УДК 004.423
DOI 10.51691/2541-8327_2023_6_11

ОБРАБОТКА ПРОПУЩЕННЫХ ЗНАЧЕНИЙ И ДУБЛИКАТОВ В ДАННЫХ С ПОМОЩЬЮ БИБЛИОТЕКИ PANDAS ДЛЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

Дрянкова Д.А.
студент факультета информатики и вычислительной техники,
Хакасский государственный университет имени Н.Ф. Катанова,
г. Абакан, Россия[1]

Аннотация: Целью данного исследования было исследование и применение методов обработки пропущенных значений и дубликатов в данных с использованием библиотеки Pandas для языка программирования Python. Основная идея заключалась в изучении различных подходов к обработке этих проблемных ситуаций и оценке их эффективности.

В ходе исследования были рассмотрены различные методы обработки пропущенных значений, такие как удаление строк или столбцов с пропущенными значениями, заполнение пропусков средними или медианами, а также использование интерполяции для заполнения пропущенных значений. Также были исследованы методы обнаружения и удаления дубликатов, а также методы для обнаружения частичных дубликатов на основе определенных столбцов или условий.

Результаты исследования показали, что библиотека Pandas предоставляет широкий набор инструментов для обработки пропущенных значений и дубликатов, позволяющих эффективно очищать данные перед анализом.

[1] Научный руководитель: Замулин И.С. заведующий кафедрой ПОВТиАС, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия

Ключевые слова: Python, Pandas, таблицы данных, таблицы, строки, дубликаты, пропущенные значения, обработка.

PROCESSING OF MISSING VALUES AND DUPLICATES IN DATA USING THE PANDAS LIBRARY FOR THE PYTHON PROGRAMMING LANGUAGE

Dryakova D.A.
student of the Faculty of Computer Science and Computer Engineering,
N.F. Katanov Khakass State University,
Abakan, Russia

Abstract: The purpose of this study was to investigate and apply methods for processing missing values and duplicates in data using the Pandas library for the Python programming language. The main idea was to study different approaches to handling these problematic situations and evaluate their effectiveness.
The study examined various methods of processing missing values, such as deleting rows or columns with missing values, filling in the gaps with averages or medians, and using interpolation to fill in the missing values.
Methods for detecting and removing duplicates have also been investigated, as well as methods for detecting partial duplicates based on certain columns or conditions.
The results of the study showed that the Pandas library provides a wide range of tools for processing missing values and duplicates, allowing you to effectively clean up data before analysis.
Keywords: Python, Pandas, data tables, tables, rows, duplicates, missing values, processing.

Библиотека Pandas предоставляет мощные инструменты для обработки пропущенных значений в таблицах данных. Это включает заполнение пропущенных значений, удаление строк или столбцов с пропущенными

значениями, а также интерполяцию пропущенных значений на основе соседних значений [1].

Обработка пропущенных значений (missing values) является важным шагом в работе с данными. Pandas предоставляет несколько методов для работы с пропущенными значениями, в том числе:

1) Поиск пропущенных значений - методы *isnull()* и *notnull()*
2) Удаление пропущенных значений - метод *dropna()*
3) Заполнение пропущенных значений - метод *fillna()*

Далее представлен код с таблицей данных и примерами обработки пропущенных значений (Рис. 1). После представлено содержимое таблицы данных *df* (Рис. 2).

```
>>> import pandas as pd
>>> import numpy as np
>>>
>>> # Создание таблицы с пропущенными значениями
... df = pd.DataFrame({'A': [1, 2, np.nan, 4],
...                    'B': [5, np.nan, 7, 8],
...                    'C': [9, 10, 11, np.nan]})
...
>>> # Поиск пропущенных значений
... df_mn_1 = df.isnull()
...
>>> # Поиск обратных значений
... df_mn_2 = df.notnull()
...
>>> # Удаление строк с пропущенными значениями
... df_del_mn = df.dropna()
...
>>> # Заполнение пропущенных значений средним значением по столбцу
... df_fill_av = df.fillna(df.mean())
```

Рисунок 1 – Таблица данных и методы обработки пропущенных значений [разработано автором]

```
>>> df
     A    B     C
0  1.0  5.0   9.0
1  2.0  NaN  10.0
2  NaN  7.0  11.0
3  4.0  8.0   NaN
```

Рисунок 2 – Содержимое таблицы данных *df* [разработано автором]

Метод *isnull()* позволяет найти все пропущенные значения в таблице и вернуть таблицу с булевыми значениями *True/False* вместо пропущенных значений. Результат выполнения данного метода (Рис. 3).

```
>>> df_mn_1
       A      B      C
0  False  False  False
1  False   True  False
2   True  False  False
3  False  False   True
```

Рисунок 3 – Результат выполнения метода *isnull()* [разработано автором]

Метод *notnull()* возвращает булеву таблицу с обратными значениями. Результат выполнения данного метода (Рис. 4).

```
>>> df_mn_2
       A      B      C
0   True   True   True
1   True  False   True
2  False   True   True
3   True   True  False
```

Рисунок 4 – Результат выполнения метода *notnull()* [разработано автором]

Метод *dropna()* удаляет все строки, содержащие хотя бы одно пропущенное значение. Этот метод может быть полезен, если пропущенные значения составляют небольшую долю данных в таблице и их можно удалить без серьезного искажения результатов. Результат выполнения данного метода (Рис. 5).

```
>>> df_del_mn
     A    B    C
0  1.0  5.0  9.0
```

Рисунок 5 - Результат выполнения метода *dropna()* [разработано автором]

Метод *fillna()* заполняет пропущенные значения определенным значением, таким как среднее, медианное или наиболее часто встречающееся значение в столбце. Результат выполнения данного метода (Рис. 6).

```
>>> df_fill_av
          A         B     C
0  1.000000  5.000000   9.0
1  2.000000  6.666667  10.0
2  2.333333  7.000000  11.0
3  4.000000  8.000000  10.0
```

Рисунок 6 – Результат выполнения метода *fillna()* [разработано автором]

Таким образом, обработка пропущенных значений является важным шагом в работе с данными, и библиотека Pandas предоставляет несколько методов для работы с пропущенными значениями в таблицах данных.

Также Pandas предоставляет инструменты для обнаружения и удаления дубликатов в таблицах данных. Это позволяет избежать ошибок при обработке данных и улучшить качество анализа [2].

В библиотеке Pandas есть несколько способов работы с дубликатами, один из них - это метод *duplicated()*, который позволяет определить, есть ли в таблице дубликаты и сколько их. Этот метод возвращает булев массив, где *True* указывает на дубликаты, а *False* - на уникальные значения.

Код с таблицей данных, содержащей повторяющиеся значения и результат выполнения метода *duplicated()* (Рис. 7).

```
>>> import pandas as pd

>>> df = pd.DataFrame({'A': ['foo', 'bar', 'foo', 'bar', 'foo', 'bar', 'foo', 'foo'],
...                    'B': ['one', 'one', 'two', 'two', 'two', 'one', 'one', 'two'],
...                    'C': [1, 2, 1, 2, 1, 1, 2, 1]})
>>>
>>> df.duplicated()
0    False
1    False
2    False
3    False
4     True
5    False
6    False
7     True
dtype: bool
```

Рисунок 7 – Таблица данных и результат работы метода *duplicated()*

[разработано автором]

Другой способ обработки дубликатов - это метод *drop_duplicates()*, который удаляет все дубликаты из таблицы. Этот метод имеет несколько параметров, таких как *keep*, который указывает, какой из дубликатов нужно сохранить (первый или последний), и *subset*, который позволяет указать столбцы, в которых нужно искать дубликаты. Далее представлена таблица данных с дубликатами, а также использование метода *drop_duplicates()* (Рис. 8).

```
>>> import pandas as pd

>>> # создание таблицы с дубликатами
... data = {'name': ['John', 'Sarah', 'John', 'Sam', 'Sarah'],
...         'age': [32, 28, 32, 25, 28],
...         'city': ['New York', 'Los Angeles', 'New York', 'San Francisco',
    'Los Angeles']}
>>>
>>> df = pd.DataFrame(data)
>>>
>>> # удаление дубликатов
... df.drop_duplicates(subset=['name', 'age'], keep='first', inplace=True)
```

Рисунок 8 – Использование метода *drop_duplicates()* на таблице данных [разработано автором]

В данном примере будут удалены строки с дублирующимися значениями в столбцах *'name'* и *'age'*, оставив только первое вхождение. Метод *drop_duplicates()* также имеет параметр *keep*, который определяет, какой из дубликатов оставить (первый, последний или все).

В результате таблица примет следующий вид (Рис. 9).

```
>>> df
    name  age           city
0   John   32       New York
1  Sarah   28    Los Angeles
3    Sam   25  San Francisco
```

Рисунок 9 – Результат выполнения метода *drop_duplicates()* [разработано автором]

Обработка дубликатов и пропущенных значений важна для обеспечения точности анализа данных и избегания ошибок в выводах. Библиотека Pandas предоставляет мощные инструменты для обработки таких случаев, которые могут быть легко применены к большому количеству данных [3].

Заключение

Подводя итоги, можно убедится в том, что библиотека Pandas предоставляет удобные и мощные инструменты как для обработки дубликатов, так и для обработки пропущенных значений, с помощью данной библиотеки есть возможность манипулировать как с небольшим объёмом данных, так и с более внушительными базами данных.

Библиографический список

1. Никола Лейси. Python, например [Текст] / Никола Лейси – СПб.: Питер, 2021 г. – 192 с.
2. Алексей Васильев, Программирование на Python в примерах и задачах [Текст] / Алексей Васильев – М.: Бомбора, 2021 г. – 616 с.
3. В. Ф. Очков, К. А. Орлов, Ю. В. Чудова. Информационные технологии в инженерных расчетах. SMath и Python. Учебное пособие [Текст] / В. Ф. Очков, К. А. Орлов, Ю. В. Чудова. – СПб.: Лань, 2023 г. – 212 с.

Оригинальность 76%