

Лекція 6.

Оброблення відсутніх даних. Перетворення типів даних та маніпулювання дата фреймами у Python

План лекції

- 6.1. Оброблення відсутніх даних.
- 6.2. Перетворення типів даних.
- 6.3. Маніпулювання дата фреймами.

6.1. Оброблення відсутніх даних

Прикладом набору даних, який потребує попереднього очищення, є набір даних із наявністю значень NaN (Not-A-Number, не число). NaNs використовуються для представлення даних, які не визначені або не можуть бути представлені. Pandas відносить відсутні дані як значення NaN, які також часто називають значеннями NA. NaNs можуть змусити функції аналізу даних різко припинитися під час обчислень, викидати помилки або давати неправильні результати. NaNs також можуть бути призначені цілеспрямовано для рівномірного представлення всіх частин інформації, які відсутні у наборі даних, або неправильних, або нульових значень, або даних, яких просто немає. У багатьох наборах даних відсутні дані, оскільки їх не вдалося зібрати правильно або вони відсутні з самого початку. Ще одна поширена причина NaNs – це перевстановлення даних у наборі даних. Розглянемо приклад.

```
import pandas as pd
import numpy as np
# Create a dataframe with random numbers
dataframe = pd.DataFrame(np.random.randn(3,2), index=['a','c','e'], columns=['one','two'])
dataframe
Dataframe with Random Numbers
```

	one	two
a	-0.610609	0.761838
c	-0.460771	-0.646487

```
# Reindexing the dataframe creates NaNs
dataframe = dataframe.reindex(['a','b','c','d','e'])
dataframe
```

Dataframe with NaN values

	one	two
a	-0.610609	0.761838
b	NaN	NaN
c	-0.460771	-0.646487
d	NaN	NaN

Відсутні значення можуть приймати різні форми на основі типу даних. Типи даних Pandas: об'єкти / рядки, int64 / цілі числа, float64 / floats та datetime64 / timestamps. NaN використовуються для невизначених рядків, цілих та дійсних чисел, а NaT – для часових міток. Можуть також бути ситуації, коли значення Python None буде також представляти відсутні дані. Щоб полегшити виявлення відсутніх значень у наборі даних, Pandas надають функції isnull () та notnull ().

6.2. Перетворення типів даних

Pandas має багато вбудованих функцій для перетворення типів даних. У наступному прикладі набір даних 2 складається з цілих чисел, рядків та чисел типу float. Змінимо стовпець 2 з типу даних об'єкта / рядка на числовий тип даних.

```
import pandas as pd
```

```
data2 = [[22, '2017', 0.20], [100, '0.33', 1.112], [6, '12', 0.33]]  
df2 = pd.DataFrame(data2, columns = ['col1', 'col2', 'col3'])  
df2
```

Data2 Data Set

	col1	col2	col3
0	22	2017	0.200
1	100	0.33	1.112
2	6	12	0.330

```
df2.dtypes
```

```
col1    int64  
col2    object  
col3    float64  
dtype: object
```

Функція convert_objects перетворює стовпчик 2 з рядка / об'єкта в числовий тип даних.

```
df2['col2'] = df2['col2'].convert_objects(convert_numeric=True)  
df2
```

Column 2 converted from a string/object to a numeric datatype

	col1	col2	col3
0	22	2017.00	0.200
1	100	0.33	1.112
2	6	12.00	0.330

df2.dtypes

```
col1    int64
col2    float64
col3    float64
dtype: object
```

Стовпець 2 перетворився на тип float64. Це було пов'язано з наявністю рядка "0.33" у стовпці 2. Якби рядок був "33", стовпець перетворився б на цілі числа. Якби рядок був "x" замість "0,33", перетворення призвело б до помилки через неможливість перетворення "x" на числове значення, ціле число або дійсне.

У наступному прикладі дані в стовпці 3 перетворюються з типу даних float64 в об'єктний (рядковий) тип даних. Властивість **dtypes** використовується для перевірки зміни типу даних.

```
df2['col3'] = df2['col3'].astype(str)
df2
```

Data in column 3 is converted from the float64 datatype to the object (string) datatype

	col1	col2	col3
0	22	2017.00	0.2
1	100	0.33	1.112
2	6	12.00	0.33

df2.dtypes

```
col1    int64
col2    float64
col3    object
dtype: object
```

В наступному прикладі дані в стовпці 1 перетворюються з типу даних int64 в тип даних float64. Властивість **dtypes** використовується для перевірки зміни типу даних.

```
df2['col1'] = df2['col1'].astype(float)
df2
```

Data in column 1 is converted from the int64 datatype to the float64 datatype

	col1	col2	col3
0	22.0	2017.00	0.2
1	100.0	0.33	1.112
2	6.0	12.00	0.33

```
df2.dtypes
```

```
col1    float64
col2    float64
col3     object
dtype: object
```

6.3. Маніпулювання дата фреймами

Очищення набору даних є попереднім завданням перед тим, як здійснювати аналіз даних. Маніпулювання двовимірним фреймом даних за допомогою Pandas у Python може включати видалення, додавання або перейменування стовпців або рядків даних. Для цього потрібно викликати функцію **drop ()**, функцію **loc ()** та функцію **rename ()**.

Щоб скинути стовпець даних, викликайте функцію **drop ()**. Створимо простий набір даних. Будемо усувати та додавати стовпці та рядки за допомогою функцій **drop ()** та **add ()**.

```
import pandas as pd
data3 = [[.351, .446, .512], [.112, .980, .122], [.216, .612, .575]]
df3 = pd.DataFrame(data3, columns=['one', 'two', 'three'])
df3
```

Simple data set

	one	two	three
0	0.351	0.446	0.512
1	0.112	0.980	0.122

Перший стовпець видалимо за допомогою функції **drop ()**.

```
df3.drop(['one'], axis=1, inplace=True)
df3
```

Column one is removed using the drop() function

	two	three
0	0.446	0.512
1	0.980	0.122

Вісь відноситься до стовпців, пронумерованих зліва направо, починаючи зі стовпця індексу на **axis=0** і одного стовпця на **axis=1**.

Такого ж результату можна досягти за допомогою команди **del**.

```
del df3 ['one']
```

Тепер видалимо перші два рядки (0,1) з наступною функцією **drop ()**, де [0,1] – це рядки, які будуть скинуті, а axis=0 відноситься до стовпця індексу вкрай ліворуч.

```
df3.drop([0,1], axis=0, inplace=True) df3
```

Axis=0 refers to the index column on the far left

	two	three
--	-----	-------

Додамо стовпець, призначивши йому мітку та значення.

```
df3['four'] = .323  
df3
```

Щоб додати рядок, можна скористатися методом **location** або **loc ()**. Метод визначення місцезнаходження також знаходить максимальний індекс або номер останнього рядка, а потім додає до нього 1, створюючи рядок 3.

```
df3.loc[df3.index.max() +1] = [.232, .444, .587]  
df3
```

Location method also finds the maximum index or last row number and then adds 1 to it, creating row 3

	two	three	four
--	-----	-------	------

Якщо викликати функцію **loc ()** і передати їй номер індексу, вона буде додана як новий нижній рядок. Зверніть увагу, як доданий рядок нумерується 1, хоча це останній рядок.

```
df3.loc[1] = .763  
df3
```

Index number passed to the loc() function, it will be appended as a new bottom row

	two	three	four
2	0.612	0.575	0.323

Ми можемо змінити індекс, призначивши нові значення індексу за допомогою властивості **indexframe** dataframe.

```
i = [1,2,3]  
df3.index = i  
df3
```

New index values assigned with the dataframe index property

	two	three	four
1	0.612	0.575	0.323
2	0.232	0.444	0.587

Поряд із функціями **drop ()** і **loc ()** Pandas також пропонує функцію **rename()** для перейменування мітки стовпців на «один», «два» та «три» відповідно. Для цього потрібно використати функцію **rename()** та призначити стовпці у парі ключ: значення зі старим іменем та новою назвою як пара ключ - значення.

```
df3.rename(columns = {'two':'one', 'three':'two', 'four':'three'}, inplace = True)
df3
```

Use the rename function and assign the columns in a key:value pair with the old name and the new name as the key:value pair

	one	two	three
1	0.612	0.575	0.323

Pandas має вбудовані функції для статистичного аналізу набору даних, включаючи функції для обчислення середніх значень, стандартних відхилень та кореляцій. Створимо набір даних та дата фрейм за допомогою масиву даних data4. Фрейм даних виводиться на екран.

```
import pandas as pd
import numpy as np
```

```
data4 = [[2,3,5,2,11,3,7,8,10,2,12,7,9,7,4,7,8,12,9,10,6,7]
df4 = pd.DataFrame(data4, columns = ['nums'])
df4
```

data4 array
of numbers

	Nums
0	2
1	3
2	5
3	4
4	11
5	3
6	7
7	8
8	10
9	2

Середнє значення всіх чисел обчислюється за допомогою методу dataframe **mean ()**. Середнє значення буде 6,863636. Використовуючи крапковий синтаксис, результат також можна округлити до найближчого цілого числа, приєднавши метод **round ()** після **mean ()**.

```
means = df4.mean()
print(means)

means = df4.mean().round()
print(means)
```

```
nums 6.863636
dtype: float64
```

```
nums 7.0
dtype: float64
```

```
thesum = df.sum()
print(thesum)

thecount = df4.count()
print(thecount)

themean = (df4.sum()/df4.count()).round()
print(themean)
```

```
nums 151
dtype: int64
nums 22
dtype: int64
nums 7.0
dtype: float64
```

Якщо в наборі даних є непарна кількість елементів, медіана – це середнє число, відсортоване за числовим порядком. Якщо в наборі даних є парна кількість елементів, медіана обчислюється за допомогою двох середніх чисел.

```
median = df4.median()
print(median)

thestands = df4.std()
print(thestands)
```

```
nums 7.0
dtype: float64
```

```
nums 3.196657
dtype: float64
```

У другому полі показаний метод **std ()** для обчислення стандартного відхилення. Стандартне відхилення показує величину варіації в наборі значень даних. Низьке стандартне відхилення вказує на те, що числа в наборі даних, як правило, близькі до середнього. Стандартне відхилення знаходимо, беручи квадратний корінь середнього квадратичного відхилення значень від середнього або середнього значення в наборі даних.

Висновок до лекції 6

Найчастіше набори даних, з якими ми працюємо, матимуть несумісність. Очищення даних може включати видалення відсутніх або небажаних значень або зміну формату значень для їх узгодження.

Значення NaN (не число) використовуються для представлення даних, які не визначені або не можуть бути представлені. Pandas посилається на відсутні дані як значення NaN. NaT використовуються для позначок часу. Pandas має багато вбудованих функцій для перетворення типів даних, маніпулювання дата фреймами та проведення статистичного аналізу наборів даних.

Питання для закріплення

1. Як відбувається оброблення відсутніх даних у Pandas?
2. Як відбувається перетворення типів даних?
3. Як відбувається маніпулювання дата фреймами у Pandas?
4. Для чого використовуються функції `convert_objects`, `drop ()`, `loc ()` та `rename ()`?

Список рекомендованої літератури

1. IoT Fundamentals: Big Data & Analytics // Електронний ресурс. Режим доступу: <https://www.netacad.com/courses/iot/big-data-analytics>
2. Pandas // Електронний ресурс. Режим доступу: https://www.w3schools.com/python/pandas/pandas_intro.asp

Лекція 7. Регресійний аналіз даних в Python

План лекції

- 7.1. Методи та типи аналізу машинного навчання.
- 7.2. Регресійний аналіз.
- 7.3. Типи регресійного аналізу.
- 7.4. Застосування регресійного аналізу.

7.1. Методи та типи аналізу машинного навчання

Машинне навчання вирішує проблеми та можливості, що надаються аналітикою Big Data для моделювання наявних даних, щоб передбачити майбутні результати.

У своїй книзі Кевін Патрік Мерфі визначає машинне навчання як "... сукупність методів, які дозволяють автоматично виявляти шаблони в даних, а потім використовувати ці шаблони для прогнозування майбутніх даних або для виконання інших видів прийняття рішень в умовах невизначеності". Наприклад,