«Системні технології» 2 (151) 2024 «System technologies»
DOI 10.34185/1562-9945-2-151-2024-04
УДК 004.42:519.25

О.Д. Земляний, О.Г. Байбуз

МЕТОДИ ІМПУТУВАННЯ ПРОПУСКІВ У ДАНИХ ПРО ІШЕМІЧНУ ХВОРОБУ СЕРЦЯ

Анотація. Пропонуються декілька модифікацій алгоритмів ітеративного множинного імпутування пропусків у змішаних даних, що представлені кількісними і якісними ознаками. Серед кількісних є і неперервні, і дискретні. Серед якісних є порядкові та бінарні. Для аналізу підходів використовується два типи тестів. В першому тесті датасет з відомими даними штучно заповнюється пропусками у випадкових позиціях, проводиться імпутування різними методами, оцінюється середньо квадратична похибка та час виконання алгоритмів. В другому тесті навчають моделі бінарної класифікації на датасетах, з імпутацією пропусків різними методами, та порівнюють точність класифікації на тестовій вибірці. Для перетворення якісних ознак на кількісні запропоновано власні алгоритми, які працюють з пропущеними даними та дозволяють виконувати зворотне перетворення знов до якісних ознак. Розглядаються два відомих датасети про спостереження стосовно ішемічної хвороби серця.

Ключові слова: імпутування даних, ітеративне множинне імпутування даних, обробка змішаних даних, регресія, бінарна класифікація, перетворення якісних ознак на кількісні, мова програмування python.

Вступ. У сфері науки про дані дослідницький аналіз є важливим етапом. Цей аналіз дозволяє дослідити певні характеристики даних, виявити закономірності, аномалії, провести очищення від викидів та побудувати початкові моделі для подальшої роботи. На цьому етапі можна встановити вид розподілу, оцінити його основні параметри, виявити викиди, побудувати матрицю кореляції ознак і таке інше.

У попередньому аналізі досить серйозною проблемою є виявлення пропущених значень, і найскладнішим є те, що немає жодного універсального алгоритму. Для кожної конкретної задачі доводиться підбирати відомі методи, їх комбінації, модифікації, або зовсім нові підходи.

Більшість моделей машинного навчання не можуть обробляти пропущені значення, тому ми не можемо просто проігнорувати пропуски в даних. Проблему пропущених даних необхідно вирішувати під час попередньої обробки.

Найпростішим рішенням є видалення кожного спостереження, що містить пропущені значення. Таке рішення реалізовано у відомих бібліотеках мови програмування Python, таких як Numpy або Pandas. Але цей підхід є крайнім випадком, бо ми втрачаємо всю корисну інформацію, яка може бути важливою для аналізу даних.

Існують наступні основні стратегії імпутування пропущених даних:

1. Замінити пропущені значення середнім/медіаною чи модою.
2. Замінити найчастішим значенням, що зустрічається, або константою.
3. Імпутування даних за допомогою методу k найближчих сусідів (алгоритм kNN).
4. Множинне імпутування пропущених даних (алгоритм MICE).
5. Імпутування даних за допомогою глибокого навчання.

Ці методи застосовуються для вирішення проблеми відсутніх даних у наборах даних.

Постановка проблеми в загальному вигляді. Розглядаємо два популярних набори даних, що є у відкритому доступі на платформі Kaggle.com. Це збірний датасет UCI Heart Disease Data [1, 2], створений чотирма кардіологічними центрами Угорщини, Швейцарії та США, та набір даних Framingham Heart Study [3, 4], який отримано з активного кардіологічного дослідження мешканців Массачусетсу та Фремінгему.

В обох наборах даних наявні пропуски. Дані містять порядкові якісні та кількісні показники, причому процент якісних показників вище кількісних. В наборі даних UCI Heart Disease Data частина пропусків спричинена об'єднанням кількох датасетів, в яких не всі з представлених показників були в наявності. Цей датасет містить цільовий стовпець зі значеннями, що відповідають статусу захворювання на ішемічну хворобу серця, від 0, що відповідає відсутності хвороби, до 4 – критична стадія. В цьому дослідженні ми будемо розглядати набір даних лише з точки зору бінарної класифікації: є хвороба чи ні. Причина такого вибору в тому, що вихідний набір даних є незбалансованим, а при розгляді його лише як задачу бінарної класифікації ми отримуємо вже збалансований датасет. Для якісного аналізу кардіологічних даних потрібно виконати імпутування пропущених даних. Дана робота присвячена пошуку можливих підходів до імпутування даних в рамках заданих умов.

Аналіз останніх досліджень. Р.Дж.Літтл та Д.Б.Рубін в своїй роботі надали класифікацію методів обробки даних з пропусками [5]. Для імпутування

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

пропусків даних популярними є такі підходи: аналіз повних спостережень (listwise deletion); методи, які використовують доступну інформацію (pairwise deletion); підстановка середнього по вибірці (mean substitution); метод хот-дек (hot deck); регресійний аналіз (regression) [6, 7]; оцінка за допомогою максимізації правдоподібності (maximum likelihood estimation); підстановка за допомогою факторного аналізу (factor analysis substitution) [8]; модель множинного відновлення даних (MICE) [9 – 13]. В нашому попередньому дослідженні [14] запропоновано ідеї імпутування даних гідрологічного моніторингу в межах знайдених груп постів спостереження, де підтверджено збіг емпіричних функцій розподілу, або в межах груп постів, де підтверджено гіпотезу про однорідність, а також за рахунок побудови регресійних залежностей між ознаками. Такий підхід був доречним, коли в межах екземплярів є ряди спостережень, по яким можна шукати групи схожих об’єктів – постів, а також, коли наявні кореляційні зв’язки між ознаками.

Мета роботи. Метою роботи є розроблення методів імпутування пропусків у кардіологічних даних з переважною більшістю якісних ознак на основі методів класифікації та регресії.

Основна частина. Почнемо розгляд з опису датасетів.

Датасет UCI Heart Disease Data містить 920 рядків даних, 15 ознак та один цільовий стовпець. Стовпці id та назва датасету ми виключаємо як неінформативні. Серед 13 ознак, що залишаються, 5 ознак – кількісні, 8 ознак – якісні порядкові, серед яких 2 – бінарні.

Первинний аналіз даних показав наявність пропусків. Серед 920 екземплярів лише 299 є повними.

Датасет Framingham Heart Study містить 4240 рядків даних, 15 ознак та один цільовий стовпець. Стовпець з ознакою куріння виявився неінформативним, тому ми його виключаємо. Серед 14 ознак, що залишаються, 8 ознак – кількісні, 6 ознак – якісні порядкові, серед яких 5 – бінарні.

Первинний аналіз даних показав наявність пропусків. Серед 4240 екземплярів 3658 є повними. Датасет є незбалансованим, кількість екземплярів одного класу в 5 разів перевищує кількість екземплярів другого класу.

Перетворення якісних ознак у кількісні. Популярним підходом для кодування якісних порядкових ознак є Label Encoding [15]. Цей метод виконує кодування унікальних якісних ознак у числа 0, 1, 2, …, k – 1, де k – кількість унікальних значень ознаки. Він реалізований в модулі

sklearn.preprocessing бібліотеки scikit-learn на мові програмування Python [16]. У цього методу є дві головні проблеми: він кодує пропущене значення як окреме; він не враховує частоту конкретного значення. В результаті кодування пропущених значень втрачається інформація про пропуски. Тому прийнято рішення запропонувати два власних алгоритми, які не кодують пропущені дані. Перший le_non_missing – це модифікація Label Encoding, який лише пропускає пропущені дані, а для інших застосовує Label Encoding (лістинг 1), другий le_non_missing_frequent – враховує частоту значення. Цей алгоритм дозволяє надавати максимальний номер значенню з найменшою частотою або з найбільшою в залежності від параметра ascending (лістинг 2). Обидва рішення мають декодер для зворотного перетворення.

Лістинг 1. Алгоритм le_non_missing

```
def label_encoding_non_missing_values(df):
    df1 = df.copy()
    encoders = dict()
    for col_name in df1.select_dtypes(
                include=['object', 'category']).columns:
        series = df1[col_name]
        label_encoder = LabelEncoder()
        df1[col_name] = pd.Series(
                        label_encoder.fit_transform(
                         series[series.notnull()]),
                   index=series[series.notnull()].index
        )
        encoders[col_name] = label_encoder
    return df1, encoders
def label_decoding_non_missing_values(df, encoders):
    df1 = df.copy()
    for col_name in encoders:
        enc = encoders[col_name]
        nmap = dict(zip(enc.transform(enc.classes_),
                        enc.classes_))
        df1[col_name] = df1[col_name]
        .apply(lambda x: x if np.isnan(x) else nmap[x])
    return df1
```

Лістинг 2. Алгоритм le_non_missing_frequent

```
def label_encoding_non_missing_values_frequent(df,
                                               ascending=False):
    df1 = df.copy()
```


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

```
    encoders = dict()
    for col_name in df1.select_dtypes(
                include=['object', 'category']).columns:
        d = df[col_name].value_counts().sort_values(
                            ascending=ascending).to_dict()
        k = 0
        for key in d:
            d[key] = k
            k = k + 1
        encoders[col_name] = d
        df1[col_name] = df1[col_name]
            .apply(lambda x: x if x != x else d[x])
    return df1, encoders
def label_decoding_non_missing_values_frequent(df,
                                                encoders):
    df1 = df.copy()
    for col_name in encoders:
        d = encoders[col_name]
        nd = dict()
        for key, value in d.items():
            nd[value] = key
                  df1[col_name] = df1[col_name]
                       .apply(lambda  x:  x  if  x  !=  x  else
nd[x])
    return df1
```

Особливістю запропонованих алгоритмів є формування словнику для кожної якісної ознаки для подальшого зворотного перетворення та пропуск пропущеного значення. В подальшому ці методи будуть переписані на класи в архітектурі scikit-learn для універсального застосування. Слід врахувати те, що алгоритми спрацюють тільки для стовпців, типи яких відносяться до якісних (object та category). Наприклад, датасет Framingham Heart Study містить вже закодовані ознаки і це не те, що нам потрібно. Тому для аналізу спочатку фактично якісні ознаки датасету було перекодовано на їх первинний стан, що містить дійсно якісні ознаки згідно словнику, наданому в описі даних.

Запропоновані алгоритми дають можливість одразу перетворити всі якісні ознаки на кількісні без втрати інформації про пропуски.

Імпутування даних. Розглянемо ідею імпутування даних як задачу прогнозу значення в стовпці, що містить пропуск, на основі даних зі всіх інших

стовпців, враховуючи цільовий стовпець. Якщо цільова ознака є функцією від інших ознак:

$$y_i = f\left(x_{i,1}, x_{i,2}, \dots, x_{i,k}, \dots, x_{i,p}\right) + \varepsilon ,$$

то можна очікувати, що існує також стохастична залежність *k*-ї ознаки від інших ознак, враховуючи цільову:

$$x_i = \varphi\left(x_{i,1}, x_{i,2}, \dots, x_{i,k-1}, x_{i,k+1}, \dots, x_{i,p}, y_i\right) + \varepsilon .$$

Ця ідея певною мірою лежить в основі методів MICE [10, 11] та NoNa [13].

Запропоновано 4 власні реалізації такої ідеї з кількома модифікаціями. Ми будемо виходити з можливої наявності таких патернів у пропусках даних:

a) в даних немає жодного рядка чи стовпця з повними даними;

b) в даних є *k* стовпців з повністю наявними даними у всіх екземплярах;

c) в даних є *n* рядків з повністю наявними даними за всіма ознаками.

Ми також будемо враховувати, що серед ознак можуть бути присутні як кількісні, так і якісні ознаки (вже закодовані).

Запропоновані такі алгоритми:

Алгоритм 1. Метод *fillna_k_columns* – враховує патерни a), b), застосовує регресор чи класифікатор в залежності від типу ознаки.

Алгоритм 2. Метод *fillna_k_sorted_columns* – враховує патерни a), b), застосовує регресор чи класифікатор в залежності від типу ознаки, обробляє стовпці в залежності від кількості пропусків у них.

Алгоритм 3. Метод *fillna_2steps_rg_class* – враховує патерни a), b), c), застосовує регресор чи класифікатор в залежності від типу ознаки, обробляє стовпці в залежності від кількості пропусків у них.

Алгоритм 4. Метод *fillna_2steps_rg* – враховує патерни a), b), c), застосовує тільки регресор з коригуванням значення для якісних ознак, обробляє стовпці в залежності від кількості пропусків у них.

Далі наведемо ці алгоритми.

Алгоритм 1. Метод *fillna_k_columns*

1. Цикл по стовпцям датасету *for j in columns*:

2. Якщо в *j*-му стовпці є пропущені дані, переходимо на крок 3, інакше – на наступну ітерацію, крок 1.

3. Знаходимо індекси рядків, де є пропуски в *j*-му стовпці, та зберігаємо до масиву *indexes_nan*.

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

4. Знаходимо підмножину *full_data* датасету, де в стовпцях немає жодного пропуску. Це буде наш датасет, за рахунок якого ми будемо прогнозувати пропущені значення в стовпці *j*.

5. Якщо *full_data* пустий, заповнюємо пропускі на основі стандартного методу: кількісну ознаку заповнюємо медіаною, якісну – модою. Якщо *full_data* непустий, переходимо на крок 6.

6. Розділяємо *full_data* на тренувальну та тестову вибірки за такою ознакою: в тренувальній вибірці залишаємо всі рядки, де в *j*-му стовпці є дані, тобто всі рядки, окрім зазначених в *indexes_nan*.

7. Якщо *j*-й стовпець містить якісні дані, використовуємо класифікатор, інакше – регресор.

8. Навчаємо модель на тренувальній вибірці. Отримаємо прогнозні значення для *j*-го стовпця за даними тестової вибірки.

9. Переходимо на крок 1.

Наступний алгоритм є модифікацією попереднього, де надається можливість спочатку імпутувати пропущені дані у стовпцях з найменшою кількістю пропусків, або навпаки – з найбільшою.

Алгоритм 2. Метод *fillna_k_sorted_columns*

1. Сортуємо масив колонок *columns* за кількістю пропусків в них. В залежності від параметру *ascending* будемо виконувати обхід стовпців за зростанням кількості пропусків або за спаданням.

2. Цикл по стовпцям датасету *for j in columns*:

3. Якщо в *j*-му стовпці є пропущені дані, переходимо на крок 3, інакше – на наступну ітерацію, крок 1.

4. Знаходимо індекси рядків, де є пропуски в *j*-му стовпці, та зберігаємо до масиву *indexes_nan*.

5. Знаходимо підмножину *full_data* датасету, де в стовпцях немає жодного пропуску. Це буде наш датасет, за рахунок якого ми будемо прогнозувати пропущені значення в стовпці *j*.

6. Якщо *full_data* пустий, заповнюємо пропускі на основі стандартного методу: кількісну ознаку заповнюємо медіаною, якісну – модою. Якщо *full_data* непустий, переходимо на крок 7.

7. Розділяємо *full_data* на тренувальну та тестову вибірки за такою ознакою: в тренувальній вибірці залишаємо всі рядки, де в *j*-му стовпці є дані, тобто всі рядки, окрім зазначених в *indexes_nan*.

8. Якщо *j*-й стовпець містить якісні дані, використовуємо класифікатор, інакше – регресор.

9. Навчаємо модель на тренувальній вибірці. Отримаємо прогнозні значення для *j*-го стовпця за даними тестової вибірки.

10. Переходимо на крок 1.

Наступний алгоритм є двох-етапним. На першому етапі шукаємо підмножину датасету, що складається з рядків, де є повністю всі дані за всіма ознаками. Доєднуємо до нього ті рядки, де є лише один пропуск. Виконуємо імпутування даних в цих додаткових рядках за аналогією попередніх алгоритмів. На другому етапі повторюємо дії, описані в алгоритмі *fillna_k_sorted_columns*.

Алгоритм 3. Метод *fillna_2steps_rg_class*

Етап 1.

1. Формуємо масив ознак *cols_1nan*, для рядків, в яких є лише один пропуск саме в цих стовпцях.

2. Якщо такий масив пустий, переходимо на крок 11.

3. Цикл по стовпцям датасету *for j in cols_1nan*:

4. Знаходимо індекси рядків, де є тільки один пропуск, і він знаходиться в *j*-му стовпці, та зберігаємо до масиву *indexes_nan*.

5. Знаходимо підмножину *full_data* датасету, де в рядках немає жодного пропуску. До нього приєднуємо рядки *indexes_nan*. Це буде наш датасет, за рахунок якого ми будемо прогнозувати пропущені значення в стовпці *j* рядків *indexes_nan*.

6. Якщо *full_data* пустий, переходимо достроково до наступної ітерації, тобто на крок 3.

7. Розділяємо *full_data* на тренувальну та тестову вибірки за такою ознакою: в тренувальній вибірці залишаємо всі рядки, де в *j*-му стовпці є дані, тобто всі рядки, окрім зазначених в *indexes_nan*.

8. Якщо *j*-й стовпець містить якісні дані, використовуємо класифікатор, інакше – регресор.

9. Навчаємо модель на тренувальній вибірці. Отримаємо прогнозні значення для *j*-го стовпця за даними тестової вибірки.

10. Переходимо на крок 3.

Етап 2.

11. Викликаємо алгоритм *fillna_k_sorted_columns*.

Останній алгоритм використовує в будь-якому разі регресор, але у випадку якісної ознаки виконує коригування отриманого значення одним з двох спо-

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

собів: обирається звичайне найближче значення зі словника за цією ознакою (*measure = 'value'*) або найближче за середньо зваженою оцінкою з урахуванням частоти значень (*measure = 'weight'*).

Алгоритм 4. Метод *fillna_2steps_rg*

Етап 1.

1. Формуємо масив ознак *cols_1nan*, для рядків, в яких є лише один пропуск саме в цих стовпцях.

2. Якщо такий масив пустий, переходимо на крок 11.

3. Цикл по стовпцям датасету *for j in cols_1nan*:

4. Знаходимо індекси рядків, де є тільки один пропуск, і він знаходиться в *j*-му стовпці, та зберігаємо до масиву *indexes_nan*.

5. Знаходимо підмножину *full_data* датасету, де в рядках немає жодного пропуску. До нього приєднуємо рядки *indexes_nan*. Це буде наш датасет, за рахунок якого ми будемо прогнозувати пропущені значення в *j*-му стовпці рядків *indexes_nan*.

6. Якщо *full_data* пустий, переходимо достроково до наступної ітерації, тобто на крок 3.

7. Розділяємо *full_data* на тренувальну та тестову вибірки за такою ознакою: в тренувальній вибірці залишаємо всі рядки, де в *j*-му стовпці є дані, тобто всі рядки, окрім зазначених в *indexes_nan*.

8. В будь-якому випадку використовуємо регресор.

9. Навчаємо модель на тренувальній вибірці. Отримаємо прогнозні значення для *j*-го стовпця за даними тестової вибірки.

10. Якщо *j*-й стовпець містить якісні дані, виконуємо коригування отриманого значення за одним з двох алгоритмів (*value* або *weight*).

11. Переходимо на крок 3.

Етап 2.

12. Сортуємо масив колонок *columns* за кількістю пропусків в них. В залежності від параметру *ascending* будемо виконувати обхід стовпців по зростанню кількості пропусків або по спаданню.

13. Цикл по стовпцям датасету *for j in columns*:

14. Якщо в *j*-му стовпці є пропущені дані, переходимо на крок 15, інакше – достроково на наступну ітерацію, тобто на крок 13.

15. Знаходимо індекси рядків, де є пропуски в *j*-му стовпці, та зберігаємо до масиву *indexes_nan*.

16. Знаходимо підмножину *full_data* датасету, де в стовпцях немає жодного пропуску. Це буде наш датасет, за рахунок якого ми будемо прогнозувати пропущені значення в стовпці *j*.

17. Якщо *full_data* пустий, заповнюємо пропускі на основі стандартного методу: кількісну ознаку заповнюємо медіаною, якісну – модою. Якщо *full_data* непустий, переходимо на крок 18.

18. Розділяємо *full_data* на тренувальну та тестову вибірки за такою ознакою: в тренувальній вибірці залишаємо всі рядки, де в *j*-му стовпці є дані, тобто всі рядки, окрім зазначених в *indexes_nan*.

19. В будь-якому випадку використовуємо регресор.

20. Навчаємо модель на тренувальній вибірці. Отримаємо прогнозні значення для *j*-го стовпця за даними тестової вибірки.

21. Якщо *j*-й стовпець містить якісні дані, виконуємо коригування отриманого значення за одним з двох алгоритмів (*value* або *weight*).

22. Переходимо на крок 13.

Аналіз результатів застосування методів. Для аналізу запропонованих підходів використовується два типи тестів. В першому тесті в датасет штучно вносяться пропуски, далі проводиться імпутування пропущених значень різними методами, оцінюється середньо-квадратична похибка та час виконання алгоритмів. Розглядаються 10% пропусків, 20%, 30% та 40%. В другому 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). Датасет Framingham Heart Study попередньо балансується методами SMOTE [17]. В таблиці 1 наведено результати тесту 1 для датасетів UCI Heart Disease Data та Framingham Heart Study, в таблиці 2 – результати тесту 2 відповідно. Виконуємо тести для таких ситуацій:

1. Алгоритм *le_non_missing, SimpleImputer* – стандартне заповнення значенням з найбільшою частотою

2a. Алгоритм *le_non_missing, IterativeImputer(sample_posterior = False)*

2b. Алгоритм *le_non_missing, IterativeImputer(sample_posterior = True)*

3. Алгоритм *le_non_missing, kNNImputer*

4. Алгоритм *le_non_missing, nonaImputer*

5. Алгоритм *le_non_missing, fillna_k_columns*

6a. Алгоритм *le_non_missing, fillna_k_sorted_columns(ascending = True)*

6b. Алгоритм *le_non_missing, fillna_k_sorted_columns(ascending = False)*

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

7. Алгоритм *le_non_missing, fillna_2steps_rg_class*

8a. Алгоритм *le_non_missing_frequent(ascending=False), fillna_2steps_rg_class*

8b. Алгоритм *le_non_missing_frequent(ascending=True), fillna_2steps_rg_class*

9a. Алгоритм *le_non_missing_frequent(ascending=False), fillna_2steps_rg(measure = 'value')*

9b. Алгоритм *le_non_missing_frequent(ascending=False), fillna_2steps_rg(measure = 'weight')*

Таблиця 1

Результати тесту 1

	Датасет UCI HDD					**Датасет Framingham FHS**				
	10%	20%	30%	40%	Час, с	10%	20%	30%	40%	Час, с
1	3.23	3.9	5.48	5.73	0.037210	2.75	4.32	5.41	6.32	0.054037
2	**2.42**	**3.35**	4.72	4.82	0.642131	**2.16**	**3.24**	4.89	5.5	3.245387
3	2.63	3.59	4.44	4.82	0.614188	2.44	3.82	4.86	5.54	9.091372
4	2.49	3.51	4.31	4.66	4.917326	2.33	3.43	4.44	5.12	4.973262
5	2.49	3.51	**4.31**	**4.66**	4.990149	2.33	3.43	**4.44**	**5.12**	5.004628
6a	2.44	3.45	4.32	4.68	4.468355	2.34	3.5	4.49	5.26	5.805581
6b	2.55	3.51	4.35	4.75	4.465862	2.4	3.63	4.65	5.24	5.847364
7	2.55	3.51	4.35	4.75	7.567574	2.4	3.63	4.65	5.24	9.386514
8a	2.56	3.82	4.35	4.68	7.706980	2.37	3.62	4.64	5.24	9.479322
8b	2.56	3.65	4.35	4.65	7.583769	2.37	3.62	4.64	5.24	9.406302
9a	2.55	3.77	4.37	4.71	0.605775	2.36	3.61	4.63	5.24	**1.187455**
9b	2.55	3.76	4.35	4.73	**0.596536**	2.36	3.61	4.63	5.24	1.241692

Жирним шрифтом виділено переможців по кожному рядку за точністю та часом виконання. Запропоновані методі показують приблизно однакову точність, однак останній метод *fillna_2steps_rg* дає значний виграш у часі виконання. І якщо порівняти запропоновані методи з результатами методу *IterativeImputer* з бібліотеки *scikit-learn*, то слід зауважити, що цей метод використовує регресор без корегування значень якісних даних, а значить, він не на-

дає можливості декодувати отримані результати для подальшого використання. На відміну від нього будь-який з запропонованих чотирьох алгоритмів отримує значення з потрібної множини для кожної якісної ознаки, а тому допускає декодування та подальше використання.

Далі розглянемо тест 2. Додамо в таблицю ще нульовий рядок з базовим результатом – класифікація за датасетами, з яких видалено рядки з пропусками. В таблиці 2 жирним шрифтом виділено переможців за точністю.

Таблиця 2

Результати тесту 2

	Датасет UCI HDD		**Датасет Framingham FHS**	
	Найкращі гіперпараметри моделі	Точність	Найкращі гіперпараметри моделі	Точність
0	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 50}	0.83	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	0.89
1	{'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}	0.80	{'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}	0.89
2a	{'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 150}	0.88	{'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	**0.90**
2b	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}	0.82	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	0.88
3	{'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 50}	0.82	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}	**0.90**
4	{'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split':	0.89	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split':	0.89

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

	10, 'n_estimators': 50}		2, 'n_estimators': 150}	
5	{'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 50}	0.89	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	0.89
6a	{'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 100}	0.88	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	0.88
6b	{'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 50}	0.91	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	0.87
7	{'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}	0.91	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	0.88
8a	**{'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 50}**	**0.92**	**{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}**	**0.90**
8b	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}	0.90	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	0.88
9a	{'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 150}	0.90	{'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}	0.88
9b	{'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}**	**0.90**

З аналізу результатів випливає, що запропоновані методи дозволяють покращити точність класифікації, а на датасеті UCI Heart Disease Data навіть суттєво покращити порівняно з 83% до 92%. Серед переваг запропонованих методів слід віднести можливість декодування імпутованих даних та виграш в часі для останнього методу імпутування.

Висновки та перспективи подальших досліджень. У даній роботі запропоновано два алгоритми кодування та декодування якісних порядкових ознак та чотири алгоритми імпутування пропусків даних:

1. Алгоритм кодування / декодування *le_non_missing*, який дозволяє працювати з даними, що містять пропуски та не втратити інформацію про пропуски.

2. Алгоритм кодування / декодування *le_non_missing_frequent*, який дозволяє працювати з даними, що містять пропуски та не втратити інформацію про пропуски, і враховує частоту значень.

3. Метод *fillna_k_columns*, який виконує імпутування пропущених даних за *k* повними стовпцями. Використовується регресор або класифікатор в залежності від типу стовпця.

4. Метод *fillna_k_sorted_columns*, який виконує обхід стовпців у порядку, що відповідає кількості пропущених значень. Використовується регресор або класифікатор в залежності від типу стовпця.

5. Метод *fillna_2steps_rg_class*, який виконується у 2 етапи: спочатку за повними рядками, а потім за повними стовпцями. Використовується регресор або класифікатор в залежності від типу стовпця.

6. Метод *fillna_2steps_rg*, який виконується у 2 етапи: спочатку за повними рядками, а потім за повними стовпцями. Використовується тільки регресор з коригуванням значення для якісних стовпців за двома критеріями.

7. Проведено два типи тестів на двох наборах даних. Аналіз показав виграш у часі для методу *fillna_2steps_rg* та покращення точності моделі класифікації у випадках використання методу кодування з урахуванням частоти та методу імпутування *fillna_2steps_rg_class*.

Таким чином, запропоновані методи показали гарні результаті, які можуть стати альтернативами існуючих методів та надати досліднику додатковий інструментарій для підвищення точності прийняття рішень.

Надалі планується оформити запропоновані методи в архітектурі бібліотеки *scikit-learn* для уніфікованого використання дослідниками.

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

ЛІТЕРАТУРА / REFERENCE

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

2. 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

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

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

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

6. Sefidian A. M., Daneshpour N. Estimating missing data using novel correlation maximization based methods // Applied Soft Computing. Volume 91. 2020. 106249. DOI: 10.1016/j.asoc.2020.106249

7. A.Barrios, G. Trincado, René Garreaud. Alternative approaches for estimating missing climate data: application to monthly precipitation records in South-Central Chile // Forest Ecosystems. 2018. 5(1). Pp. 1-10. DOI 10.1186/s40663-018-0147-x

8. Kamakura W.A., Wedel M. Factor Analysis and Missing Data // Journal of Marketing Research. 2000. Vol. 37. No. 4: Nov. P. 490–498.

9. van Buuren, S. and Groothuis-Oudshoorn, K. 2011. mice: Multivariate Imputation by Chained Equations in R. Journal of Statistical Software. 45, 3 (Dec. 2011), 1–67. DOI:https://doi.org/10.18637/jss.v045.i03.

10. A complete guide on how to handle missing data with IterativeImputer in Python. – Learning AI, 2023. – Режим доступу: https://justlearnai.com/a-complete-guide-on-how-to-handle-missing-data-with-iterativeimputer-in-python-6b224cf0896c

11. Imputation of missing values in scikit-learn 1.4.1. – [Електронний ресурс]. – Режим доступу: https://scikit-learn.org/stable/ modules/impute.html

12. NoNa: Missing Data Imputation Algorithm. – Medium, 2023. – [Електронний ресурс]. – Режим доступу: https://medium.com/@abdualimov/nona-missing-data-imputation-algorithm-d6ff92f70ab8

13. nona: Python gap filling toolkit. [Електронний ресурс]. – Режим доступу: https://pypi.org/project/nona/

14. Земляний О.Д., Ізмайлова М.К., Антоненко С.В. Методи поповнення пропусків даних гідрологічного моніторингу // Актуальні проблеми автоматизації та

інформаційних технологій: Зб. наук. пр. / наук. ред. О.Г. Байбуз. – Дніпро, 2020. – Т. 24. – С. 3 – 15

15. Feature Encoding. – Medium, 2023. – [Електронний ресурс]. – Режим доступу: https://medium.com/@denizgunay/feature-encoding-f099a6c1abe8

16. sklearn.preprocessing.LabelEncoder– [Електронний ресурс]. – Режим доступу: https://scikit-learn.org/stable/modules/generated/sklearn.
preprocessing.LabelEncoder.html

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

Received 27.02.2024.
Accepted 29.02.2024.

Methods for imputing missing data on coronary heart disease

Preliminary analysis is an important stage of data analysis. A significant problem is the detection of missing values, and the most difficult part is that there is no universal algorithm to resolve this problem. For each specific task, known methods, their combinations, modifications, or completely new approaches have to be selected.

Most machine learning models cannot handle missing values, so we cannot simply ignore gaps in the data. The problem of missing data needs to be addressed during preprocessing. The simplest solution is to delete each observation containing missing values. This solution is implemented in well-known Python programming language libraries such as NumPy or Pandas. However, this approach is extreme because we lose all the useful information that may be important for data analysis. There are several main strategies for imputing missing data: replacing missing values with mean/median or mode; replacing with the most frequently occurring value or a constant; data imputation using the kNN algorithm; multiple imputation of missing data (MICE algorithm); data imputation using deep learning.

We suppose several modifications of algorithms for iterative multiple imputing of mixed data represented by quantitative and qualitative features. To convert qualitative features into numerical ones, we propose our own algorithms that work with missing data and allow for the conversion back to qualitative features. Two well-known datasets on observations of coronary heart disease are considered.

The following is a brief description of the data imputation algorithms. The fillna_k_columns method, which performs data imputation based on k complete columns. It uses a regressor or classifier depending on the column type. The fillna_k_sorted_columns method, which traverses columns in the order corresponding to the number of missing values. It uses a regressor or classifier depending on the column type. The fillna_2steps_rg_class method, which is executed in 2 steps: first by complete rows, then by complete columns. It uses a regressor or classifier depending on the column

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

type. The fillna_2steps_rg method, which is executed in 2 steps: first by complete rows, then by complete columns. It only uses a regressor with value adjustment for qualitative columns based on two criteria.

Two types of tests are used to analyse the approaches. In the first test, a dataset is artificially filled with gaps at random positions, imputed using different methods, and the mean square error and execution time of the algorithms are estimated. In the second test, binary classification models are trained on datasets imputed with different methods and the classification accuracy is compared. The analysis showed a time advantage for the fillna_2steps_rg method and improved classification model accuracy in cases of using encoding method considering frequency and the fillna_2steps_rg_class imputation method.

Thus, the proposed methods have shown promising results, which can serve as alternatives to existing methods and provide researchers with additional tools to enhance decision-making accuracy.

Further, the plan is to formalize the proposed methods in the scikit-learn library architecture for unified use by researchers.

Keywords: data imputation, iterative multiple data imputation, mixed data processing, regression, binary classification, transformation of qualitative features into quantitative ones, python.

Земляний Олексій Дмитрович – аспірант 1-го року навчання Дніпровського національного університету імені Олеся Гончара.

Zemlianyi Oleksii - postgraduate student of the 1st year of study at the Dnipro National University named after Oles Honchar.

Байбуз Олег Григорович - доктор технічних наук, професор, завідувач кафедри математичного забезпечення ЕОМ Дніпровського національного університету імені Олеся Гончара.

Baibuz Oleh - doctor of technical sciences, professor, head of the department of computer mathematical support of Dnipro National University named after Oles Honchar.