УДК 004.932
DOI: 10.31891/2219-9365-2021-68-2-11

БЕДРАТЮК Г.І.
Хмельницький національний університет

ЗАСТОСУВАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ЗНАХОДЖЕННЯ МАКСИМАЛЬНОГО ЕЛЕМЕНТУ

В роботі запропоновано реалізації технологіями класичного машинного навчання та аналіз складової частини алгоритму сортування, а саме, знаходження максимального елементу масиву. Було реалізовано такі методи – лінійну регресію, дерева рішень, метод опорних векторів, метод k-найближчих сусідів. Проведено порівняльний аналіз точності роботи за кожним алгоритмом.

Ключові слова: алгоритм сортування, максимальний елемент, машинне навчання, регресія, наївний баєсівський класифікатор, метод опорних векторів, дерево рішень

BEDRATYUK A.
Khmelnytsky National University

APPLICATION OF MACHINE LEARNING METHODS FOR FINDING THE MAXIMUM ELEMENT

Sorting is the core operation of many computational tasks, including design, digital signal processing, networking, database management, and data processing, for which it is estimated that sorting accounts for more than 25% of total runtime. Although there are well-known fast classical sorting algorithms, there are still interesting sorting algorithms that simulate the work of the human brain, in particular on the basis of neural networks and in general, on the basis of machine learning methods. Systems based on neural networks are probabilistic, ie algorithms based on them will always have a certain percentage of errors and, therefore, they are not used in classical programming.

In this paper, classical algorithms are used to construct machine learning models with a teacher that find the maximum element of the array, namely: linear regression, naive Bayesian classifier, reference vectors method, decision tree. The task of finding the maximum element was reduced to the problem of classification in this way - to obtain a marked data set generated random arrays, each of which was labeled a class equal to the position number of the maximum element of the array. Thus, all arrays of dimension n are divided into n classes - the class with number i includes all arrays in which the maximum element is in the i-th place. Data sets of different dimensions were formed - 2, 3, 4, 10, 20 and different quantities - 100,500,1000,2000, 5000,10000, 100000 arrays. Models were trained on each of the 7 datasets and the accuracy of training was found. Based on computational experiments, it was found that the maximum quality of all models - more than 99.9% - was achieved on an array of two elements and deteriorated with increasing dimension of the array, up to 96%. This quality is not inferior to the quality achieved by neural networks. Also, it was found that the best quality of learning among all classical machine learning methods tested is achieved for linear regression and decision tree.

The paper proposes the implementation of classical machine learning technologies and analysis of the component of the sorting algorithm, namely, finding the maximum element of the array. . The following methods were implemented: linear regression, decision trees, support-vector machines, the method of k-nearest neighbors. A comparative analysis of the accuracy of each algorithm.

Keywords: sorting algorithm, maximum element, machine learning, regression, naive Bayesian classifier, support-vector machines, decision tree

Постановка проблеми у загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями

Сортування є основною операцією багатьох обчислювальних завдань, включаючи проектування, цифрову обробку сигналів, мережевий зв'язок, управління базами даних та обробка даних, для яких вона підраховано, що на операції сортування припадає понад 25% загального часу роботи. Хоча існують добре відомі швидкі класичні алгоритми сортування, тим не менше викликають інтерес алгоритми сортування, які моделюють роботу людського мозку, зокрема на основі нейронних мереж і взагалі, на основі методів машинного навчання. Системи, які базуються на нейронних мережах мають ймовірнісний характер, тобто алгоритми, побудовані на їх основі, будуть завжди мати деякий відсоток помилок і, з цієї причини, вони не застосовуються в класичному програмуванні. В багатьох роботах пропонуються різні архітектури нейронних мереж для сортування масивів, див. [1]-[4].

Найбільш популярним підходом до побудови алгоритмів сортування, які використовують операцію порівняння елементів, тобто мають обчислювальну складність $O(n\ln(n)$ див. [5], є підхід, при якому знаходиться максимальний елемент, який вилучається з масиву, а потім процедура повторюється для масиву меншої розмірності. Для таких алгоритмів, наприклад для алгоритму пірамідального сортування, найважливішою компонентою є процедура знаходження максимального елементу. Вперше архітектура нейронної мережі для знаходження максимального елементу масиву була запропонована в [6].

В даній роботі, для побудови моделей машинного навчання з учителем, які знаходять максимальний елемент масиву, застосовуються класичні алгоритми, а саме: лінійна регресія, наївний баєсівський

класифікатор, метод опорних векторів, дерево рішень. Задача знаходження максимального елементу зводилася до задачі класифікації таким способом – для отримання розміченого набору даних генерувався випадкові масиви, кожному з яких ставилася мітка класу, яка рівна номеру позиції максимального елементу масиву. Таким чином всі масиви розмірності n розбиваються на n класів – до класу з номери i відносяться всі масиви у який максимальний елемент знаходиться на i-тому місці. Формувалися набори даних із різної розмірності – 2, 3, 4, 10, 20 та різної кількості – 100,500,1000,2000, 5000,10000 , 100000 масивів. Моделі навчалися на кожному з 7 датасетів і знаходилася точність навчання. На основі проведених обчислювальних експериментів встановлено що, максимальна якість роботи всіх моделей – понад 99.9% – досягалася на масиві із двох елементів і погіршувалася із збільшенням розмірності масиву, до 96%. Така якість не поступається якості, яка досягається нейронними мережамию. Також, встановлено, що найкращу якість навчання серед усіх класичних методів машинного навчання, які тестувалися, досягається для лінійної регресії та для дерева рішень.

Для виконання обчислень і та їх візуалізації використовувалися бібліотеки Python numpy, pandas, sklearn, mathplotlib.

Виклад основного матеріалу

2. Алгоритми машинного навчання. Дамо короткий огляд алгоритмів машинного навчання з учителем, які будуть використані в статті. Взагалі кажучи, машинне навчання це частина штучного інтелекту, яка розробляє і вивчає технології розробки алгоритмів, що навчаються на великій кількості готових розв'язків задач. В процесі навчання вони отримують досвід і їхня ефективність збільшується. Том Мітчел [7] дав таке означення поняття навчання програми – комп'ютерна програма навчається на основі досвіду $\boldsymbol{E}$ по відношенню до деякого класу задач $\boldsymbol{T}$ і міри якості $\boldsymbol{P}$, якщо якість вирішення завдань з $\boldsymbol{T}$, які виміряні на основі $\boldsymbol{P}$, покращується з набуттям досвіду $\boldsymbol{E}$ Навчання з учителем використовує помічені дані, тобто дані для яких уже відомий правильний результат. Якщо мітки заповнюють інтервал, тобто є неперервними, то така задача машинного навчання називається регресією. Якщо ж мітки належать до дискретної множини, то задача називається задачею класифікації. Ми будемо розглядати задачі класифікації, яка формалізується наступним чином. Нехай задана множина об'єктів $\boldsymbol{X}$, множина допустимих відповідей $\boldsymbol{Y}$, і існує цільова функція $\boldsymbol{y}^*: \boldsymbol{X} \to \boldsymbol{Y}$, значення якої $\boldsymbol{y_i} = \boldsymbol{y}^*(\boldsymbol{x_i})$ відомі тільки на скінченній підмножині із $\boldsymbol{l}$ об'єктів $\{\boldsymbol{x_1}, \dots, \boldsymbol{x_l}\} \subset \boldsymbol{X}$. Пари об'єкт-відповідь $(\boldsymbol{x_i}, \boldsymbol{y_i})$ називаються прецедентами. Сукупність $\boldsymbol{l}$ пар $\boldsymbol{X^l} = \{(\boldsymbol{x_i}, \boldsymbol{y_i})\}_{\boldsymbol{i=0}}^{\boldsymbol{l}}$ називається навчальною вибіркою. Завдання машинного навчання полягає в тому, щоб за вибіркою $\boldsymbol{X^l}$ відновити залежність $\boldsymbol{y}^*$, тобто побудувати вирішальну функцію $\boldsymbol{a}: \boldsymbol{X} \to \boldsymbol{Y}$, яка наближала (апроксимувала) б цільову функцію $\boldsymbol{y}^*(\boldsymbol{x})$, причому не тільки на об'єктах навчальної вибірки, а й на всій множині $\boldsymbol{X}$. Вирішальна функція $\boldsymbol{a}$ повинна допускати ефективну комп'ютерну реалізацію; з цієї причини будемо називати її алгоритмом. Адгоритми вибираються із деякої параметричної сім'ї моделей алгоритмів

$$A = \{g(x, \theta) | \theta \in \Theta\},$$

де $g: X \times \Theta \to Y$ – деяка фіксована функція, Θ - множина допустимих значень параметра θ, яке називається простором параметрів або простором пошуку(search space). Тут θ – шуканий вектор параметрів нашої моделі, X – набір ознак, а Y – множина допустимих відповідей. Задача знаходження невідомих параметри моделі зводиться до деякої оптимізаційної задачі.

Для знаходження максимального елемента ми використаємо такі алгоритми, які реалізовано в Python-бібліотеці sklearn.

- Лінійна регресія,
- метод опорних векторів (Support Vector Machine - SVM),
- дерева прийняття рішень (decision tree),
- метод k найближчих сусідів (k-nearest neighbors).

Дамо короткий опис цих алгоритмів

2.1. Лінійна регресія. Лінійна регресія є найпростішим алгоритмом машинного навчання який для задачі прогнозування навчає лінійну модель, яка є простою лінійною комбінацією вхідних даних (ознак). Лінійною моделлю для задачі відновлення регресії $\boldsymbol{Y} = \mathbb{R}$ із $\boldsymbol{n}$ числовими ознаками $\boldsymbol{f_j}: \boldsymbol{X} \to \mathbb{R}$ та вектором параметрів $\boldsymbol{\theta} \in \boldsymbol{\Theta} = \mathbb{R}^{\boldsymbol{n}}$ називається функція

$$g(\theta, x) = \sum_{j=1}^{n} \theta_j f_j(x), x = (x_1, x_2, \dots, x_n) \in X.$$

Для задачі бінарної класифікації $Y = \{-1,1\}$

$$g(\theta, x) = sign \sum_{j=1}^{n} \theta_j f_j(x).$$

Найчастіше розглядається випадок

$$g(\theta, x) = \sum_{j=1}^{n} \theta_j x_1, x = (x_1, x_2, \dots, x_n) \in X.$$

Тоді для знаходження параметрів мінімізується така функція помилки

$$\sum_{i=0}^{l} (y_i - g(\theta, x_i))^2.$$

Розв'язок цієї оптимізаційної задачі знаходиться методом градієнтного спуску.

Геометрично, лінійна модель бінарної класифікації намагається побудувати розділюючу гіперплощину в n-вимірному просторі, так, щоб об'єкти різних класів знаходилися в різних півплощинах простору.

2.2. Дерево рішень. Дерево рішень (decision trees) це алгоритм машинного навчання для задач класифікації та регресії, який зображується у вигляді повного дерева, як правило бінарного. Кожен листок (термінальний вузол) дерева помічений міткою класу, а в кожному вузлі знаходиться деякий предикат, аргументом якого є об'єкт призначений для класифікації. Процес класифікації відбувається шляхом руху об'єкта деревом від кореневого вузла до листка, мітка якого і присвоюється об'єкту, чим і завершується класифікація. Рух об'єкта регулюється предикатами, які в кожному нелистковому вузлі визначають на основі перевірки ознак об'єкта, в який саме із дочірніх вузлів він переміститься. Саме дерево, та форма предикатів в кожному його вузлі, конструюється у процесі навчання. Всі алгоритми навчання використовують ідею зменшення інформаційної ентропії на підвибірках. Алгоритм обирає найкращу ознаку на кожному кроці під час побудови дерева рішень. Для вибору найкращої ознаки, використовує коефіцієнт приросту інформації або індекс Джині.

Коефіцієнт приросту інформації обчислює зменшення ентропії та вимірює, наскільки дана ознака відокремлює або класифікує цільові класи. Функція, що має найбільший приріст інформації і є найкращою. Простими словами, ентропія є мірою невпорядкованості системи, а ентропія набору даних - мірою невпорядкованості набору даних. Нехай множина S складається із p_1 елемента класу 1 і p_2 елементів класу 2. Тоді ентропія множини S рівна

$$H(S) = -\frac{p_1}{p_1 + p_2} \log_2 \frac{p_1}{p_1 + p_2} - \frac{p_2}{p_1 + p_2} \log_2 \frac{p_2}{p_1 + p_2}.$$

Якщо всі елементи S належать до одного класу, тобто невизначеність відсутня, то тоді, очевидно, що тоді ентропія рівна 0. Якщо ж, елементів обох класів однакова кількість, то ентропія максимальна і рівна 1.

Наприклад , для множини S яка складається із 14 елементів, з яких 9 належать до першого класу і 5 елементів належать до другого класу, ентропія рівна

$$H(S) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.94$$

Якщо ж множина складається із елементів k різних класів, кількості p_i кожен, то її ентропія обчислюється за формулою

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

Вираз $\frac{p_i}{|S|}$ легко інтерпретується як ймовірність того, що випадково вибраний елемент належить до i-го класу.

Тепер ми можемо визначити міру ефективності ознаки при класифікації навчальних даних. Міра, яку ми будемо використовувати, називається інформаційним приростом, це просто очікуване зменшення ентропії, спричинене розділенням елементів вибірки за цим атрибутом. Нехай ознака X приймає деяку дискретну множину значень $V(X)$. Нехай $S_v(X)$ множина елементів вибірки, для яких ця ознака рівна v, тобто

$$S_v(X) = \{s \in S | X(s) = v \in V(X)\}.$$

Множини $S_v(X)$ задають розбиття множини S за ознакою X.
Інформаційним приростом ознаки X називається число

$$IG(S|X) = H(S) - \sum_{v \in V(X)} \frac{|S_v(X)|}{|S|} H(S_v(X))$$

Перший вираз є ентропія множини, а сума називається очікуваною ентропією ознаки X, яка є зваженою сумою ентропій розбиття за ознакою X.

2.3. Метод опорних векторів. Метод опорних векторів (Support Vector Machipe - SVM) - це потужна та універсальна модель машинного навчання, здатна виконувати лінійну чи нелінійну класифікацію, регресію та виявлення викидів. Вона є однією з найпопулярніших моделей машинного навчання. Методи SVM особливо добре підходять для класифікації складних, але невеликих чи середніх наборів даних. Інтуїтивно, хороший поділ досягається гіперплощиною, яка має найбільшу відстань до найближчих навчальних точок будь-якого класу оскільки чим більша ширина проміжку між класами, тим менша помилка узагальнення класифікатора. На Рис.1 показано розв'язок для задачі з лінійним розділенням із трьома точками на межах класів, які називаються «опорними векторами».

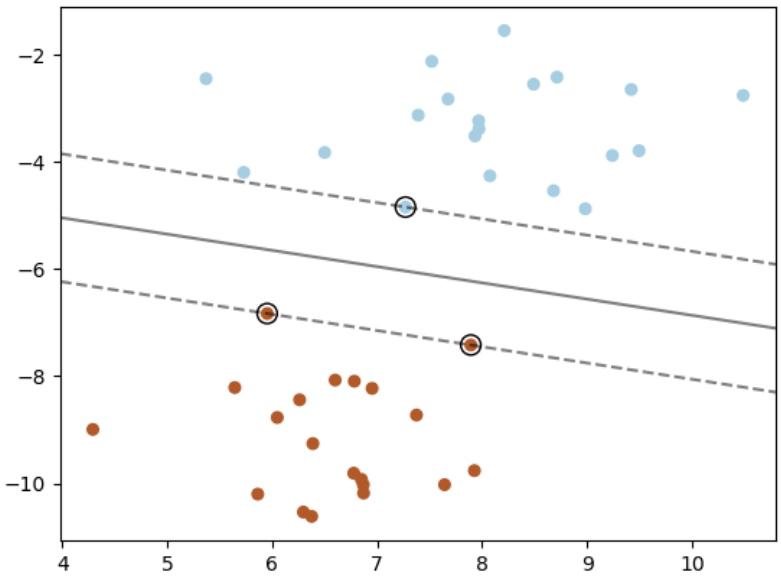

Рис 1. Опорні вектори

Вибір параметрів прямої лінії, яка максимізує проміжок між класами можна формалізувати до задачі оптимізації. Загальне рівняння прямої лінії $Ax_1 + Bx_2 + b = 0$ запишемо у векторному вигляді $\vec{w} \cdot \vec{x} + b = 0$, де $\vec{w} = (A, B)$, $\vec{x} = (x_1, x_2)$. Тут $\vec{w} \cdot \vec{x}$ – скалярний добуток векторів. Відстань від точки $\vec{x}_0$ до прямої $\vec{w} \cdot \vec{x} + b = 0$ знаходиться за формулою

$$\frac{|\vec{w} \cdot \vec{x}_0 + b|}{||\vec{w}||},$$

де

$$||\vec{w}|| = \sqrt{A^2 + B^2}.$$

Відстань між середньою прямою і прямою яка проходить через опорні вектори дорівнює

$$\frac{|\vec{w} \cdot \vec{x} + b|}{||\vec{w}||} = \frac{1}{||w||}$$

Відстань між опорними прямими дорівнює

$$\frac{2}{||w||}.$$

Для того щоб максимізувати відступ нам потрібно мінімізувати $||w||$ при умові що немає ніяких точок між опорними прямими.

$$\vec{x}_i \cdot \vec{w} + b \geq +1 \quad \text{коли} \quad y_i = +1$$
$$\vec{x}_i \cdot \vec{w} + b \leq -1 \quad \text{коли} \quad y_i = -1$$

Ці дві умови можуть бути об'єднані в одну

$$y_i(\vec{x}_i \cdot \vec{w} + b) \geq +1$$

Ми прийшли до задачі оптимізації квадратичної функції

$$\min \frac{1}{2}||\vec{w}||^2$$

при обмеженнях ($\vec{x}_i$ пробігають всю вибірку):

$$y_i(\vec{x}_i \cdot \vec{w} + b) - 1 = 0$$

тут ми замість мінімізації функції $||\vec{w}||$ розглядаємо еквівалентну задачі мінімізації $||\vec{w}||^2$.

Оскільки графік цільової функції є параболоїдом, то вона має єдиний глобальний мінімум, який знаходиться методом методом множників Лагранжа.

Зустрічаються задачі класифікації коли класи не можна добре розділити прямою лінією. Проте якщо ми виконаємо деяке перетворення змінних і збільшимо розмірність простору то в нових координатах данні можуть розділиться і до них вже можна буде застосувати метод опорних векторів. Проте для початкового набору даних розділююча лінія уже не буде прямою лінією. Виявляється для методу опорних векторів немає потреби перераховувати всі данні в нових координатах, для цього достатньо перерахувати новий скалярний добуток. Новий скалярний добуток задається функцією, яка називається *ядром*, а сам метод називається *ядровим трюком.*

Приклад розділення класів нелінійним ядром

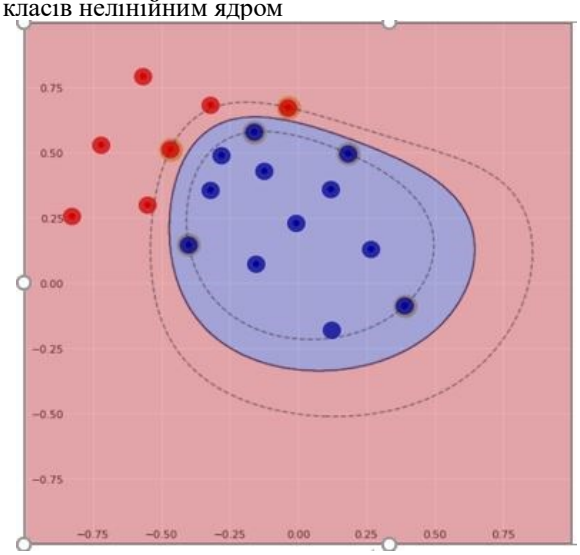

Рис 2. Області прийняття рішень

Часто використовуються такі ядра – поліноміальні, гаусівські, сигмоїдне. Гаусівське ядро має вигляд

$$K(x, y) = e^{-\gamma||x-y||^2}.$$

Константа γ визначає міру близькості між точками x, y і є гіперпараметром моделі.

2.4. Наївний баєсівський класифікатор. В основі класифікації лежить гіпотеза максимальної правдоподібності, тобто вважається, що об'єкт $\boldsymbol{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_n})$ належить класу $\boldsymbol{c}^*$ з множини класів $\boldsymbol{C}$ якщо на $\boldsymbol{c}^*$ досягається найбільша апостеріорна (після випробовування) ймовірність

$$c^* = \underset{c \in C}{\operatorname{argmax}} P(c_i|x).$$

За формулою Баєса

$$P(c|x) = \frac{P(c)P(x|c)}{P(x)}$$

Оскільки $P(x)$ константа, то

$$\underset{C}{\operatorname{argmax}} P(c|x) = \underset{C}{\operatorname{argmax}} \frac{P(c_i)P(x|c)}{P(x)} = \underset{C}{\operatorname{argmax}} P(c)P(x|c)$$

Далі робиться "наївне" припущення що для кожного класу ознаки об'єкту x незалежні між собою, тобто

$$P(x_i|c, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i|\mathrm{c}).$$

Звідси

$$P(x|c) = P(x_1, x_2, \dots, x_n|c) = P(x_1|c)P(x_2|c) \cdots P(x_n|c) = \prod_{i=1}^{n} P(x_i|c).$$

Тоді

$$c^* = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i=1}^{n} P(x_i|c).$$

На практиці, щоб уникнути роботи з малими числами, застосовують логарифмування

$$c^* = \underset{c \in C}{\operatorname{argmax}} \log\left(P(c) \prod_{i=1}^{n} P(x_i|c)\right) = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i=1}^{n} \log P(x_i|c).$$

Це можна робити оскільки логарифм є монотонно зростаючою функцією і логарифмування не не змінить точку максимуму.

2.5. Метод *k*-найближчих сусідів. Алгоритм $\boldsymbol{k}$ найближчих сусідів є найпростішим непараметричним алгоритмом машинного навчання. Побудова моделі полягає в запам'ятовуванні навчального набору даних. Для того, щоб зробити прогноз для нової точки даних, алгоритм знаходить найближчі до неї точки навчального набору, тобто знаходить «найближчих сусідів». Замість того, щоб враховувати лише одного найближчого сусіда, ми можемо розглянути будь-яку кількість ($\boldsymbol{k}$) сусідів. У цьому випадку для присвоєння мітки використовується голосування. Це означає, що для кожної точки тестового набору ми підраховуємо кількість сусідів, що відносяться до класу 0, і кількість сусідів, що відносяться до класу 1. Потім ми присвоюємо точці тестового набору найбільш часто зустрічається клас: іншими словами, ми вибираємо клас, який набрав більшість серед $\boldsymbol{k}$ найближчих сусідів.

3. Реалізація алгоритму та результати обчислень.

3.1. Датасет. Для обчислень ми використаємо набір масивів випадкових чисел які мають рівномірний розподіл на $(-\mathbf{1}, \mathbf{1})$. В кінці масиву додаємо елемент який вказує номер позиції максимального елементу

масиву.

Код на Python для формування набору даних:

```
nn=100000 # Кількість масивів
ss=5# Розмірність масиву
# додаємо стовпчик із позицій максимальних елементів
data = pd.DataFrame(np.random.randn(nn, ss))      data['F']=data[list(range(ss))].idxmax(axis=1)
X=data.iloc[:,:-1].values.reshape(-1,ss)
y=data.iloc[:,-1:].values.reshape(-1,1)
```

Приклад даних: шість масивів з п'яти елементів:

Таблиця 1.

Приклад датасету

0	1	2	3	4	Позиція макс. елемента
-0.084439	1.454161	0.634093	-0.837453	-0.720010	1
.117342	-0.156516	0.169550	-1.318699	-0.767414	2
.604625	0.154779	-0.921115	1.452857	1.239920	0
-0.339263	1.201032	-0.204102	-1.556671	-0.083587	1
.568657	-0.289558	-1.050814	2.101250	-1.085881	3

В останньому стовпчику знаходиться номер позиції (нумерація починається з 0) максимального елемента масиву який знаходиться у відповідному рядку.

Для аналізу алгоритмів для масивів довжини 2,3,4,10,20 сформуємо 7 датасетів розміром 100,500,1000,2000,5000,10000,100000 масивів.

3.2 Метод опорних векторів. Класифікацію проведемо методом опорних векторів нерегуляризованою моделлю з гаусівським ядром. Точності отриманих моделей наведено в таблиці нижче.

Таблиця 2.

Точність моделі опорних векторів з гаусівським ядром

	100	500	1000	2000	5000	10000	100000
розмірність 2	1.00	0.992	0.992	0.992	0.9928	0.9980	0.99916
розмірність 3	0.92	0.936	0.956	0.986	0.9864	0.9936	0.99784
розмірність 4	0.84	0.928	0.936	0.944	0.9592	0.9668	0.99152
розмірність 10	0.72	0.640	0.820	0.828	0.8888	0.9188	0.96160
розмірність 20	0.20	0.536	0.660	0.720	0.7936	0.8184	0.90716

В першому рядку вказано розмір датасету.

Візуалізація точності обчислень

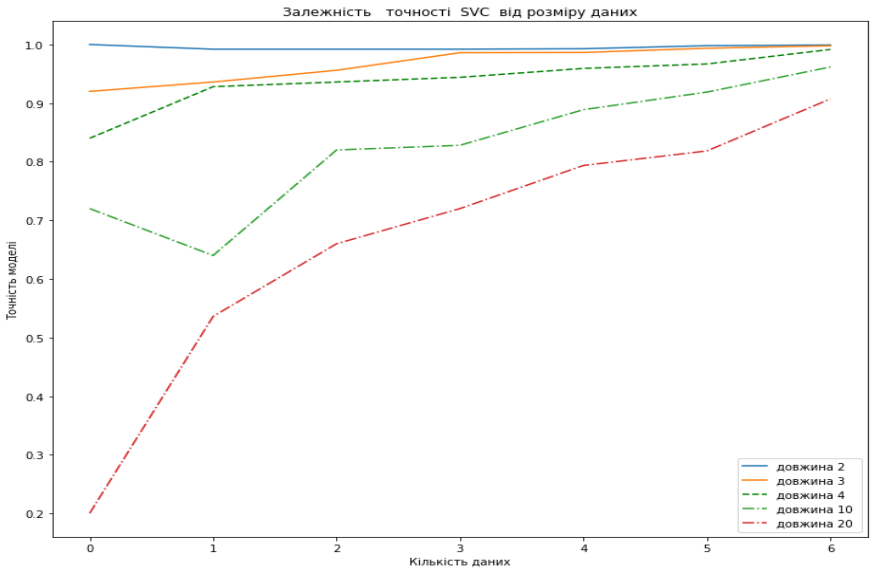

Рис 3. Графіки точності моделі опорних векторів з гаусівським ядром

Точність лінійної моделі

Таблиця 3.

Точність лінійної моделі опорних векторів.

	100	500	1000	2000	5000	10000	100000
розмірність 2	1.00	0.984	0.984	0.994	0.9928	0.9952	0.99904
розмірність 3	0.84	0.984	0.992	0.992	0.9960	0.9944	0.99900
розмірність 4	0.92	0.960	0.944	0.976	0.9808	0.9828	0.99736
розмірність 10	0.76	0.808	0.808	0.884	0.9360	0.9592	0.98824
розмірність 20	0.24	0.512	0.648	0.678	0.8200	0.8640	0.96192

Візуалізація точності обчислень.

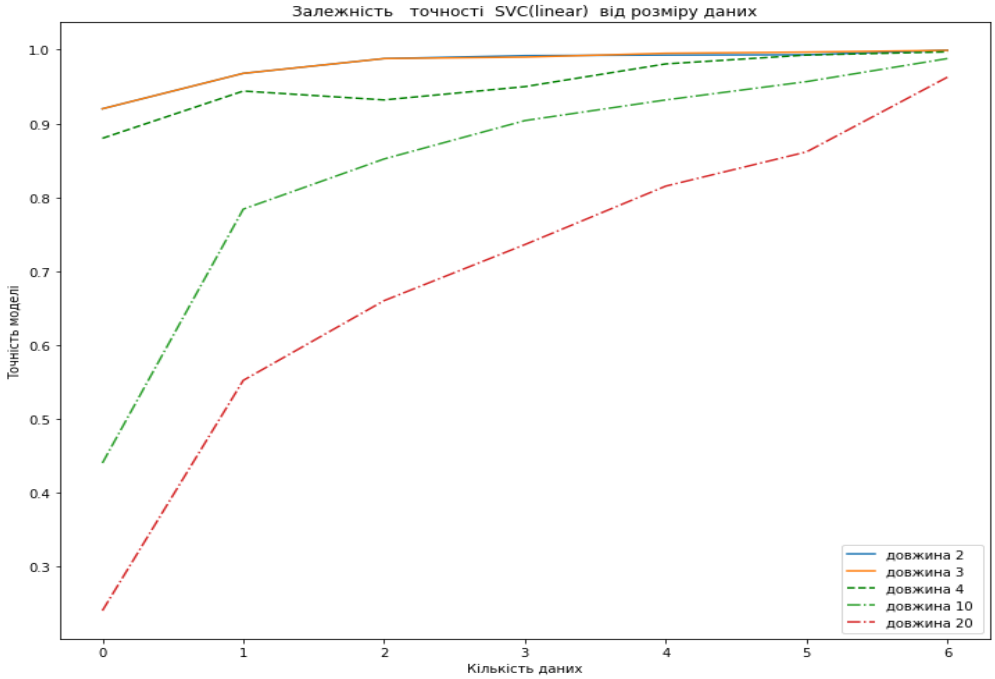

Рис 4. Графіки точності моделі опорних векторів з лінійним ядром

3.3 Метод k-найближчих сусідів. Виконуємо класифікацію методом k-найближчих сусідів при $k = 5$.

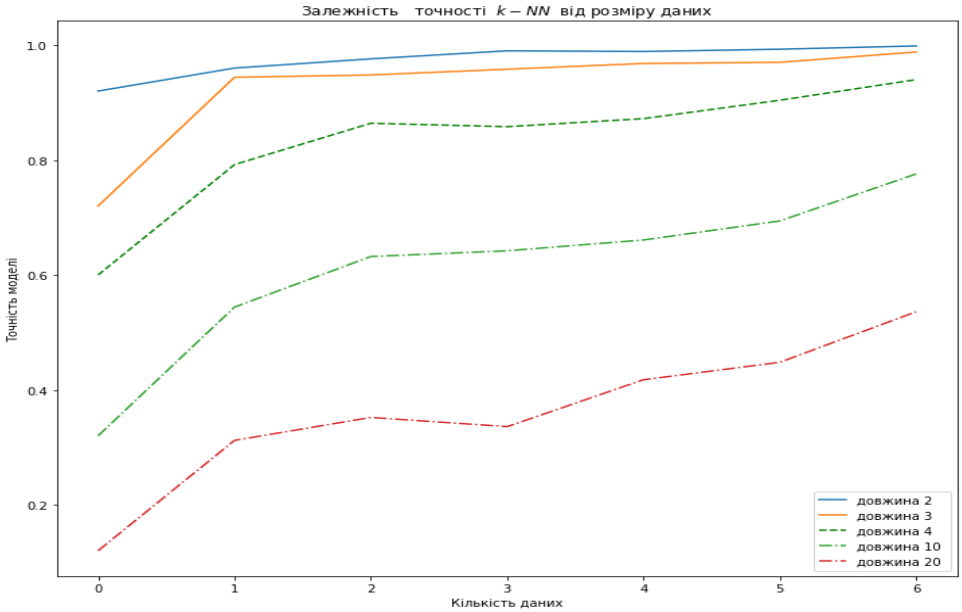

Рис 5. Графіки точності моделі k-найближчих сусідів

Таблиця 4.

Точність моделі *k*-найближчих сусідів.

	100	500	1000	2000	5000	10000	100000
розмірність 2	0.92	0.960	0.976	0.990	0.9888	0.9928	0.99848
розмірність 3	0.72	0.944	0.948	0.958	0.9680	0.9700	0.98796
розмірність 4	0.60	0.792	0.864	0.858	0.8720	0.9040	0.93992
розмірність 10	0.32	0.544	0.632	0.642	0.6608	0.6940	0.77584
розмірність 20	0.12	0.312	0.352	0.336	0.4176	0.4480	0.53592

3.4. Наївний класифікатор Баєса. Проведемо класифікацію наївним гаусівським класифікатором Баєса. Результати класифікації наведені в таблиці.

Таблиця 6.

Точність моделі наївним гаусівським класифікатором Баєса.

	100	500	1000	2000	5000	10000	100000
розмірність 2	0.96	0.968	0.964	0.974	0.9872	0.9952	0.99360
розмірність 3	0.76	0.888	0.980	0.980	0.9864	0.9836	0.99688
розмірність 4	0.72	0.936	0.928	0.940	0.9696	0.9808	0.99188
розмірність 10	0.56	0.784	0.872	0.868	0.9240	0.9440	0.98176
розмірність 20	0.20	0.536	0.644	0.762	0.8216	0.8760	0.95440

Візуалізація точності обчислень

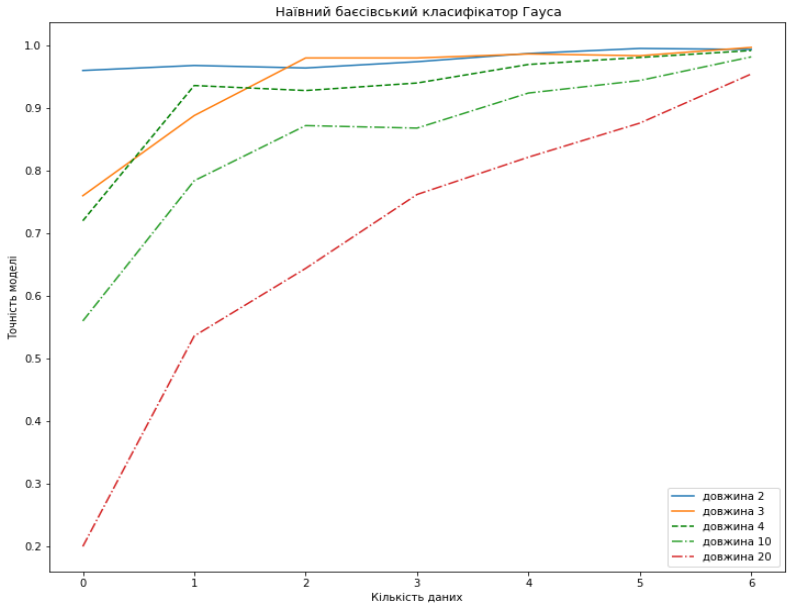

Рис 6. Графіки точності наївного класифікатора Баєса

3.5. Дерева прийняття рішень. Проводимо класифікацію Деревом (критерій entropy)

Таблиця 7.

Точність моделі дерева прийняття рішень (ентропія).

	100	500	1000	2000	5000	10000	100000
розмірність 2	0.84	0.960	0.964	0.980	0.9872	0.9948	0.99760
розмірність 3	0.88	0.992	0.964	0.974	0.9872	0.9876	0.99656
розмірність 4	0.68	0.912	0.932	0.954	0.9640	0.9796	0.99196
розмірність 10	0.52	0.720	0.836	0.892	0.9184	0.9512	0.98148
розмірність 20	0.16	0.600	0.664	0.738	0.8296	0.8836	0.95784

Візуаліазція точності моделей

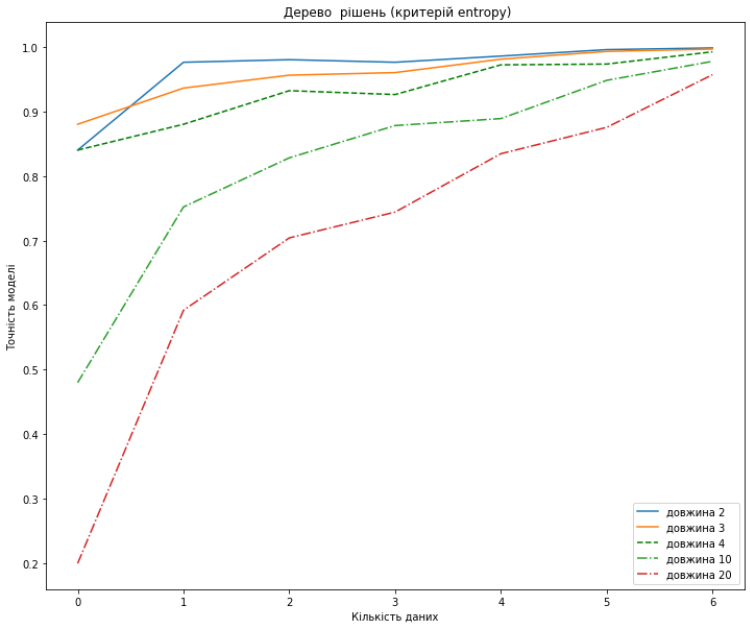

Рис 7. Графіки точності моделі дерева прийняття рішень

Проводимо класифікацію Деревом (критерій Джині)

Таблиця 8.

Точність моделі дерева прийняття рішень (Джині)

	100	500	1000	2000	5000	10000	100000
розмірність 2	0.84	0.976	0.980	0.976	0.9856	0.9956	0.99820
розмірність 3	0.88	0.936	0.956	0.960	0.9808	0.9928	0.99648
розмірність 4	0.84	0.880	0.932	0.926	0.9720	0.9732	0.99216
розмірність 10	0.48	0.752	0.828	0.878	0.8888	0.9480	0.97744
розмірність 20	0.20	0.592	0.704	0.744	0.8344	0.8752	0.95672

Візуаліазція точності моделей

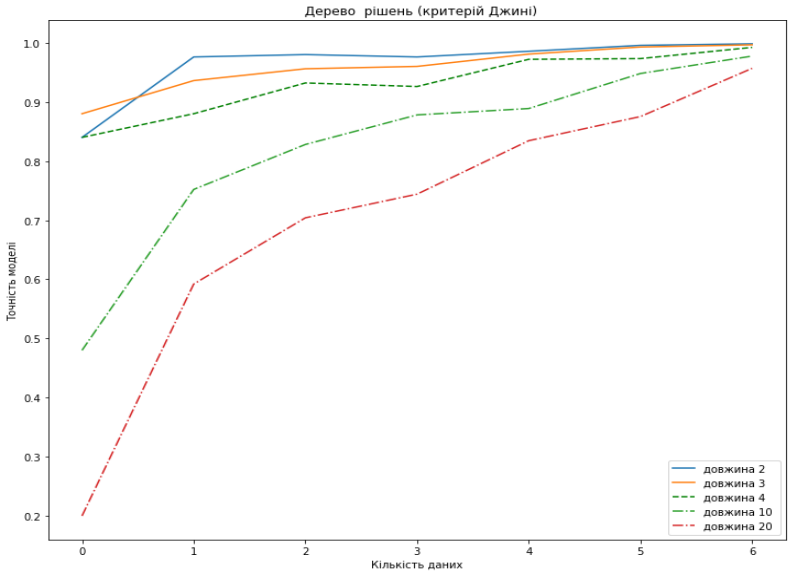

Рис 8. Графіки точності моделі дерева прийняття рішень (Джині)

Висновки з даного дослідження і перспективи подальших розвідок у даному напрямі

В результаті проведених обчислень було встановлено, що практично всі моделі класичного навчання можна навчити розпізнавати максимальний елемент масиву. Було встановлено що найбільшу точність для двохелементного масиву дає метод опорних векторів з гаусівським ядром - 99.916%, а найменшу точність дав наївний баєсівський класифікатор з гаусівським ядром – 99.36%. Найбільшу точність 96.192 % для масиву із 20 елементів дає лінійний класифікатор, а найменшу точність – 53.59% дає метод найближчих ***k***-сусідів.

Найкраща точність досягалася на датасеті із 100 000 масивів. Таким чином, встановлено, що класичні алгоритми можуть з великою точністю знаходити найбільший елемент масиву, і тому алгоритми сортування з використанням класичних методів навчання нічим не поступаються методам глибокого навчання.

References

1. Jayadeva, Rahman S. A., "A neural network with O(N) neurons for ranking N numbers in O(1/N) time," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 51, no. 10, pp. 2044-2051, Oct. 2004
2. Tambouratzis, T.,"A novel artificial neural network for sorting," IEEE Trans. Systems, Man, Cybern., vol. 29, pp. 271–275, Apr. 1999.
3. Lin, S. S., Hsu, S. H., A low-cost neural sorting network with O(1) time complexity, Neurocomputing — Int. J. 14 (1996), 289–299.
4. Koutroumbas, K., Kalouptsidis, N.: Neural networks architectures for selecting the maximum input, Int. J. Comput. Math., 67 (1998), 25–32.
5. Кормен Т., Лейзерсон Ч., Роналд Л. Рівест Р., Стайн К., Вступ до алгоритмів. — К. : К. І. С., 2019. — 1288 с
6. Koutroumbas K., Recurrent Algorithms for Selecting the Maximum Input, Neural Processing Letters 20: 179–197, 2004.
7. Mitchell T., Machine Learning, McGraw Hill, 1997.