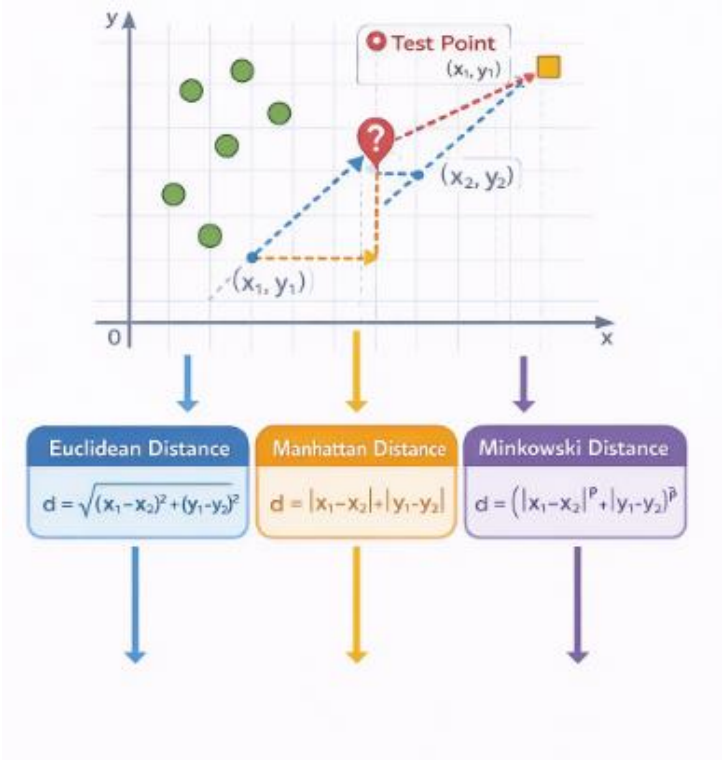
k-Найближчих сусідів (kNN)

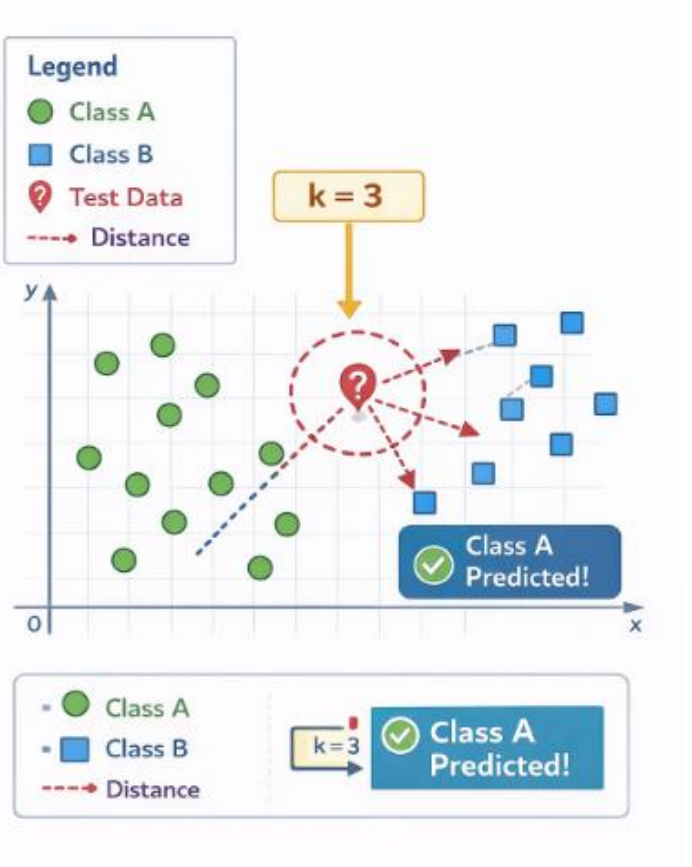
Як працює k-Nearest Neighbors?

1. Завантажте всі навчальні дані, а також дані тестів
2. Виберіть відповідне значення K, яке означає найближчі точки даних, і K може бути будь-яким цілим числом. Тепер для окремої точки даних виконайте наступне:
   - Виміряйте відстань між тестовими даними та кожним рядком навчальних даних, використовуючи такі методи подібності, як відстань (Евклідова відстань, Манхеттенська відстань, відстань Мінковського або метрика перекриття/відстань Хеммінга для дискретних змінних).
   - Відсортуйте точки даних у порядку зростання
   - Виберіть перші K рядків із відсортованого масиву
3. Призначте клас тестовим даним.

k-Найближчих сусідів (kNN)

Сині та зелені точки — це два класи даних навчання





k-Найближчих сусідів (kNN)

Переваги

Легко реалізувати. Враховуючи простоту і точність алгоритму.

Легко адаптується. Коли додаються нові навчальні зразки, алгоритм налаштовується з урахуванням будь-яких нових даних, оскільки всі навчальні дані зберігаються в пам’яті.

Кілька гіперпараметрів. KNN вимагає лише значення k і метрики відстані, що є низьким порівняно з іншими алгоритмами машинного навчання.

Недоліки

Погано масштабується. Оскільки KNN є ледачим алгоритмом, він займає більше пам’яті та зберігання даних порівняно з іншими класифікаторами.

Прокляття розмірності. Алгоритм KNN, як правило, стає жертвою прокляття розмірності, що означає, що він погано працює з вхідними даними великої розмірності.

Схильність до переобладнання. Через «прокляття розмірності» KNN також більш схильний до переобладнання. Хоча методи вибору функцій і зменшення розмірності використовуються для запобігання цього, значення k також може впливати на поведінку моделі.

Дерева рішень (Decision Tree)

Принцип

Ієрархічна структура тест-вузлів. Кожен вузол перевіряє ознаку, гілки — можливі значення, листи — класи.

Критерії розбиття

Індекс Gini: мінімізація домішок у вузлі.
Інформаційний приріст (Entropy / ID3).
Gain Ratio (C4.5).

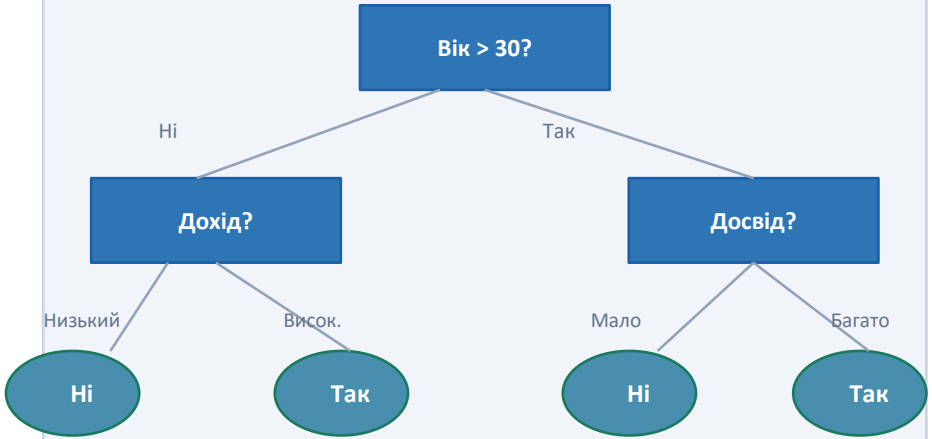
Переваги

Інтерпретованість. Не потребує масштабування. Працює з категоріальними ознаками.

Проблеми

Схильне до перенавчання. Нестабільність. Потрібне підрізання (pruning).

Приклад дерева рішень



$$Gini = 1 - (p1^2 + p2^2) \quad | \quad IG = H(S) - \Sigma|Sv|/|S| \cdot H(Sv)$$

Популярні бібліотеки: sklearn.tree.DecisionTreeClassifier
Параметри: max_depth, min_samples_split, criterion

Дерева рішень (Decision Tree)

Дерево рішень — це модель, яка приймає рішення шляхом послідовних перевірок умов.
Структура нагадує дерево:
корінь (root) — початкова перевірка
вузли (nodes) — умови
гілки (branches) — результати перевірок
листя (leaves) — кінцевий результат (клас)

Наївний класифікатор Байєса

Теорема Байєса:

$P(y \mid X) = (P(X \mid y) \cdot P(y)) / P(X)$ де $P(X \mid y) = \Pi P(xi \mid y)$ (наївне припущення незалежності)

$P(y \mid X)$ **(Апостеріорна ймовірність)** - ймовірність того, що об'єкт належить до класу за умови, що ми бачимо ознаки. Це те, що ми намагаємося передбачити (наприклад, "чи є імейл спамом?").

$P(X \mid y)$ **(Правдоподібність)** - ймовірність появи ознак у середині класу.

$P(y)$ **(Апріорна ймовірність)** - загальна ймовірність класу у наборі даних (як часто взагалі зустрічається спаму).

$P(X)$ - Загальна ймовірність появи ознак. При класифікації вона зазвичай ігнорується, бо вона однакова для всіх класів.

Переваги. Швидке навчання і передбачення. Добрий базовий класифікатор. Мало тренувальних даних. Легко інтерпретується.

Обмеження. Наївне припущення незалежності рідко виконується. Проблема нульових ймовірностей (лапласове згладжування).

Наївний класифікатор Байєса

Гауссівський
Gaussian NB

Неперервні ознаки. Апроксимація гауссовим розподілом P(xi|y). Підходить для кількісних даних.

Мультиноміальний
Multinomial NB

Для дискретних підрахунків (частоти слів). Популярний у класифікації тексту та NLP.

Бернуллі
Bernoulli NB

Бінарні ознаки (присутність/відсутність). Ефективний для бінарного bag-of-words.

Тип Байєса	Тип даних	Типовий приклад
Gaussian	Числові, неперервні	Прогноз погоди, фізичні виміри
Multinomial	Лічильники, частоти	Класифікація довгих текстів, статей
Bernoulli	Бінарні (0/1)	Короткі SMS, перевірка наявності ознак

Наївний класифікатор Байєса

Метод опорних векторів (SVM)

Ідея

SVM шукає гіперплощину з максимальним зазором (margin) між класами. Опорні вектори - точки, найближчі до межі.

Kernel Trick

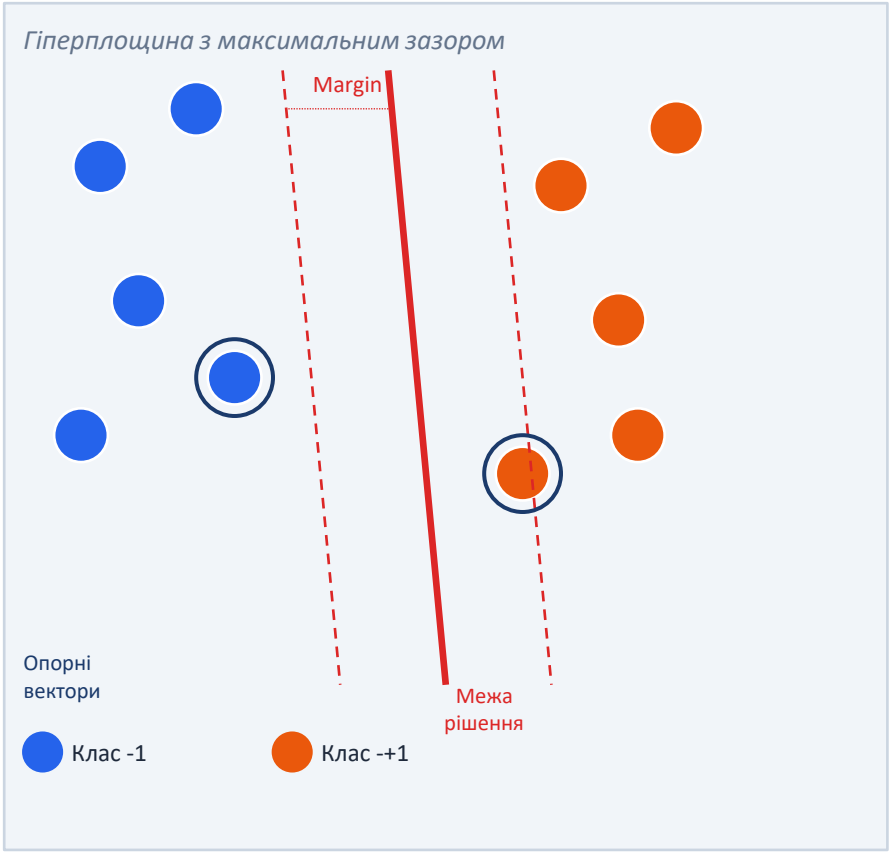
Нелінійні задачі: дані проектуються у вищий вимір. Ядра: RBF, Polynomial, Sigmoid, Linear.

Параметри

C - штраф за помилку класифікації. gamma - ширина ядра RBF, epsilon - для SVR.

Застосування

Текстова класифікація. Розпізнавання облич. Біоінформатика. Висока точність при малих даних.



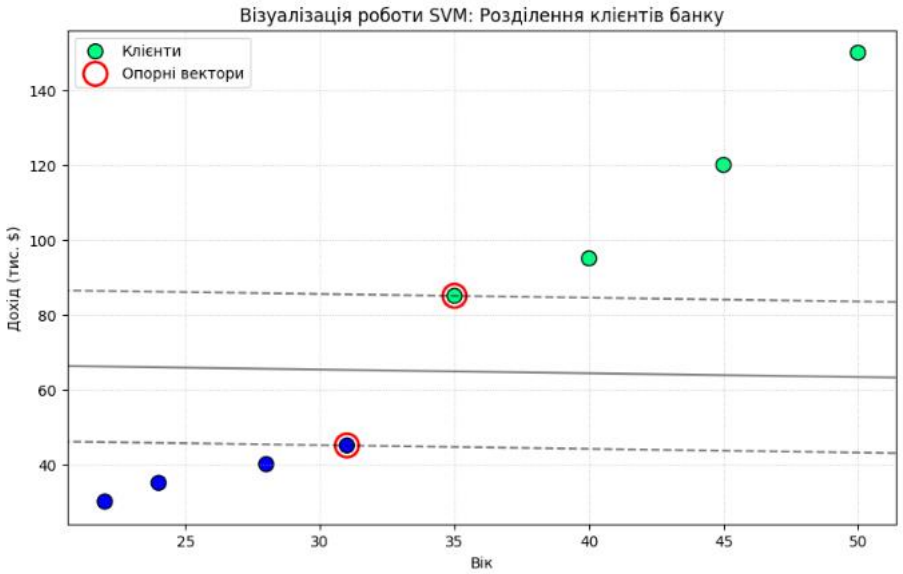
Метод опорних векторів (SVM)

Банк хоче автоматично розподіляти клієнтів на дві групи: заможні (Клас = +1) та звичайні (Клас = −1) на основі двох ознак — вік та річний дохід (тис. $). Побудуйте SVM-класифікатор та проаналізуйте результат.

Приклад

Для прикладу візьмемо навчальну вибірку:

```
data = {'Вік': [45, 22, 35, 28, 50, 24, 40, 31],
    'Дохід': [120, 30, 85, 40, 150, 35, 95, 45],
    'Досвід': [20, 1, 10, 3, 25, 2, 15, 5],
    'Освіта': [3, 2, 3, 1, 3, 2, 2, 1],
# 1 - середня, 2 - бакалавр, 3 - магістр
    'Клас': [1, -1, 1, -1, 1, -1, 1, -1]
# 1 - Premium, -1 - Standard
}
```



Логістична регресія

Логістична регресія — це статистичний метод прогнозування бінарних класів.

Це лінійна модель, яка застосовується для бінарної класифікації. Алгоритм використовується для прогнозування ймовірності настання певної події.

Мета логістичної регресії — знайти найкращі коефіцієнти (ваги), які мінімізують похибку між прогнозованою ймовірністю та спостережуваним результатом.

Це робиться за допомогою алгоритму оптимізації, наприклад градієнтного спуску, для коригування коефіцієнтів, доки модель якомога краще відповідатиме навчальним даним.

Логістична регресія

Sigmoid функція: $\sigma(z) = 1 / (1 + e^{(-z)})$ де $z = w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n$

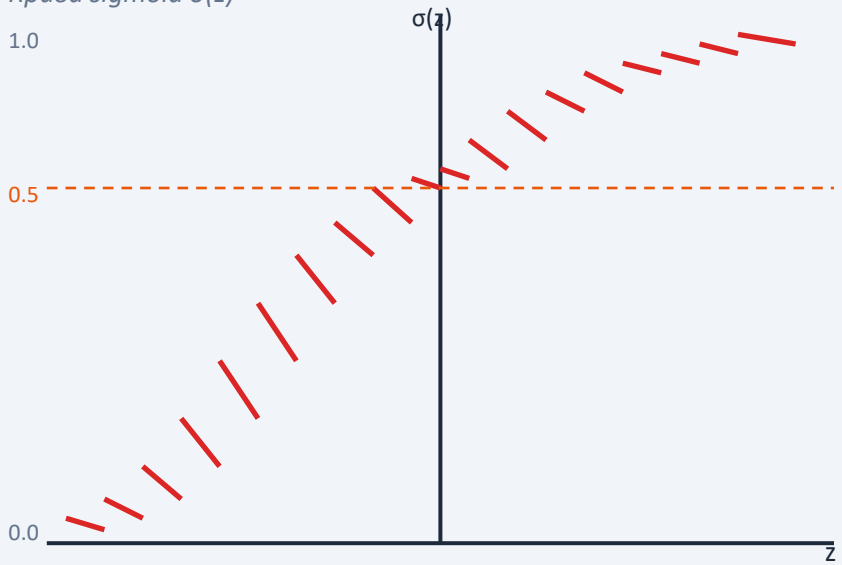
Ймовірнісний вихід
Повертає ймовірність $P(y=1|x) \in [0,1]$. Поріг 0.5 для бінарної класифікації.

Функція втрат
Binary Cross-Entropy: $L = -[y \cdot \log(p) + (1-y) \cdot \log(1-p)]$. Мінімізується градієнтним спуском.

Регуляризація
L1 (Lasso) — розріджує ваги. L2 (Ridge) — зменшує ваги. Параметр C в sklearn.

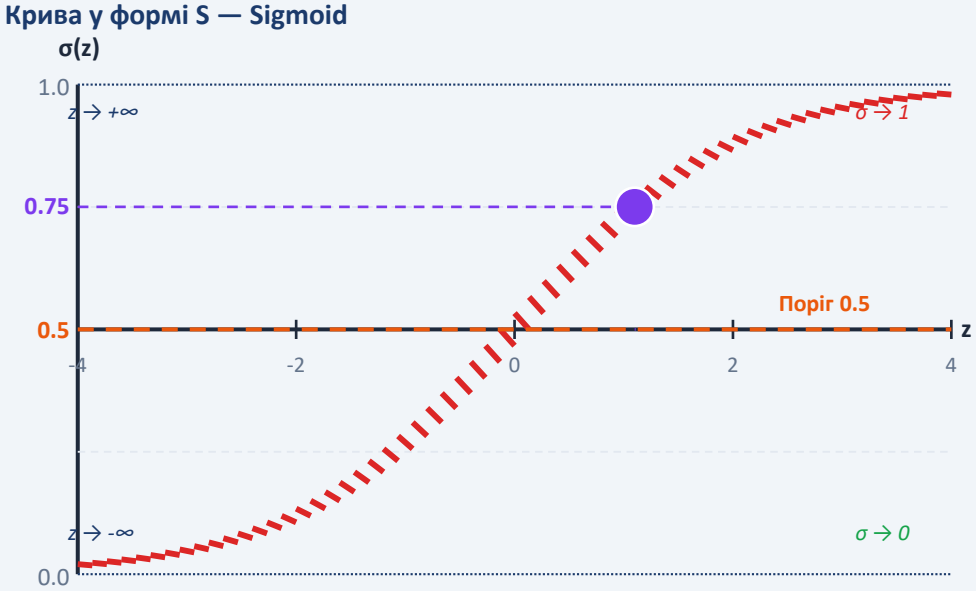
Multiclass
One-vs-Rest (OvR) або One-vs-One (OvO). Softmax — для мультикласової задачі.

Крива sigmoid σ(z)



Сигмоїдна (Логістична) функція

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Якщо σ(z) > 0.5 → Клас 1 (ТАК)

Крива прямує до +∞, прогнозований y = 1. Результат класифікується як позитивний.

Якщо σ(z) < 0.5 → Клас 0 (НІ)

Крива прямує до -∞, прогнозований y = 0. Результат класифікується як негативний.

Ймовірнісна інтерпретація

σ(z) = 0.75 означає 75% ймовірність позитивного класу. Приклад: 75% ймовірність того, що пацієнт хворіє на рак.

Діапазон виходу: (0, 1)

Функція ніколи не досягає рівно 0 або 1. Вихід — завжди дійсне число між 0 та 1 (виключно).

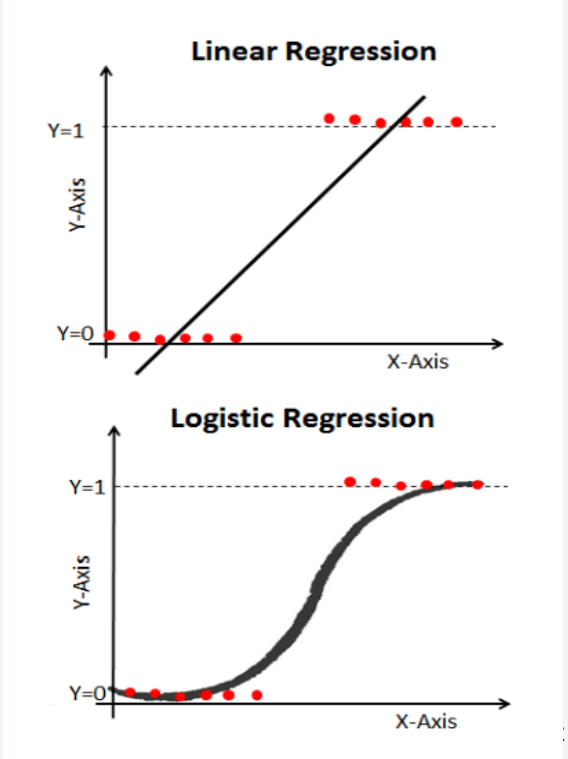
Логістична регресія

Властивості логістичної регресії

- Залежна змінна в логістичній регресії відповідає розподілу Бернуллі.
- Оцінка здійснюється через максимальну ймовірність.
- логістична регресія оцінюється за допомогою підходу максимальної правдоподібності (MLE) (Лінійна регресія оцінюється за допомогою методу звичайних найменших квадратів (OLS))

Типи логістичної регресії

- **Бінарна логістична регресія**: цільова змінна має лише два можливі результати.
- **Мультиноміальна логістична регресія**: цільова змінна має три або більше номінальних категорій.
- **Порядкова логістична регресія**: цільова змінна має три або більше порядкових категорій, наприклад рейтинг



Приклад

```
import numpy
from sklearn import linear_model

# X - розмір пухлини в сантиметрах
# y означає, чи є пухлина злоякісною (0 для «Ні», 1 для «Так»).
# Примітка!
# Щоб функція LogisticRegression() працювала, X потрібно змінити на стовпець із рядка.

X = numpy.array([3.78, 2.44, 2.09, 0.14, 1.72, 1.65, 4.92, 4.37, 4.96, 4.52, 3.69, 5.88]).reshape(-1,1)
y = numpy.array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1])

logr = linear_model.LogisticRegression()
logr.fit(X,y)

#передбачити, чи є пухлина раковою, якщо розмір становить  3.29 мм:
predicted = logr.predict(numpy.array([3.29]).reshape(-1,1))
print(predicted)
```

Приклад

```
log_odds = logr.coef_
odds = numpy.exp(log_odds)

print(odds)

def logit2prob(logr,x):
  log_odds = logr.coef_ * x + logr.intercept_
  odds = numpy.exp(log_odds)
  probability = odds / (1 + odds)
  return(probability)

print(logit2prob(logr, X))
```

```
[[0.60749955]
 [0.19268876]
 [0.12775886]
 [0.00955221]
 [0.08038616]
 [0.07345637]
 [0.88362743]
 [0.77901378]
 [0.88924409]
 [0.81293497]
 [0.57719129]
 [0.96664243]]
```

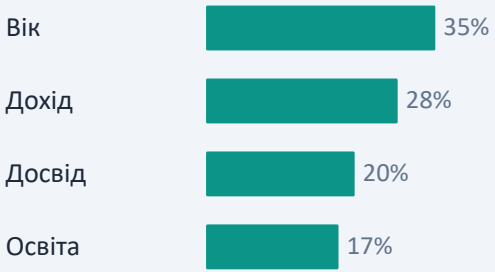
```
[[4.03541657]]
```

Випадковий ліс (Random Forest)

Ансамблевий метод, що будується з N незалежних дерев рішень на Bootstrap-вибірках. Фінальне рішення - голосування більшості (для класифікації) або середнє (для регресії).

1. Bootstrap вибірка
2. Випадковий підпростір ознак
3. Побудова дерева
4. Агрегація (Bagging)
5. Фінальне передбачення

Feature Importance



Переваги

Стійкість до перенавчання. Важливість ознак. Робота з пропусками. Паралелізується.

Обмеження

Повільніший за одне дерево. Модель-чорний ящик. Великий обсяг пам'яті.

Випадковий ліс (Random Forest)

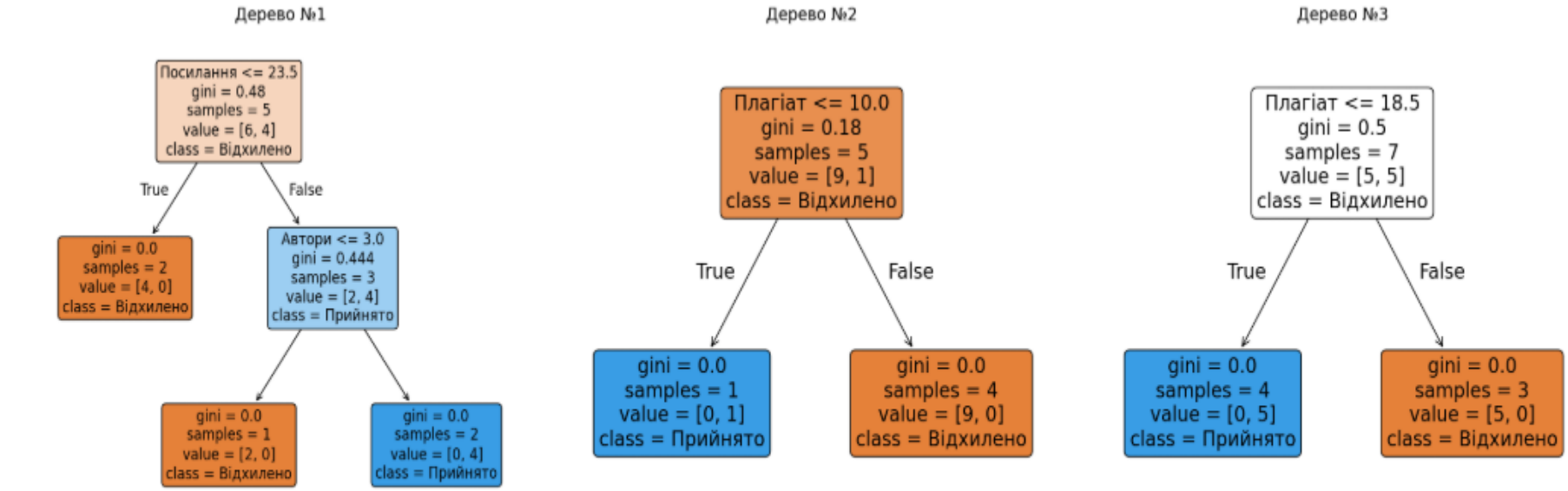
Приклад. Чи приймуть наукову статтю до журналу?

Уявімо, що ми хочемо передбачити долю рукопису на основі чотирьох ознак:
Кількість авторів (від 1 до 10).
Кількість посилань у списку літератури (наскільки глибокий аналіз).
Рівень плагіату (той самий відсоток, про який ми згадували — 94% чи 5%).
Наявність математичних формул (0 — немає, 1 — є).

Як працює «Ліс»?
Алгоритм створює, наприклад, **3 незалежні дерева рішень**. Кожне дерево бачить лише частину даних і частину ознак (це називається *Feature Randomness*).
Дерево №1. Дивиться лише на плагіат. Якщо він > 20% - відразу «Відхилити».
Дерево №2. Дивиться на кількість посилань. Якщо їх < 10 - «Відхилити», якщо > 10 і є формули - «Прийняти».
Дерево №3. Дивиться на кількість авторів. Якщо їх 5+ (колективна праця), можливо, стаття надійна - «Прийняти».

Випадковий ліс (Random Forest)

Приклад. Чи приймуть наукову статтю до журналу?




```
Важливість ознаки 'Автори': 18.52%
Важливість ознаки 'Посилання': 14.81%
Важливість ознаки 'Плагіат': 66.67%
Важливість ознаки 'Формули': 0.00%
```

Нейронні мережі для класифікації

Архітектура MLP

Вхідний шар → Приховані шари (Dense) → Вихідний шар (Softmax/Sigmoid). Backpropagation для навчання.

Функції активації

ReLU: max(0,x) — приховані шари. Sigmoid — бінарний вихід. Softmax — мультиклас.

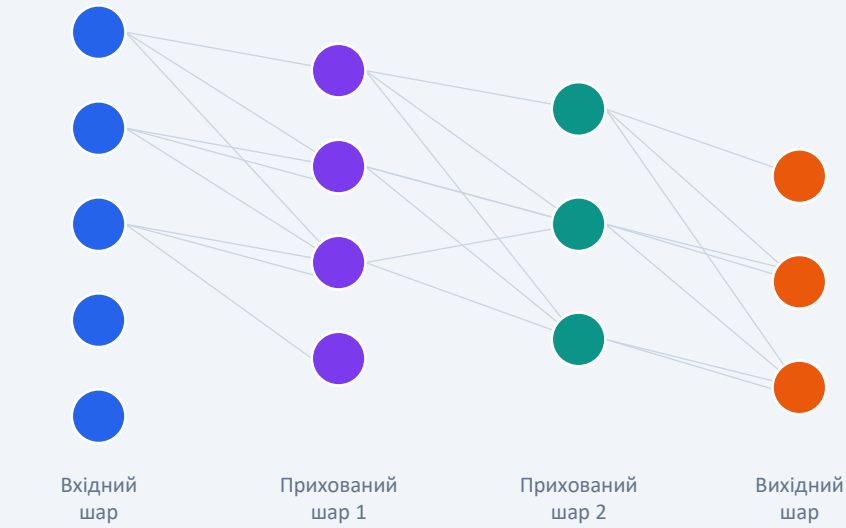
Функції втрат

Binary cross-entropy: бінарна класифікація. Categorical cross-entropy: мультикласова задача.

Регуляризація

Dropout — випадкове відключення нейронів. Batch Normalization. L1/L2 weight decay.

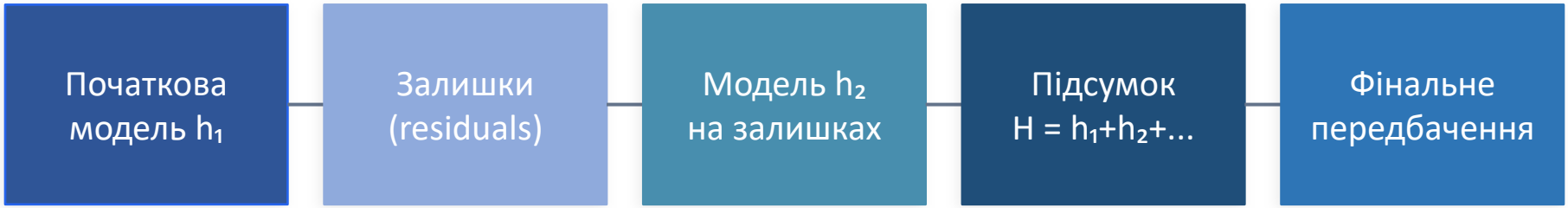
Багатошаровий перцептрон (MLP)



Keras: model.add(Dense(64, activation='relu')) → model.add(Dense(3, activation='softmax'))

Градієнтний бустинг (XGBoost, LightGBM)

Бустинг — послідовне навчання слабких класифікаторів. Кожна нова модель виправляє помилки попередньої, мінімізуючи функцію втрат за градієнтним спуском у просторі функцій.



XGBoost

- Level-wise побудова дерев
- Регуляризація L1 та L2
- Вбудована обробка пропусків
- Паралельна побудова дерев

LightGBM

- Leaf-wise (швидше та точніше)
- Гістограмне квантування ознак
- Підтримка категоріальних ознак
- Ефективний при великих даних

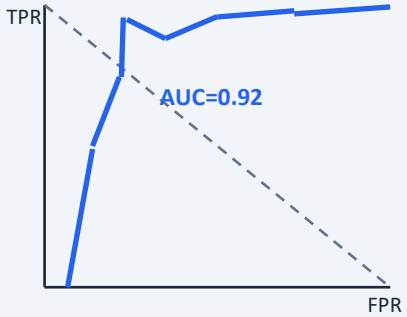
Оцінка якості класифікатора

Матриця невідповідності

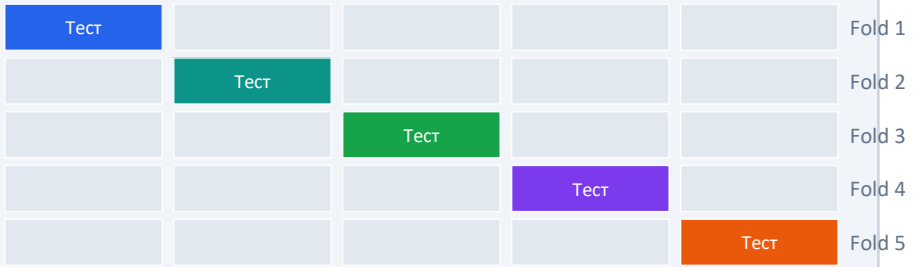
		Передбачено: Позитив	Передбачено: Негатив
Факт Поз.		**TP** True Positive	**FN** False Negative
Факт Нег.		**FP** False Positive	**TN** True Negative

Accuracy:	(TP+TN) / (TP+TN+FP+FN)
Precision:	TP / (TP+FP)
Recall:	TP / (TP+FN)
F1-Score:	2·(Prec·Rec) / (Prec+Rec)

ROC-крива та AUC



K-Fold крос-валідація (K=5)



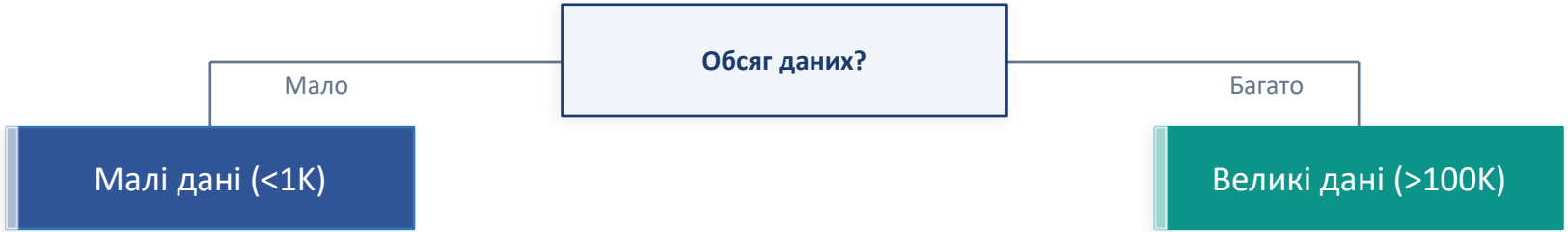
Результат: середнє ± std по K фолдам

Порівняльний аналіз алгоритмів

Алгоритм	Тип	Навчання	Передб.	Інтерпрет.	Перенавчання	Масштаб
kNN	Ліниве	O(1)	O(n·d)	Середня	Низьке	Погано
Дерево рішень	Жадібне	O(n·d)	O(log n)	Висока	Високе	Добре
Наївний Байєс	Ймовірнісне	O(n·d)	O(d)	Висока	Низьке	Відмінно
SVM	Ядрове	$O(n^2)$	O(sv·d)	Низька	Низьке	Погано
Логіст. регресія	Лінійне	O(n·d)	O(d)	Висока	Низьке	Відмінно
Random Forest	Ансамбль	O(n·d·T)	O(T·log)	Середня	Низьке	Добре
XGBoost	Бустинг	O(n·d·T)	O(T·log)	Низька	Низьке*	Добре
Нейр. мережа	Глибоке	O(n·d·it)	O(d·L)	Дуже низька	Середнє	Відмінно

** — з правильно підібраними гіперпараметрами (early stopping, regularization)*

Як обрати алгоритм класифікації?



Інтерпрет.? | **Точність?** | **Табличні?** | **Зобр./текст?** | **Лінійні?** | **Велика розм.**

Байєс / Лог. Рег. | SVM / XGBoost | Random Forest | Нейронна мережа | Логіст. Рег. | LightGBM / XGB

Практична порада: починайте з базових моделей (Logistic Regression, NB), потім ускладнюйте. Завжди порівнюйте з baseline. Оцінюйте за cross-validation.

Підсумки

Supervised Learning

Класифікація — навчання з вчителем. Модель вчиться на розмічених даних передбачати категорії.

Різноманіття методів

kNN, Дерева рішень, Байєс, SVM, Логістична регресія, Random Forest, XGBoost, Нейронні мережі.

Оцінка якості

Accuracy, Precision, Recall, F1. ROC-AUC. K-Fold крос-валідація для надійної оцінки.

Вибір алгоритму

Залежить від обсягу даних, потреби в інтерпретованості, типу ознак і обчислювальних ресурсів.

No Free Lunch

Жоден алгоритм не є кращим у всіх задачах. Завжди порівнюйте кілька підходів.

Практика

scikit-learn, XGBoost, LightGBM, TensorFlow, PyTorch — основні інструменти ML-інженера.