DOI 10.34185/1562-9945-6-155-2024-12
УДК 004.42:519.25

О.Д. Земляний, О.Г. Байбуз

АЛГОРИТМИ ІМПУТУВАННЯ ПРОПУСКІВ У ДАНИХ НА ОСНОВІ ЕНТРОПІЇ

Анотація.Пропонується два алгоритми імпутування пропусків у даних (неітераційний та ітераційний) в задачах класифікації, оснований на мінімізації умовної ентропії. Розглядається рішення для кількісних та якісних ознак, у випадку кількісних – для дискретних та неперервних. Для аналізу алгоритмів пропонується три типи тестів. Перший тест працює з повним датасетом, в який штучно вносяться пропуски, проводиться імпутування різними методами, порівняння алгоритмів відбувається за середньоквадратичною похибкою та часом виконання алгоритмів. Другий тип тесту порівнює умовну ентропію до та після імпутування для різних методів. Третій тип тесту пов'язаний із задачею класифікації, коли моделі бінарної класифікації навчають на датасетах з імпутованими різними методами даними, та порівнюють точність класифікації на тестових вибірках. Розглядаються два відомих датасети про спостереження стосовно ішемічної хвороби серця.
Ключові слова: *імпутування даних, пропуски у даних, умовна ентропія, теорія інформації, алгоритми обробки даних, мінімізація невизначеності, класифікація, якісні та кількісні ознаки, ітераційний метод, ентропійний підхід, машинне навчання, обробка відсутніх даних, взаємна інформація, інженерія програмного забезпечення, інтелектуальний аналіз даних, діаграма діяльності.*

Вступ. Пропуски в даних можуть виникати з різних причин і мають різні характеристики. Розуміння типів пропусків у даних є важливим для вибору відповідних методів обробки та аналізу даних. Існують три основні типи пропусків [1]:

1. MCAR (Missing Completely at Random) — пропуски, що виникають повністю випадково. Пропуски вважаються повністю випадковими, якщо відсутність значення у змінній не пов'язана ні з якими іншими змінними або з самим значенням змінної. Наприклад, якщо при введенні даних деякі значення відсутні через випадкові помилки в процесі запису даних або спостереження не відбулось з випадкової причини, і ці пропуски не залежать від значень самих даних, то ці пропуски можна вважати MCAR. Аналіз даних за умов MCAR є найменш проблематичним, оскільки пропуски є випадковими і не упереджують результати аналізу.

2. MAR (Missing at Random) — пропуски, що виникають випадково. Пропуски вважаються випадковими, якщо ймовірність виникнення пропуску може залежати від інших спостережуваних даних, але не залежить від неспостережуваних значень, тобто пропуски можна пояснити іншими змінними в наборі даних. Наприклад, припустимо,

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

що люди з низьким доходом в опитуванні частіше не відповідають на питання про дохід, але це все одно можна передбачити на основі інших даних, таких як рівень освіти чи професія. У такому випадку пропуски можна вважати MAR. Пропуски типу MAR можуть бути оброблені такими методами, як множинна імпутація або регресійна імпутація, які враховують зв'язки між змінними для прогнозування відсутніх значень. Методи, які не враховують ці зв'язки, можуть призвести до зміщених результатів.

3. MNAR (Missing Not at Random) — пропуски, що не виникають випадково. Пропуски вважаються невипадковими, якщо ймовірність виникнення пропуску залежить від неспостережуваних значень або самого відсутнього значення. Це означає, що сам факт відсутності даних пов'язаний з конкретними характеристиками даних. Наприклад, якщо люди з високим рівнем доходу не хочуть відповідати на питання про свій дохід у соціальному опитуванні, і ця інформація про дохід недоступна, то пропуски не є випадковими (MNAR). У цьому випадку відсутні дані безпосередньо залежать від значень самих даних. Пропуски типу MNAR є найбільш складними для обробки, оскільки відсутність даних залежить від самих даних, які відсутні. Для коректного аналізу потрібно робити додаткові припущення або використовувати спеціалізовані методи моделювання пропусків.

Розуміння типів пропусків у даних є критично важливим для вибору правильних методів обробки даних. Правильне ідентифікування типу пропусків (MCAR, MAR або MNAR) дозволяє вибрати відповідні методи заповнення пропусків або коригування моделей, що забезпечує більш точні та надійні результати аналізу. Дана робота орієнтована на перші два типи пропусків MCAR та MAR.

Постановка проблеми в загальному вигляді. У задачі класифікації ми маємо набір даних, який містить декілька ознак $X = \{X_1, X_2, \ldots, X_n\}$ та мітку класу Y, яка приймає значення з певної множини класів $\{C_1, C_2, \ldots, C_k\}$. Метою є навчити класифікатор, який може передбачати мітку класу Y на основі вхідних ознак X.

Проте в реальних наборах даних деякі значення в матриці ознак X є відсутніми (позначається як NaN або інші спеціальні значення). Відсутні дані ускладнюють процес навчання класифікатора, оскільки багато алгоритмів машинного навчання не можуть обробляти пропуски безпосередньо. Пропуски даних можуть вплинути на класифікацію різними способами:

- зниження точності моделі, оскільки модель навчається на неповних даних і може пропустити важливі закономірності;
- зсув оцінок моделі, тобто якщо пропуски не є випадковими (випадок MNAR), модель може вивчити упереджені оцінки, що призведе до неправильних прогнозів;
- скорочення розміру вибірки для навчання, якщо обирати метод видалення рядків з пропусками, що може призвести до менш стабільних моделей.

Метою імпутування пропусків є заповнення відсутніх значень таким чином, щоб зменшити вплив пропусків на результати класифікації та забезпечити якомога точніше навчання моделі.

Аналіз останніх досліджень. В нашому попередньому дослідженні [2] описано популярні підходи та запропоновано декілька власних методів імпутування пропусків у даних на основі машинного навчання. Крім того, останнім часом набуває популярності глибинне навчання для імпутування пропусків, особливо у складних і великих наборах даних. В роботі [3] представлено метод з використанням генеративних змагальних мереж, в роботі [4] пропонується використовувати автоенкодери. Приділяється увага ансамблевим методам, наприклад, MissForest [5] та XGBoost [6, 7]. В роботі [8] представлено ідею методу імпутування пропущених даних на основі ентропійного підходу. Дана робота представляє алгоритм імпутування даних на основі мінімізації умовної ентропії.

Мета роботи. Метою роботи є розроблення алгоритму імпутування пропусків у даних, які представлені якісними та кількісними ознаками, в тому числі дискретними та неперервними даними.

Основна частина. Метод ентропійної інтерполяції відсутніх даних у задачах класифікації використовує поняття ентропії з теорії інформації для керування процесом заповнення відсутніх значень у наборах даних. У цьому контексті ентропія вимірює кількість невизначеності або випадковості в даних. Метою такого методу є зменшення невизначеності щодо міток класів шляхом вибору найбільш інформативних значень для відсутніх даних.

В теорії інформації ентропія є мірою невизначеності або випадковості. Для випадкової величини Y з розподілом ймовірностей *P(Y) ентропія H(Y)* визначається як:

$$H(Y) = -\sum_{y \in Y} P(y)\, log\, P(y).$$

В контексті інтерполяції пропущених даних ентропія може бути використана для кількісної оцінки невизначеності, пов'язаної з інтерполяцією пропущених значень.

У задачах класифікації ентропія може бути використана для вимірювання невизначеності міток класів за даними спостережень. Якщо деякі ознаки відсутні, мета полягає в тому, щоб врахувати ці відсутні значення таким чином, щоб мінімізувати ентропію передбачуваних міток класів, тим самим покращуючи ефективність класифікації.

Для дискретної випадкової величини *Y* з можливими класами $C_1, C_2, \ldots, C_k$ і ймовірностями $P(C_1), P(C_2), \ldots, P(C_k)$ ентропія *H(Y)* визначається як:

$$H(Y) = -\sum_{i=1}^{k} P(C_i)\, log_2\, P(C_i) \qquad (1)$$

Основа логарифма визначає одиницю ентропії. Зазвичай основа дорівнює 2 та одиниця ентропії – це біти.

На практиці, щоб обчислити ентропію для класів, потрібно виконати наступні кроки:

1. Порахувати кількість входжень кожного класу в наборі даних.
2. Обчислити ймовірності кожного класу: поділити кількість кожного класу на загальну кількість рядків даних.
3. Обчислити ентропію розподілу класів за формулою (1).

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Для набору даних з кількома ознаками ентропія класів може змінюватися в залежності від значення конкретної ознаки. Умовна ентропія вимірює невизначеність класів за умови заданої ознаки.

Для ознаки *X* з можливими значеннями $x_1, x_2, \ldots, x_m$ *умовна ентропія* $H(Y|X)$ визначається за формулою

$$H(Y|X) = \sum_{j=1}^{m} P(x_j) H(Y|X = x_j), \quad (2)$$

де

$P(x_j)$ – ймовірність того, що ознака X приймає значення x_j;

$H(Y|X = x_j)$ – ентропія Y за умови $X = x_j$, яка обчислюється наступним чином

$$H(Y|X = x_j) = -\sum_{i=1}^{k} P(C_i|X = x_j) \log_2 P(C_i|X = x_j). \quad (3)$$

З формул (2) та (3) отримаємо наступну загальну формулу для умовної ентропії

$$H(Y|X) = -\sum_{j=1}^{m} P(x_j) \sum_{i=1}^{k} P(C_i|X = x_j) \log_2 P(C_i|X = x_j) \quad (4)$$

На практиці, щоб обчислити умовну ентропію, потрібно виконати наступні кроки:

1. Обчислити $P(C_i|X = x_j)$ – умовні ймовірності для кожного значення класу C_i за умови значення ознаки x_j, тобто порахувати для кожного класу C_i, скільки в наборі даних рядків, для яких $Y = C_i$ та $X = x_j$.
2. Визначити ентропію для кожного значення ознаки за формулою (3).
3. Зважити ентропії на ймовірність кожного значення ознаки та підсумувати їх.

Взаємна інформація, або *інформаційний виграш* показує, наскільки знання ознаки *X* зменшує невизначеність щодо *Y*, та обчислюється за формулою

$$G(Y, X) = H(Y) - H(Y|X). \quad (5)$$

Розглянемо *приклад обчислення*. Нехай в нас є набір даних, що має одну ознаку X з двома можливими значеннями x_1 та x_2 та бінарним класом *Y* з двома можливими значеннями C_1 та C_2. В таблиці 1 представлено кількість рядків, що відповідають значенням x_j та C_i

Бачимо, що в нас є набір даних, що містить:

100 рядків, де 60 належать до класу C_1 і 40 - до класу C_2;

x_1 з'являється у 70 рядках: 50 з C_1 і 20 з C_2;

x_2 з'являється у 30 рядках: 10 з C_1 і 20 з C_2.

Таблиця 1

Набір даних прикладу

	C_1	C_2	Разом
x_1	50	20	70
x_2	10	20	30
Разом	60	40	100

Обчислимо ентропію *H(Y)* за формулою (1):

$$P(C_1) = \frac{60}{100} = 0.6, \qquad P(C_2) = \frac{40}{100} = 0.4$$

$$H(Y) = -(0.6\,log_2\,0.6 + 0.4\,log_2\,0.4) \approx 0.97$$

Обчислимо ентропію для кожного значення ознаки за формулою (3):

$$P(C_1|X = x_1) = \frac{50}{70} = \frac{5}{7},\quad P(C_2|X = x_1) = \frac{20}{70} = \frac{2}{7},$$

$$H(Y|X = x_1) = -\left(\frac{5}{7}\log_2\frac{5}{7} + \frac{2}{7}\log_2\frac{2}{7}\right) \approx 0.86,$$

$$P(C_1|X = x_2) = \frac{10}{30} = \frac{1}{3},\quad P(C_2|X = x_2) = \frac{20}{30} = \frac{2}{3},$$

$$H(Y|X = x_2) = -\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) \approx 0.92,$$

Загальна умовна ентропія $H(Y|X)$ обчислюється за формулою (4):

$$H(Y|X) = \frac{70}{100} \times 0.86 + \frac{30}{100} \times 0.92 \approx 0.88$$

Обчислимо взаємну інформацію за формулою (5):

$$G(Y,X) = H(Y) - H(Y|X) = 0.97 - 0.88 = 0.09.$$

Головна ідея методу полягає в виборі для ознаки X серед всіх можливих значень $x_1, x_2, \ldots, x_m$ такого значення x_j для імпутування, яке буде максимізувати взаємну інформацію $G(Y,X) \to max$. Оскільки $H(Y)$ не залежить від X, то нам достатньо мінімізувати умовну ентропію, тобто знайти такий x_j, для якого $H(Y|X) \to min$.

Пропонується виконувати імпутування кожної ознаки окремо за наступним алгоритмом. Спочатку розглянемо випадок дискретної випадкової величини (категоріальна ознака).

Алгоритм 1. Імпутування категоріальної ознаки

1. Розглянемо піднабір даних *D*, що складається з ознаки *X* та цільового стовпця *Y* . Розділимо дані на дві множини: повна D^f та неповна D^{nf}.

2. Беремо перший рядок з D^{nf} і виконуємо крок підбору значення, що імпутується.

3. Крок підбору значення: обчислюємо умовну ентропію для кожного можливого значення x_j, $j \in [1, \ldots, m]$ на множині D^f, до якої доєднаний x_j.

4. Вибираємо в якості значення, що імпутується, таку x_j, яка мінімізує умовну ентропію.

5. Виконуємо оновлення набору даних, виконавши імпутацію в рядку, що розглядається.

6. Повторюємо кроки 1 – 5, поки множина D^{nf} непуста.

Також пропонується модифікація алгоритму 1, де використовується декілька ітерацій підбору, поки загальна умовна ентропія продовжує зменшуватись.

Фактично виявляється, що для пропущеного значення класу C_i значення x_j максимізує умовну ймовірність $P(C_i|X = x_j)$ в заповненому наборі даних.

Алгоритм 2. Імпутування категоріальної ознаки ітераційно

1. Встановлюємо максимально припустиму кількість ітерацій.

2. Встановлюємо номер ітерації в 1.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

3. Встановлюємо поточне значення умовної ентропії у заздалегідь задане велике число, наприклад, 10000.

4. Якщо номер ітерації не перевершує максимально припустиму кількість, переходимо до наступного кроку, інакше завершуємо алгоритм.

5. Запам’ятовуємо поточний вміст ознаки *X*.

6. Виконуємо алгоритм 1.

7. Обчислюємо умовну ентропію після імпутації.

8. Якщо значення умовної ентропії зменшилось у порівнянні з попереднім кроком ітерації, переходимо на крок 9, інакше – переходимо на крок 11.

9. Виконуємо оновлення набору даних, прийнявши всі імпутації.

10. Збільшуємо номер ітерації та переходимо на крок 4.

11. Якщо значення умовної ентропії не зменшилось у порівнянні з попереднім кроком ітерації, виконуємо відкат до попередньої версії імпутації та завершуємо алгоритм.

Для наочності представимо діаграми діяльності UML обох алгоритмів на рисунку 1.

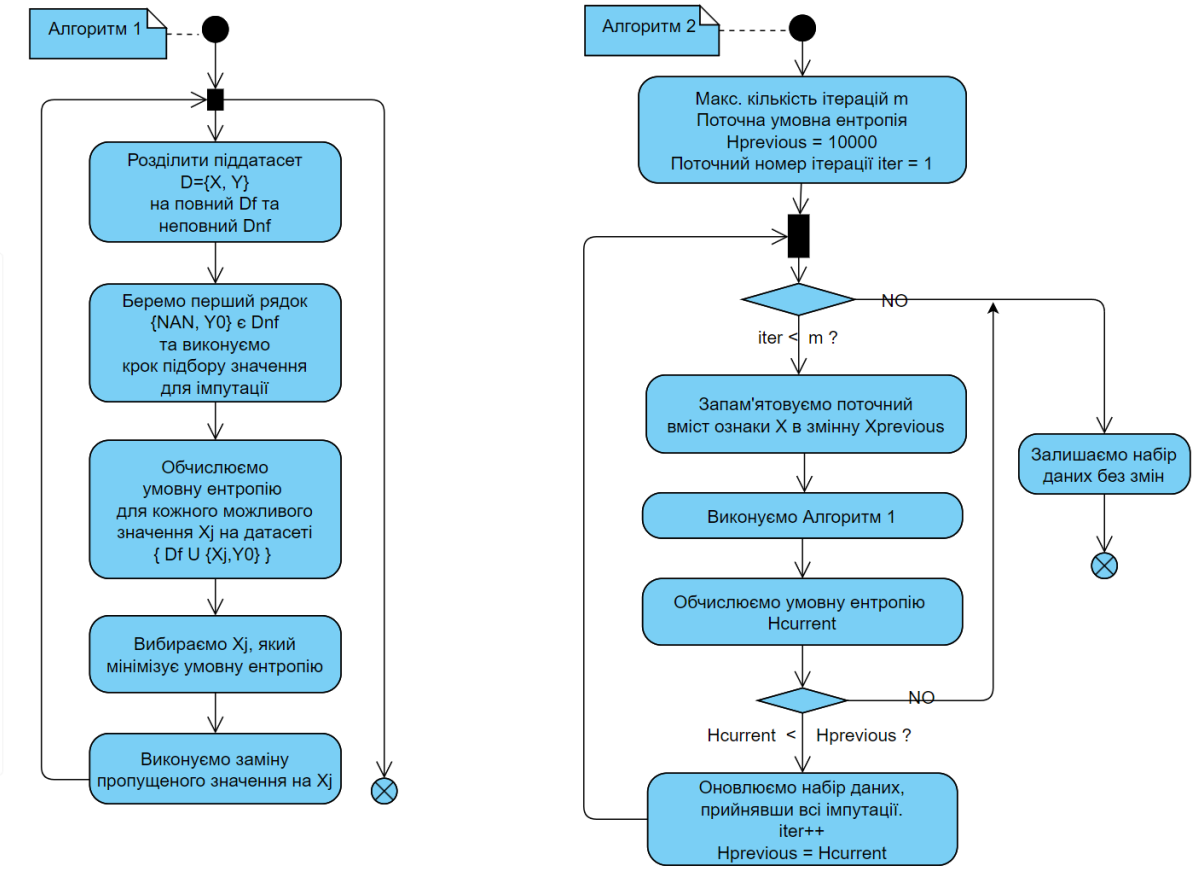

Рисунок 1 – Діаграми діяльності UML алгоритмів 1 та 2

В наших обчисленнях виявилось, що потрібно виконати максимум 3 – 4 ітерації, після чого умовна ентропія вже перестає зменшуватись.

Якщо наша ознака *якісна порядкова,* пропонується виконати крок перетворення у кількісну, описаний у в нашій попередній роботі [2]. Користь цих методів полягає в тому, що вони дають можливість перетворити всі якісні ознаки на кількісні без втрати інформації про пропуски. Після цього ми отримаємо випадок дискретної величини.

Якщо наша ознака відноситься до *неперервних* величин, ми можемо *звести* задачу до дискретного випадку, виконавши дискретизацію.

Пропонується *кількість* проміжків обчислювати за кількістю класів варіаційного ряду за формулою

$$M = 1 + 3.32 \cdot lg(n), \quad (6)$$

де n – кількість рядків даних в множині повних даних D^f.

Таким чином, ми перейдемо знову до випадку дискретної величини. Для виконання зворотного перетворення, коли ми будемо з номеру імпутованого інтервалу отримувати значення, пропонується генерувати випадкову величину з нормального розподілу $\mathrm{N}(\mathrm{m};\ \sigma)$, де m – середнє значення для ознаки X з D^f, σ – середньо-квадратичне відхилення цієї ознаки.

Цей алгоритм є жадібним до ресурсів. Продуктивність може бути покращена за рахунок використання інженерних технік, таких як багатопоточність, оскільки задача дуже гарно може бути розділена на підзадачі за рахунок того, що імпутування кожної ознаки може бути виконано незалежно від інших в окремому потоці. Крім того, можливі покращення за рахунок застосування технік, специфічних для обраної мови програмування, наприклад, векторизація обчислень в мові Python.

Аналіз результатів застосування методів. В якості наборів даних використовуються два набори даних UCI Heart Disease Data [9, 10] та Framingham Heart Study [11, 12], в яких представлено дані спостережень ішемічної хвороби серця. Обидва набори даних містять пропуски типу MAR або MCAR.

В наборі даних UCI Heart Disease Data 920 рядків, 13 інформативних ознак та один цільовий стовпець. Серед 13 ознак 5 ознак – кількісні неперервні, 8 ознак – якісні порядкові, серед яких 2 – бінарні. В ході первинного аналізу даних виявлено наявність пропусків. Тільки 299 рядків з 920 є повними (33%). Балансування відбулося за рахунок переходу від 4 класів до 2 класів, тобто до задачі бінарної класифікації.

В наборі даних Framingham Heart Study 4240 рядків, 14 інформативних ознак та один цільовий стовпець з двома класами. Серед 14 ознак 8 ознак – кількісні неперервні, 6 ознак – якісні порядкові, серед яких 5 – бінарні. В ході первинного аналізу даних виявлено наявність пропусків. 3658 рядків з 4240 є повними (86%). Датасет виявився дуже незбалансованим, причому кількість екземплярів одного класу перевищує кількість екземплярів іншого класу в 5 разів. Для балансування використовується метод SMOTE [13].

Для аналізу запропонованих підходів використовується три типи тестів. Перший та другий тести розглядають повний датасет, куди штучно вносяться пропуски, користуючись механізмом MCAR для генерування пропусків у даних. Для цього згенерованого датасету виконується імпутування за допомогою різних методів, оцінюється сере-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

дньо-квадратична похибка імпутування та час виконання алгоритмів. Розглядаються 10% пропусків, 20%, 30%, 40%.

Другий тест вимірює умовну ентропію для кожного методу і виконує задачу тесту 1 ітераційно, поки зменшується ентропія для кожного методу. Ми виконуємо уточнення значень інтерполяції по кожній позиції, де був пропуск, при цьому вважаємо, що інші імпутовані значення належать датасету. Далі виконується порівняння значень умовної ентропії відносно початкової.

Третій тест стосується аналізу якості класифікації. Базовими моделями для порівняння якості класифікації виберемо дві моделі: отриману після навчання, використовуючи тільки повну частину датасету, та модель, де пропуски імпутовані найпростішим стандартним методом, наприклад, середнім значенням чи модою за допомогою *SimpleImputer* з бібліотеки scikit-learn [14] python. Виконуємо імпутування пропусків різними методами, навчаємо отримані датасети, отримуємо метрику accuracy (чи точність класифікації) на тестовій вибірці, порівнюємо значення цієї метрики для отриманих моделей. Для отримання стійких до зсуву результатів виконуємо процедуру розбиття датасетів на навчальну та тестові вибірки декілька разів, навчаємо моделі, отримуємо значення accuracy для поточної тестової вибірки, і наприкінці усереднюємо отримані значення. В третьому тесті для вихідних датасетів виконують імпутування пропущених даних різними методами, навчають моделі бінарної класифікації та порівнюють точність класифікації на тестовій вибірці. В якості методу класифікації використовується випадковий ліс (Random Forest).

В таблицях 2, 4 наведено результати тесту типу 1 для датасету UCI Heart Disease Data для алгоритмів 1 та 2 відповідно. В таблиці 3 та 5 представлено результати тесту типу 2 на тому ж датасеті для алгоритмів 1 та 2 відповідно. В таблиці 6 представлено результати запуску всіх методів ітераційно. В таблиці 7 ми наводимо результати тесту типу 3 для датасетів UCI Heart Disease Data та Framingham Heart Study.

Виконуємо тест першого типу для методів, аналогічно представленому переліку методів в роботі [2], куди додамо також наш новий метод на основі умовної ентропії. Візьмемо тільки повну частину датасету UCI Heart Disease Data. Згенеруємо 30% пропусків, додавши NAN у випадкові місця таблиці ознак. Оскільки методи на основі класифікації та регресії з роботи [2] перевіряють наявність патерну в пропусках, в умовах якого пропонується спеціальна реакція, що покращує результати методу, змоделюємо його наявність. Додамо патерн (рисунок 2), коли в нас є випадкова кількість повних стовпців (не більше половини), випадкова кількість повних рядків (не більше половини) та випадкова кількість рядків, в яких є тільки один пропуск.

Нам потрібно спочатку виконати перетворення якісних ознак на кількісні. Для цього застосовуємо три варіанти енкодерів, запропонованих в роботі [2]. Для енкодеру *le_non_missing* назвемо отриманий новий набір даних data_1, для енкодеру *le_non_missing_frequent(ascending=True)* - data_2, для енкодеру *le_non_missing_frequent(ascending=False)* - data_3.

Для набору даних data_1 виконуємо імпутування наступними методами: *SimpleImputer* – стандартне заповнення значенням з найбільшою частотою, *IterativeIm-*

puter(sample_posterior = False), kNNImputer, nonaImputer, fillna_k_columns, fillna_k_sorted_columns(ascending = True), fillna_k_sorted_columns(ascending = False), fillna_2steps_rg_class, ***fill_entropy_based_1step (алгоритм 1)***

Для набору даних data_2 виконуємо такий метод імпутування: *fillna_2steps_rg_class.*

Для набору даних data_3 виконуємо такі методи імпутування: *fillna_2steps_rg_class, fillna_2steps_rg(measure = 'value'), fillna_2steps_rg(measure = 'weight').*

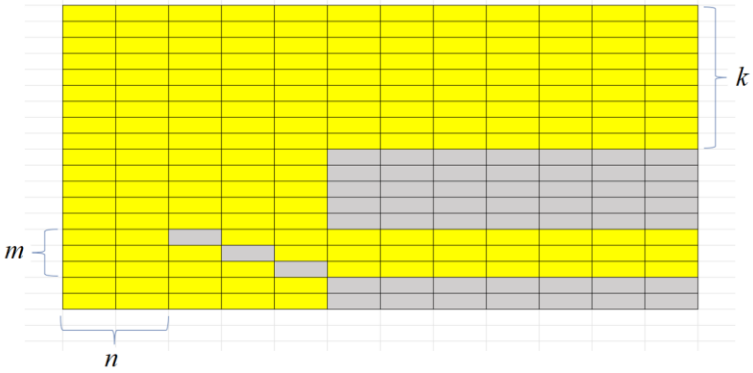

Рисунок 2 – Патерн для пропусків в даних
(жовтий колір – наявні дані, сірий колір – пропуск у даних)

В таблиці 2 представлено результати тесту типу 1, де наводяться середньо-квадратичні похибки імпутування та час виконання алгоритмів. Нас цікавлять результати запропонованого методу на основі ентропійного підходу у порівнянні з вже проаналізованими в попередній роботі.

З аналізу цього прикладу бачимо, що метод працює довше, ніж інші методи, але не набагато, хоча з точки зору метрики середньо-квадратичного відхилення він значно програє всім іншим методам.

Таблиця 2

Результат тесту типу 1 в порівнянні з імпутуванням
за допомогою алгоритму 1 (ентропійний неітераційний)

Назва методу	**Імпутування 30% пропусків**	**Час виконання**
data_1		
SimpleImputer	**3.47**	**0.008939**
IterativeImputer	**2.46**	0.162614
kNNImputer	2.75	**0.012442**
NonaImputer	2.54	1.082877
fillna_k_columns	2.55	1.083617
fillna_k_columns SortedAsc	2.62	1.080535
fillna_k_columns SortedDesc	2.51	1.071694

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

2steps	**2.51**	1.934818
entropyBased (алгоритм 1)	**9.59**	**2.345332**
data_2		
2stepsEncTrue	2.54	1.917507
data_3		
2stepsEncFalse	2.55	1.907776
2stepsRegValue	2.54	0.159464
2stepsRegWeight	2.55	**0.154090**

Порівняємо умовну ентропію для вказаних методів, тобто виконаємо тест типу 2. В таблиці 3 представлено значення умовної ентропії до та після імпутування. В якості прикладів візьмемо представників ознак різних типів (chol – це рівень холестерину, відноситься до неперервних величин, restecg – результат електрокардіографії у стані спокою, відноситься до якісних порядкових величин, sex – стать, відноситься до бінарних величин).

В таблиці 3 для набору даних data_1 жирним курсивом виділено базовий рядок зі значенням умовної ентропії до імпутування, далі жирним виділено рядок зі значеннями умовної ентропії для найкращого методу (на основі ентропії) та найгіршого (Simple). Стрілочками вверх чи вниз помічено яскраві приклади збільшення чи зменшення ентропії відносно базового значення.

З таблиці 3 можна побачити, що майже всі методи імпутування зменшують базову ентропію, але значний виграш дає саме метод, що оснований на ентропійному підході, причому для неперервної величини цей виграш найбільший і найменший – для бінарної величини.

Таблиця 3

Зміна значення умовної ентропії після застосування методів імпутування в порівнянні з алгоритмом 1

Назва методу	**chol**	**restecg**	**sex**
data_1	***0.786876***	***0.968004***	***0.967503***
SimpleImputer	**0.795206** ↑	**0.975178** ↑	**0.96852** ↑
IterativeImputer	0.781841	0.838706	0.952813
kNNImputer	0.785193	0.783078	0.952813
NonaImputer	0.784556	0.957632	0.967657
fillna_k_columns	0.784556	0.957632	0.967657
fillna_k_columns SortedAsc	0.781794	0.958821	0.96852
fillna_k_columns SortedDesc	0.795206	0.96003	0.967657
2steps	0.783231	0.961332	0.96852
entropyBased	**0.594163** ↓	**0.870099** ↓	**0.96741** ↓

(алгоритм 1)			
data_2	***0.786876***	***0.968004***	***0.967503***
2stepsEncTrue	0.78362 ↓	0.961307 ↓	0.96852 ↓
data_3	***0.786876***	***0.968004***	***0.967503***
2stepsEncFalse	0.78362	0.96013	0.96852
2stepsRegValue	0.783262 ↓	0.96133	0.967657 ↓
2stepsRegWeight	0.783262	0.961302 ↓	0.967657

Далі наведемо результати імпутування за допомогою алгоритму 2 (ентропійний ітераційний метод) у порівнянні з SimpleImputer та IterativeImputer (таблиця 4).

Таблиця 4

Результат тесту типу 1 в порівнянні з імпутуванням
за допомогою алгоритму 2 (ентропійний ітераційний)

Назва методу	**10% пропусків**	**20% пропусків**	**30% пропусків**	**40% пропусків**	**Час виконання**
SimpleImputer	1.576	3.03	2.27	3.15	0.039279
IterativeImputer	1.238	2.30	1.79	2.76	0.491748
entropyBased (алгоритм 2)	5.634	8.9	6.77	9.66	16.97883

В таблиці 5 представлено кроки зменшення умовної ентропії на кожній ітерації алгоритму 2, час виконання алгоритму 11.73 мс.

Таблиця 5

Кроки зменшення умовної ентропії на кожній ітерації алгоритму 2

Номер ітерації	**trestbps**	**chol**	**thalach**	**ca**
0	0.6140	0.683	0.526	0.779
1	0.5764	0.654	0.494	0.709
2	0.5760	-	0.492	0.708
3	-	-	-	0.706
4	-	-	-	0.705

Бачимо, що вже на 4-й ітерації ентропія перестала зменшуватись для всіх ознак.

Далі проведемо тест другого типу, коли ми всі методи запускаємо ітераційно, поки зменшується умовна ентропія або не буде досягнуто максимальне значення кількості ітерацій. Наведемо результати для декількох представників різних типів ознак та для датасету data_1 (таблиця 6).

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Таблиця 6

Зміна значення умовної ентропії
після застосування методів імпутування ітераційно

Назва методу	**chol**	**restecg**	**sex**
data_1	**0.791235** 0.790242 ↓	**0.970871** 0.972011 ↑	**0.967796** 0.96585 ↓
SimpleImputer	**0.796228** ↑ 0.753609	**0.975102** ↑ 0.972962	0.967657 ↑ 0.970045
IterativeImputer	0.791185 0.780171	0.938441 0.959821	0.949335 0.830418
kNNImputer	0.789338 0.783518	0.922918 0.959821	0.949519 0.820332
NonaImputer	0.789553 0.779655	0.968682 0.971386	0.967657 0.968787
fillna_k_columns	0.789553 0.779655	0.967621 0.971386	0.967657 0.968787
fillna_k_columns SortedAsc	0.787801 0.77828	0.964656 0.971386	0.967657 0.970045
fillna_k_columns SortedDesc	0.795379 0.782818	0.964656 0.971386	0.967657 0.968787
2steps	0.79057 0.778978	0.965685 0.971386	0.967657 0.969433
entropyBased	**0.622744** ↓ 0.613428 ↓	**0.875675** ↓ 0.961728 ↑	**0.960461** ↓ **0.976096** ↑

З таблиці 6 бачимо результати, аналогічні наведеним в таблиці 3, тобто Simple-метод не зменшує ентропію, інші методи її зменшують, але більший виграш з точки зору зменшення ентропії дає саме запропонований ентропійний метод. Також можна побачити, що ітерації уточнення майже не покращують результат, а навіть на другій ітерації можуть його погіршувати з точки зору умовної ентропії.

Далі розглянемо тест типу 3, який стосується аналізу якості класифікації. Базовими моделями для порівняння виступають дві моделі: отримана після навчання, використовуючи тільки повну частину датасету, та модель, де пропуски поповнені найпростішим стандартним методом, найчастішим значенням за допомогою *SimpleImputer* з бібліотеки scikit-learn [14] python. Будемо імпутувати пропуски різними методами, далі навчати класифікатор Random Forest та оцінювати метрику accuracy (чи точність класифікації) на тестовій вибірці. Значення цієї метрики будемо використовувати для порівняння отриманих моделей. Виконуємо процедуру розбиття датасетів на навчальну та тестові вибірки декілька разів, навчаємо моделі, отримуємо значення accuracy для поточної тестової вибірки, і наприкінці усереднюємо отримані значення. Така процедура

дозволить отримати стійкі до зсуву результати. В таблиці 7 жирним курсивом виділено базові результати та жирним шрифтом переможців за метрикою accuracy. Також жирним виділено результати ентропійного методу.

Таблиця 7

Результати тесту типу 3

Назва методу	**Датасет UCI HDD**		**Датасет Framingham FHS**	
	Найкращі гіперпараметри моделі	Точність	Найкращі гіперпараметри моделі	Точність
data_1				
Baseline	{'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 150}	***0.83***	{'max_depth': None, 'min_samples_leaf': 1, min_samples_split': 2, 'n_estimators': 150}	***0.88***
SimpleImputer	{'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 100}	***0.82***	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}	***0.87***
IterativeImputer(sample_posterior=False)	{'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 50}	0.88	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}	**0.90**
IterativeImputer(sample_posterior=True)	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 150}	0.82	{'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}	0.89
kNNImputer	{'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}	0.84	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}	**0.90**
nonaImputer	{'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 100}	0.89	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	0.89
fillna_k_columns	{'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 100}	0.89	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	0.89
fillna_k_columns SortedAsc	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}	0.89	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	0.89

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

fillna_k_columns SortedDesc	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 100}	**0.91**	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	0.89
2steps	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	**0.91**	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	0.88
entropyBased (алгоритм 2)	{'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 50}	**0.90**	**{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}**	**0.89**
data_2				
2stepsEncTrue	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}	0.89	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	0.88
data_3				
2stepsEncFalse	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 100}	0.89	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	0.89
2stepsRegValue	{'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 150}	0.89	{'max_depth': 20, 'min_samples_leaf': 1,min_samples_split': 2, 'n_estimators': 150}	0.88
2stepsRegWeight	{'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': 100}	**0.91**	{'max_depth': 20, 'min_samples_leaf':1,' min_samples_split':2, 'n_estimators': 150}	**0.90**

З аналізу результатів класифікації випливає, що запропонований ентропійний метод дозволяє покращити точність класифікації на рівні з найкращими методами, а на датасеті UCI Heart Disease Data результат класифікації покращено суттєво, порівняно з базовим результатом з 83% до 90%, порівняно з найпростішим імп'ютером - з 82% до 90%.

Висновки та перспективи подальших досліджень. У даній роботі запропоновано два алгоритми імпутування пропусків даних на основі ентропійного підходу: неітераційний (алгоритм 1) та ітераційний (алгоритм 2), ідея яких полягає в мінімізації умовної ентропії по кожній ознаці окремо.

Проведено три типи тестів на двох наборах даних. Аналіз показав, що запропоновані алгоритми є достатньо повільними у порівнянні з іншими методами і можуть бути покращені, наприклад, за рахунок багатопроцесорного виконання, як описано в нашій

роботі [15]. Тест типу 1 показав, що запропоновані алгоритми не дають виграш за метрикою середньо-квадратичного відхилення, але вони суттєво зменшують ентропію (тест типу 2). В той же час ці методи показують покращення результатів класифікації порівняно з базовими моделями (тест типу 3).

Таким чином, запропоновані методи імпутування на основі ентропійного підходу показали гарні результати та можуть претендувати на те, щоб стати додатковим інструментарієм дослідника для підвищення точності прийняття рішень, але потрібно виконувати подальші дослідження щодо оптимізації обчислень для збільшення швидкодії цих методів.

ЛІТЕРАТУРА / REFERENCES

1. Roderick J. A. Little, Donald B. Rubin. Statistical Analysis with Missing Data, 3rd Edition. -Wiley, 2019. - 464 p. ISBN: 978-0-470-52679-8

2. Земляний О.Д., Байбуз О.Г. Методи імпутування пропусків у даних про ішемічну хворобу серця // Системні технології. Регіональний міжвузівський збірник наукових праць. - Випуск 2(151). – Дніпро, 2024. – С.33 – 49. DOI: https://doi.org/10.34185/1562-9945-2-151-2024-04

3. Yoon, J., Jordon, J., & Schaar, M.V. (2018). GAIN: Missing Data Imputation using Generative Adversarial Nets. ArXiv, abs/1806.02920. DOI: https://doi.org/10.48550/arXiv.1806.02920

4. Gondara, L., & Wang, K. (2017). Multiple Imputation Using Deep Denoising Autoencoders. ArXiv, abs/1705.02737. DOI: https://doi.org/10.48550/arXiv.1705.02737

5. Stekhoven, D. J., & Bühlmann, P. (2012). MissForest — non-parametric missing value imputation for mixed-type data. Bioinformatics, 28(1), 112-118. DOI: https://doi.org/10.1093/bioinformatics/btr597

6. Rusdah, D.A., Murfi, H. XGBoost in handling missing values for life insurance risk prediction. SN Appl. Sci. 2, 1336 (2020). DOI: https://doi.org/10.1007/s42452-020-3128-y

7. Deng, Y., & Lumley, T. (2023). Multiple Imputation Through XGBoost. Journal of Computational and Graphical Statistics, 33(2), 352–363. DOI: https://doi.org/10.1080/10618600.2023.2252501

8. Delavallade, Thomas & Dang, Thanh. (2007). Using Entropy to Impute Missing Data in a Classification Task. Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'07, London, UK. 1 - 6. DOI: 10.1109/FUZZY.2007.4295430

9. Janosi, Andras, Steinbrunn, William, Pfisterer, Matthias, and Detrano, Robert. (1988). Heart Disease. UCI Machine Learning Repository. https://doi.org/10.24432/C52P4X

10. UCI Heart Disease Data. Heart Disease Data Set from UCI data repository. – [Електронний ресурс]. – Режим доступу: https://www.kaggle.com/datasets/redwankarimsony/heart-disease-data

11. Framingham Heart Study-Cohort (FHS-Cohort). – [Електронний ресурс]. – Режим доступу: https://biolincc.nhlbi.nih.gov/studies/framcohort/

12. Framingham heart study dataset. – [Електронний ресурс]. – Режим доступу: https://www.kaggle.com/datasets/aasheesh200/framingham-heart-study-dataset

13. N. V. Chawla, K. W. Bowyer, L. O.Hall, W. P. Kegelmeyer, "SMOTE: synthetic minority oversampling technique," Journal of artificial intelligence research, 321-357, 2002

14. Imputation of missing values in scikit-learn. – [Електронний ресурс]. – Режим доступу: https://scikit-learn.org/stable/modules/impute.html#impute

15. Земляний О.Д., Байбуз О.Г. Порівняння багатопроцесорної та багатопоточної реалізацій ентропійного підходу для імпутування пропусків у даних на мові програмування Python // Виклики та

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

проблеми сучасної науки [Електронний ресурс]: зб. наук. пр. – Дніпро : [б.в.], 2024. – Т. 2. – С. 300 – 304. – Режим доступу: https://cims.fti.dp.ua/j/article/view/131/159

Received 19.09.2024.
Accepted 25.09.2024.

Algorithms for data imputation based on entropy

Recent advancements in data imputation have focused on various machine learning techniques, including methods like mean, median, and mode imputation, along with more complex approaches like k-nearest neighbors (KNN) and multiple imputation by chained equations (MICE). Research into entropy-based methods offers a promising direction. This method minimizes uncertainty by selecting imputation values that reduce the overall entropy of the dataset.

The goal of this work is to develop an algorithm that imputes missing data by minimizing conditional entropy, thus ensuring that the missing values are filled in a way that preserves the relationships between the variables. The method is designed for both qualitative and quantitative data, including discrete and continuous variables, aiming to reduce uncertainty in classification tasks and enhance the performance of machine learning models.

The proposed algorithm is based on conditional entropy minimization, using entropy as a measure of uncertainty in data. For each incomplete row, the algorithm computes the conditional entropy for possible imputation values. The value that minimizes conditional entropy is selected, as it reduces uncertainty in the target variable. This process is iterated for each missing value until all missing data is imputed.

Three types of tests were performed on two datasets. The analysis showed that the proposed algorithms are quite slow compared to other methods and can be improved, for example, by multiprocessing, as described in our work [15]. The type 1 test showed that the proposed algorithms do not give a gain on the RMS deviation metric, but significantly reduce entropy (type 2 test). At the same time, these methods show an improvement in classification performance over the baseline models (type 3 test).

Thus, the proposed entropy-based imputation methods have shown good results and can be considered by researchers as an additional tool to improve the accuracy of decision making, but further computational optimisation studies are needed to improve the performance of these methods.

The algorithm shows promise in improving classification accuracy by selecting imputation values that minimize conditional entropy. Future research will focus on optimizing the method for large datasets and expanding its application to various domains.

Земляний Олексій Дмитрович – аспірант 3-го року навчання Дніпровського національного університету імені Олеся Гончара; 380638802605; zemlyanoy.aleksey@gmail.com.
Байбуз Олег Григорович – доктор технічних наук, професор, завідувач кафедри математичного забезпечення ЕОМ Дніпровського національного університету імені Олеся Гончара; тел. +380672828822; olegbaybuz68@gmail.com.

Zemlianyi Oleksii Dmytrovych – 3rd year postgraduate student at Oles Honchar Dnipro National University; 380638802605; zemlyanoy.aleksey@gmail.com.
Baibuz Oleh Grigorovich - – Doctor of Technical Sciences, Professor, Head of the Department of Mathematical Support for Computers at Oles Honchar Dnipro National University; tel. +380672828822; olegbaybuz68@gmail.com.