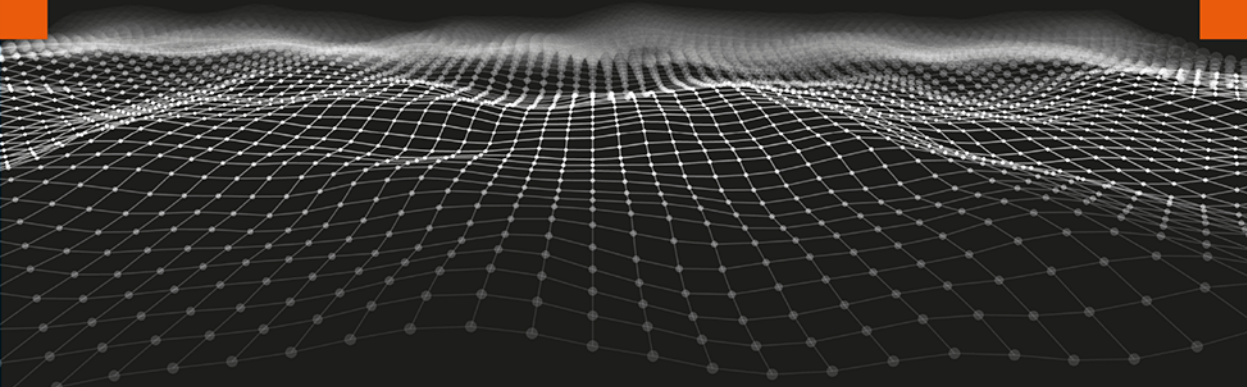
JAK ANALIZOWAĆ DANE z BIBLIOTEKĄ PANDAS

PRAKTYCZNE WPROWADZENIE

WYDANIE II



DANIEL Y. CHEN

Tytuł oryginału: Pandas for Everyone: Python Data Analysis
(Addison-Wesley Data & Analytics Series), 2nd Edition

Tłumaczenie: Piotr Pilch

ISBN: 978-83-289-0151-3

Authorized translation from the English language edition, entitled Pandas for Everyone: Python Data Analysis, 2nd Edition by Daniel Chen, published by Pearson Education, Inc, publishing as Addison-Wesley Professional, Copyright © 2023 Pearson Education, Inc.
All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.
Polish language edition published by Helion S.A., Copyright © 2023.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/jaanp2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
https://ftp.helion.pl/przyklady/jaanp2.zip

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

• Kup książkę
• Poleć książkę
• Oceń książkę

• Księgarnia internetowa
• Lubię to! » Nasza społeczność

Spis treści

Słowo wstępne do wydania drugiego **15**

Słowo wstępne do wydania pierwszego **17**

Przedmowa **19**

Podziękowania **27**

O autorze **31**

Zmiany w wydaniu drugim **32**

Część I Wprowadzenie **33**

Rozdział 1 Typ danych DataFrame biblioteki Pandas — podstawy **35**

1.1. Wprowadzenie 35

Cele rozdziału 36

1.2. Ładowanie pierwszego zbioru danych 36

1.3. Sprawdzanie kolumn, wierszy i komórek 39

1.3.1. Wybieranie i określanie podzbioru kolumn na podstawie nazwy 39

1.3.2. Określanie podzbioru wierszy 43

1.3.3. Określanie podzbioru wierszy za pomocą numeru wiersza: atrybut .iloc[] 45

1.3.4. Użycie kombinacji 47

1.3.5. Określanie podzbioru wierszy i kolumn 53

1.4. Obliczenia grupowane i agregowane 54

1.4.1. Średnie grupowane 55

1.4.2. Liczebności grupowane 58

1.5. Podstawowy wykres 58

Podsumowanie 59

Kup książkę Poleć książkę

Rozdział 2 Struktury danych biblioteki Pandas — podstawy 61
Cele rozdziału 61
2.1. Tworzenie własnych danych 61
2.1.1. Tworzenie obiektu Series 62
2.1.2. Tworzenie obiektu DataFrame 63
2.2. Obiekty Series 63
2.2.1. Obiekt Series przypomina typ ndarray 65
2.2.2. Określanie podzbioru wartości boolowskich: obiekt Series 67
2.2.3. Operacje są automatycznie wyrównywane i wektoryzowane (rozgłaszanie) 69
2.3. Obiekt DataFrame 72
2.3.1. Części obiektu DataFrame 72
2.3.2. Określanie podzbioru wartości boolowskich: obiekty DataFrame 73
2.3.3. Operacje są automatycznie wyrównywane i wektoryzowane (rozgłaszanie) 73
2.4. Wprowadzanie zmian w obiektach Series i DataFrame 75
2.4.1. Dodawanie dodatkowych kolumn 75
2.4.2. Bezpośrednie modyfikowanie kolumny 76
2.4.3. Modyfikowanie kolumn za pomocą metody .assign() 79
2.4.4. Usuwanie wartości 81
2.5. Eksportowanie i importowanie danych 81
2.5.1. „Peklowanie” 82
2.5.2. Format danych CSV 84
2.5.3. Excel 84
2.5.4. Format Feather 86
2.5.5. Projekt Arrow 87
2.5.6. Słownik 87
2.5.7. Format JSON 88
2.5.8. Inne typy danych wyjściowych 91
Podsumowanie 92

Rozdział 3 Tworzenie wykresów — podstawy 93
Cele rozdziału 93
3.1. Dlaczego warto wizualizować dane? 93
3.2. Podstawy obsługi biblioteki matplotlib 94
3.2.1. Obiekty rysunków i podwykresy z osiami 95
3.2.2. Anatomia rysunku 99
3.3. Tworzenie graficznych wizualizacji danych statystycznych za pomocą biblioteki matplotlib 100
3.3.1. Jednozmienność (pojedyncza zmienna) 101
3.3.2. Dwuzmienność (dwie zmienne) 101
3.3.3. Dane wielozmienne 103

Kup książkę Poleć książkę

3.4. Biblioteka seaborn 105
3.4.1. Jednozmienność 106
3.4.2. Dane dwuzmienne 110
3.4.3. Dane wielozmienne 120
3.4.4. Aspekty 124
3.4.5. Style i kompozycje biblioteki seaborn 129
3.4.6. Jak korzystać z dokumentacji biblioteki seaborn? 133
3.4.7. Interfejs biblioteki seaborn następnej generacji 135
3.5. Metoda tworzenia wykresów za pomocą biblioteki Pandas 136
3.5.1. Histogram 136
3.5.2. Wykres gęstości 137
3.5.3. Wykres punktowy 137
3.5.4. Wykres przedziałów sześciokątnych (hexbin) 138
3.5.5. Wykres pudełkowy 139
Podsumowanie 140

Rozdział 4 Dane uporządkowane 141

Cele rozdziału 142
Uwaga dotycząca niniejszego rozdziału 142
4.1. Kolumny zawierają wartości, a nie zmienne 142
4.1.1. Utrwalenie jednej kolumny 142
4.1.2. Utrwalenie wielu kolumn 145
4.2. Kolumny zawierają wiele zmiennych 146
4.2.1. Osobne dzielenie i dodawanie kolumn 147
4.2.2. Dzielenie i łączenie kolumn w jednym kroku 149
4.3. Zmienne znajdują się w wierszach i kolumnach 150
Podsumowanie 153

Rozdział 5 Zastosowanie funkcji 154

Cele rozdziału 154
Uwaga dotycząca niniejszego rozdziału 155
5.1. Elementarz funkcji 155
5.2. Zastosowanie funkcji (podstawy) 156
5.2.1. Zastosowanie funkcji względem obiektu Series 157
5.2.2. Zastosowanie funkcji względem obiektu DataFrame 158
5.3. Funkcje wektoryzowane 161
5.3.1. Wektoryzacja za pomocą biblioteki NumPy 162
5.3.2. Wektoryzacja za pomocą biblioteki Numba 163
5.4. Funkcje lambda (funkcje anonimowe) 164
Podsumowanie 165

Kup książkę Poleć książkę

Część II Przetwarzanie danych 167

Rozdział 6 Łączenie danych 169

Cele rozdziału 169
6.1. Łączenie zbiorów danych 169
6.2. Konkatenacja 170
6.2.1. Części przeglądowe obiektu DataFrame 171
6.2.2. Dodawanie wierszy 171
6.2.3. Dodawanie kolumn 174
6.2.4. Konkatenacja z różnymi indeksami 175
6.3. Jednostki obserwacyjne w obrębie wielu tabel 178
6.3.1. Ładowanie wielu plików za pomocą pętli 180
6.3.2. Ładowanie wielu plików przy użyciu listy składanej 182
6.4. Scalanie wielu zbiorów danych 183
6.4.1. Scalanie typu „jedna z jedną” 185
6.4.2. Scalanie typu „wiele z jedną” 186
6.4.3. Scalanie typu „wiele z wieloma” 187
6.4.4. Sprawdzanie wyników pracy za pomocą asercji 189
Podsumowanie 189

Rozdział 7 Normalizacja danych 190

Cele rozdziału 190
7.1. Wiele jednostek obserwacyjnych w tabeli (normalizacja) 190
Podsumowanie 194

Rozdział 8 Operacje grupowania: dzielenie, stosowanie i łączenie 195

Cele rozdziału 196
8.1. Agregacja 196
8.1.1. Podstawowa agregacja danych grupowanych z jedną zmienną 196
8.1.2. Wbudowane metody agregacji 198
8.1.3. Funkcje agregacji 199
8.1.4. Użycie wielu funkcji jednocześnie 202
8.1.5. Zastosowanie słownika w metodzie .agg() lub .aggregate() 202
8.2. Transformacja 204
8.2.1. Przykład wyniku standardowego z 204
8.2.2. Przykład z brakującymi wartościami 206
8.3. Filtrowanie 208
8.4. Obiekt pandas.core.groupby.DataFrameGroupBy 209
8.4.1. Grupy 209
8.4.2. Obliczenia w ramach grupowania obejmujące wiele zmiennych 210
8.4.3. Wybieranie grupy 211
8.4.4. Iteracja w obrębie grup 211

Kup książkę Poleć książkę

8.4.5. Wiele grup 213
8.4.6. „Spłaszczanie" wyników (.reset_index()) 213
8.5. Zastosowanie obiektu MultiIndex 214
Podsumowanie 218

Część III Typy danych 219

Rozdział 9 Brakujące dane 221
Cele rozdziału 221
9.1. Czym jest wartość NaN? 221
9.2. Skąd biorą się brakujące wartości? 223
9.2.1. Ładowanie danych 223
9.2.2. Scalone dane 224
9.2.3. Wartości wprowadzane przez użytkownika 225
9.2.4. Ponowne indeksowanie 226
9.3. Zajmowanie się brakującymi danymi 227
9.3.1. Znajdowanie brakujących danych i określanie ich ilości 228
9.3.2. Oczyszczanie danych z brakującymi wartościami 229
9.3.3. Obliczenia uwzględniające brakujące dane 233
9.4. Brakująca wartość NA wbudowana w bibliotece Pandas 234
Podsumowanie 235

Rozdział 10 Typy danych 236
Cele rozdziału 236
10.1. Typy danych 236
10.2. Przekształcanie typów 237
10.2.1. Konwersja do postaci obiektów łańcuchów 237
10.2.2. Przekształcanie w wartości liczbowe 238
10.3. Dane kategorialne 242
10.3.1. Przekształcanie w kategorię 242
10.3.2. Przetwarzanie danych kategorialnych 243
Podsumowanie 244

Rozdział 11 Łańcuchy i dane tekstowe 245
Wprowadzenie 245
Cele rozdziału 245
11.1. Łańcuchy 245
11.1.1. Określanie podzbioru i dzielenie łańcuchów 246
11.1.2. Uzyskanie ostatniego znaku łańcucha 247
11.2. Metody łańcuchowe 249
11.3. Dodatkowe metody łańcuchowe 251
11.3.1. Metoda join 251
11.3.2. Metoda splitlines 251

Kup książkę Poleć książkę

11.4. Formatowanie łańcuchów (f-łańcuchy) ... 253
11.4.1. Formatowanie liczb ... 254
11.5. Wyrażenia regularne ... 255
11.5.1. Dopasowanie wzorca ... 257
11.5.2. Pamiętaj, jakich używasz wzorców wyrażeń regularnych ... 259
11.5.3. Znajdowanie wzorca ... 261
11.5.4. Zastępowanie wzorca ... 261
11.5.5. Kompilowanie wzorca ... 262
11.6. Biblioteka regex ... 263
Podsumowanie ... 264

Rozdział 12 Daty i godziny ... 265

Cele rozdziału ... 265
12.1. Obiekt datetime języka Python ... 265
12.2. Przekształcanie do postaci ramki danych ... 266
12.3. Ładowanie danych zawierających daty ... 269
12.4. Wyodrębnianie składników daty ... 270
12.5. Obliczenia obejmujące daty i obiekty timedelta ... 273
12.6. Metody obiektu datetime ... 274
12.7. Uzyskiwanie danych notowań giełdowych ... 277
12.8. Określanie podzbioru danych na podstawie dat ... 278
12.8.1. Obiekt DatetimeIndex ... 279
12.8.2. Obiekt TimedeltaIndex ... 280
12.9. Zakresy dat ... 281
12.9.1. Częstotliwości ... 283
12.9.2. Przesunięcia ... 284
12.10. Wartości przesuwające ... 284
12.11. Ponowne próbkowanie ... 290
12.12. Strefy czasowe ... 292
12.13. Biblioteka Arrow do lepszej obsługi dat i godzin ... 294
Podsumowanie ... 294

Część IV Modelowanie danych ... 295

Rozdział 13 Regresja liniowa (wynikowa zmienna ciągła) ... 297

13.1. Prosta regresja liniowa ... 297
13.1.1. Użycie biblioteki statsmodels ... 298
13.1.2. Zastosowanie biblioteki scikit-learn (sklearn) ... 299
13.2. Regresja wielokrotna ... 301
13.2.1. Użycie biblioteki statsmodels ... 301
13.2.2. Zastosowanie biblioteki scikit-learn (sklearn) ... 302

Kup książkę Poleć książkę

13.3. Modele ze zmiennymi kategorialnymi 303
13.3.1. Zmienne kategorialne w bibliotece statsmodels 303
13.3.2. Zmienne kategorialne w bibliotece scikit-learn (sklearn) 305
13.4. Kodowanie One-Hot w bibliotece scikit-learn z wykorzystaniem potoków transformera 307
Podsumowanie 309

Rozdział 14 Uogólnione modele liniowe 310
Coś o tym rozdziale 310
14.1. Regresja logistyczna (binarna zmienna wyjściowa) 310
14.1.1. Użycie biblioteki statsmodels 312
14.1.2. Zastosowanie biblioteki sklearn 314
14.1.3. Zachowaj ostrożność w przypadku domyślnych wartości biblioteki scikit-learn (sklearn) 315
14.2. Regresja Poissona (ilościowa zmienna wynikowa) 317
14.2.1. Użycie biblioteki statsmodels 317
14.2.2. Ujemna regresja dwumianowa w przypadku nadmiernej dyspersji 319
14.3. Bardziej uogólnione modele liniowe 321
Podsumowanie 322

Rozdział 15 Analiza przeżycia 323
15.1. Dane analizy przeżycia 324
15.2. Krzywe Kaplana-Meiera 324
15.3. Model proporcjonalnego hazardu Coxa 326
15.3.1. Testowanie założeń modelu Coxa 327
Podsumowanie 328

Rozdział 16 Diagnostyka modeli 329
16.1. Residua 329
16.1.1. Wykresy kwantylowe K-K 332
16.2. Porównanie wielu modeli 334
16.2.1. Korzystanie z modeli liniowych 334
16.2.2. Zastosowanie uogólnionych modeli liniowych 337
16.3. Walidacja krzyżowa k-krotna 339
Podsumowanie 342

Rozdział 17 Regularyzacja 343
17.1. Dlaczego regularyzacja? 343
17.2. Regresja LASSO 345
17.3. Regresja grzbietowa 346
17.4. Sieć elastyczna 347
17.5. Walidacja krzyżowa 349
Podsumowanie 350

Kup książkę Poleć książkę

Rozdział 18 Klasteryzacja 351
18.1. k-średnie 351
18.1.1. Ograniczanie liczby wymiarów za pomocą analizy PCA 353
18.2. Klastrowanie hierarchiczne 357
18.2.1. Klastrowanie kompletne 358
18.2.2. Klastrowanie pojedyncze 358
18.2.3. Klastrowanie ze średnią 359
18.2.4. Klastrowanie z centroidem 360
18.2.5. Klastrowanie metodą Warda 360
18.2.6. Ręczne ustawianie progu 361
Podsumowanie 361

Część V Podsumowanie 363

Rozdział 19 Świat poza obrębem biblioteki Pandas 365
19.1. Stos do obliczeń (naukowych) 365
19.2. Wydajność 366
19.2.1. Pomiar czasu wykonywania kodu 366
19.2.2. Profilowanie kodu 366
19.2.3. Moduł concurrent.futures 366
19.3. Dask 367
19.4. Siuba 367
19.5. Ibis 367
19.6. Polars 367
19.7. PyJanitor 368
19.8. Pandera 368
19.9. Uczenie maszynowe 368
19.10. Publikowanie 368
19.11. Panele kontrolne 369
Podsumowanie 369

Rozdział 20 Działanie w pojedynkę jest niebezpieczne! 370
20.1. Lokalne spotkania 370
20.2. Konferencje 371
20.3. The Carpentries 371
20.4. Podcasty 372
20.5. Inne zasoby 372
Podsumowanie 372

Kup książkę Poleć książkę

Dodatki 373

Dodatek A Mapy pojęć 375

Dodatek B Instalacja i konfiguracja 378

B.1. Instalacja języka Python 378
B.1.1. Anaconda 378
B.1.2. Miniconda 379
B.1.3. Odinstalowywanie dystrybucji Anaconda lub Miniconda 379
B.1.4. pyenv 379
B.2. Instalowanie pakietów języka Python 380
B.3. Pobieranie zbiorów danych używanych w książce 380

Dodatek C Wiersz poleceń 382

C.1. Instalacja 382
C.1.1. System Windows 382
C.1.2. System Mac 383
C.1.3. System Linux 383
C.2. Podstawy 383

Dodatek D Szablony projektowe 384

Dodatek E Zastosowanie języka Python 385

E.1. Wiersz poleceń i edytor tekstu 385
E.2. Python i IPython 386
E.3. Jupyter 386
E.4. Zintegrowane środowiska programistyczne IDE 387

Dodatek F Katalogi robocze 388

Dodatek G Środowiska 390

G.1. Środowiska systemu conda 390
G.2. Pyenv + Pipenv 392

Dodatek H Instalacja pakietów 394

H.1. Aktualizowanie pakietów 395

Dodatek I Importowanie bibliotek 396

Dodatek J Styl kodu 398

J.1. Znaki podziału wiersza w kodzie 398

Dodatek K Kontenery: listy, krotki i słowniki 400

K.1. Listy 400
K.2. Krotki 401
K.3. Słowniki 402

Kup książkę Poleć książkę

Dodatek L Określanie wartości za pomocą składni wycinków 404
Dodatek M Pętle 406
Dodatek N Listy składane 408
Dodatek O Funkcje 410
O.1. Parametry domyślne 412
O.2. Parametry arbitralne 412
O.2.1. Wyrażenie *args 413
O.2.2. Wyrażenie **kwargs 413
Dodatek P Zakresy i generatory 414
Dodatek Q Przypisanie wielokrotne 416
Dodatek R Typ ndarray biblioteki NumPy 418
Dodatek S Klasy 420
Dodatek T Komunikat SettingWithCopyWarning 422
T.1. Modyfikowanie podzbioru danych 422
T.2. Zastępowanie wartości 424
T.3. Dodatkowe zasoby informacji 425
Dodatek U Tworzenie łańcuchów metod 426
Dodatek V Czas wykonywania kodu 428
Dodatek W Formatowanie łańcuchów 430
W.1. Formatowanie w stylu języka C 430
W.2. Formatowanie łańcuchów: metoda .format() 431
W.3. Formatowanie liczb 431
Dodatek X Instrukcje warunkowe (if-elif-else) 433
Dodatek Y Przykład regresji logistycznej ze zbiorem danych ACS dla Nowego Jorku 435
Y.0.1. Użycie biblioteki sklearn 439
Dodatek Z Replikowanie wyników za pomocą języka R 442
Z.1. Regresja liniowa 443
Z.2. Regresja logistyczna 445
Z.3. Regresja Poissona 446
Z.3.1. Ujemna regresja dwumianowa w przypadku nadmiernej dyspersji 446
Skorowidz 449

Kup książkę Poleć książkę

9

Brakujące dane

Rzadko uzyskasz zbiór danych bez żadnych brakujących wartości. Istnieje wiele reprezentacji brakujących danych. W bazach danych są to wartości `NULL`. W niektórych językach programowania używana jest wartość `NA`. Zależnie od tego, skąd uzyskasz dane, brakujące wartości mogą mieć postać pustego łańcucha (`''`) lub nawet wartości liczbowych, takich jak `88` lub `99`. W przypadku biblioteki Pandas brakujące wartości są wyświetlane jako wartość `NaN`.

Cele rozdziału

- Identyfikowanie sposobu reprezentowania brakujących wartości w bibliotece Pandas.
- Przedstawienie potencjalnych sytuacji powodujących brak wartości w czasie przetwarzania danych
- Użycie różnych funkcji do uzupełniania brakujących wartości.

9.1. Czym jest wartość NaN?

Zastosowana w bibliotece Pandas wartość `NaN` ma swoje źródło w bibliotece *numpy*. W przypadku biblioteki Pandas brakujące wartości mogą być używane lub wyświetlane na kilka sposobów, czyli jako wartości `NaN`, `NAN` lub `nan`. Wszystkie te postaci są jednakowe z punktu widzenia sposobu określania brakującej wartości (zmiennoprzecinkowej), lecz nie pod względem równości. W dodatku I opisano, jak te brakujące wartości są importowane.

```
# Importowane są jedynie brakujące wartości z biblioteki numpy
from numpy import NaN, NAN, nan
```

Brakujące wartości różnią się od innych typów danych tym, że tak naprawdę nie są równe czemukolwiek (nawet samym sobie). Dane są nieobecne, dlatego nie występuje pojęcie równości.

Kup książkę Poleć książkę

Wartość `NaN` nie jest równorzędna wartości 0 ani pustemu łańcuchowi (`''`). To logika trójwartościowa.

```
print(NaN == True)

False
print(NaN == 0)

False
print(NaN == '')

False
print(NaN == NaN)

False
print(NaN == NAN)

False
print(NaN == nan)

False
print(nan == NAN)

False
```

Biblioteka Pandas oferuje funkcję `isnull()` służącą do sprawdzania pod kątem brakujących wartości.

```
import pandas as pd

print(pd.isnull(NaN))

True

print(pd.isnull(nan))

True

print(pd.isnull(NAN))

True
```

Biblioteka Pandas zapewnia również funkcję `notnull()` przeznaczoną do sprawdzania pod kątem istniejących wartości.

```
print(pd.notnull(NaN))

False

print(pd.notnull(42))

True

print(pd.notnull('missing'))

True
```


Kup książkę Poleć książkę

9.2. Skąd biorą się brakujące wartości?

Brakujące wartości mogą się pojawić w wyniku załadowania ich wraz ze zbiorem danych lub przeprowadzenia procesu ujednolicania.

9.2.1. Ładowanie danych

Dane ankietowe użyte w rozdziale 6. uwzględniały zbiór danych `visited` z brakującymi danymi. Po załadowaniu tych danych biblioteka Pandas automatycznie znalazła komórkę brakujących danych i zapewniła nam ramkę danych z wartością `NaN` w odpowiedniej komórce. W przypadku funkcji `read_csv()` z wczytywaniem brakujących wartości są powiązane trzy parametry: `na_values`, `keep_default_na` i `na_filter`.

Parametr `na_values` umożliwia określenie dodatkowych brakujących wartości lub wartości `NaN`. Masz możliwość przekazania obiektu typu `str` (czyli łańcucha) języka Python lub obiektu podobnego do listy, aby w chwili wczytania pliku w kodzie został on automatycznie rozpoznany jako brakujące wartości. Domyślne brakujące wartości, takie jak `NAN`, `NaN` lub `nan`, są oczywiście już dostępne, dlatego parametr ten nie zawsze jest stosowany. W niektórych danych związanych ze służbą zdrowia brakująca wartość może być reprezentowana przez liczbę `99`. W celu skorzystania z niej należałoby użyć przypisania `na_values=[99]`.

Parametr `keep_default_na` typu `bool` (czyli taki, dla którego używa się wartości boolowskiej `True` lub `False`) umożliwia określenie, czy jakiekolwiek dodatkowe wartości muszą być rozpatrywane jako brakujące. Domyślnie parametr ten ma wartość `True`, co oznacza, że wszelkie dodatkowe brakujące wartości określone za pomocą parametru `na_values` będą dołączane do listy brakujących wartości. Dla parametru `keep_default_na` można jednak ustawić też wartość `False` (`keep_default_na=False`), która sprawi, że zostaną użyte *tylko* brakujące wartości podane za pomocą parametru `na_values`.

I wreszcie, parametr `na_filter` typu `bool` określi, czy jakiekolwiek wartości będą odczytywane jako brakujące. Wartość domyślna `True` tego parametru (`na_filter=True`) oznacza, że brakujące wartości będą kodowane w postaci wartości `NaN`. Jeśli zastosuje się przypisanie `na_filter=False`, nic nie będzie ponownie kodowane jako brakująca wartość. Parametr `na_filter` może być traktowany jako sposób na wyłączenie wszystkich parametrów ustawionych w powiązaniu z parametrami `na_values` i `keep_default_na`. Bardziej prawdopodobne jest jednak to, że parametr `na_filter` zostanie użyty, gdy będziesz chciał osiągnąć wzrost wydajności w wyniku załadowania danych bez brakujących wartości.

```
# Określenie lokalizacji danych
visited_file = 'data/survey_visited.csv'

print(pd.read_csv(visited_file))
```

```
   ident   site       dated
0    619   DR-1  1927-02-08
1    622   DR-1  1927-02-10
2    734   DR-3  1939-01-07
3    735   DR-3  1930-01-12
4    751   DR-3  1930-02-26
5    752   DR-3         NaN
```


Kup książkę Poleć książkę

```
6    837  MSK-4  1932-01-14
7    844   DR-1  1932-03-22

print(pd.read_csv(visited_file, keep_default_na=False))

    ident   site       dated
0     619   DR-1  1927-02-08
1     622   DR-1  1927-02-10
2     734   DR-3  1939-01-07
3     735   DR-3  1930-01-12
4     751   DR-3  1930-02-26
5     752   DR-3
6     837  MSK-4  1932-01-14
7     844   DR-1  1932-03-22

print(
  pd.read_csv(visited_file, na_values=[""], keep_default_na=False)
)

   ident   site       dated
0    619   DR-1  1927-02-08
1    622   DR-1  1927-02-10
2    734   DR-3  1939-01-07
3    735   DR-3  1930-01-12
4    751   DR-3  1930-02-26
5    752   DR-3         NaN
6    837  MSK-4  1932-01-14
7    844   DR-1  1932-03-22
```

9.2.2. Scalone dane

W rozdziale 6. wyjaśniono, jak połączyć zbiory danych. Niektóre z przykładów zamieszczonych w niniejszym rozdziale zawierały brakujące wartości w danych wyjściowych. Jeśli ponownie utworzymy scaloną tabelę z punktu 6.4.3, w danych wynikowych operacji scalania będą widoczne brakujące wartości.

```
visited = pd.read_csv('data/survey_visited.csv')
survey = pd.read_csv('data/survey_survey.csv')

print(visited)

   ident   site       dated
0    619   DR-1  1927-02-08
1    622   DR-1  1927-02-10
2    734   DR-3  1939-01-07
3    735   DR-3  1930-01-12
4    751   DR-3  1930-02-26
5    752   DR-3         NaN
6    837  MSK-4  1932-01-14
7    844   DR-1  1932-03-22

print(survey)

   taken person quant reading
```


Kup książkę Poleć książkę

```
0    619   dyer   rad    9.82
1    619   dyer   sal    0.13
2    622   dyer   rad    7.80
3    622   dyer   sal    0.09
4    734     pb   rad    8.41
..   ...    ...   ...     ...
16   752    roe   sal   41.60
17   837   lake   rad    1.46
18   837   lake   sal    0.21
19   837    roe   sal   22.50
20   844    roe   rad   11.25

[21 rows x 4 columns]

vs = visited.merge(survey, left_on='ident', right_on='taken')
print(vs)

   ident   site       dated taken person quant  reading
0    619   DR-1  1927-02-08   619   dyer   rad     9.82
1    619   DR-1  1927-02-08   619   dyer   sal     0.13
2    622   DR-1  1927-02-10   622   dyer   rad     7.80
3    622   DR-1  1927-02-10   622   dyer   sal     0.09
4    734   DR-3  1939-01-07   734     pb   rad     8.41
..   ...    ...         ...   ...    ...   ...      ...
16   752   DR-3         NaN   752    roe   sal    41.60
17   837  MSK-4  1932-01-14   837   lake   rad     1.46
18   837  MSK-4  1932-01-14   837   lake   sal     0.21
19   837  MSK-4  1932-01-14   837    roe   sal    22.50
20   844   DR-1  1932-03-22   844    roe   rad    11.25

[21 rows x 7 columns]
```

9.2.3. Wartości wprowadzane przez użytkownika

Użytkownik również może spowodować pojawienie się brakujących wartości, tworząc na przykład wektor wartości w wyniku obliczeń lub zapewniając ręcznie przygotowany wektor. Aby bazować na przykładach z podrozdziału 2.1, utworzymy własne dane z brakującymi wartościami. Wartości `NaN` są poprawne zarówno w przypadku obiektów `Series`, jak i obiektów `DataFrame`.

```
# Brakująca wartość w obiekcie Series
num_legs = pd.Series({'goat': 4, 'amoeba': nan})
print(num_legs)

goat 4.0
amoeba NaN
dtype: float64

# Brakująca wartość w obiekcie DataFrame
scientists = pd.DataFrame(
  {
    "Name": ["Rosaline Franklin", "William Gosset"],
    "Occupation": ["Chemist", "Statistician"],
    "Born": ["1920-07-25", "1876-06-13"],
    "Died": ["1958-04-16", "1937-10-16"],
```


Kup książkę Poleć książkę

```
    "missing": [NaN, nan],
  }
)
print(scientists)
```

```
                Name     Occupation        Born        Died  missing
0   Rosaline Franklin        Chemist  1920-07-25  1958-04-16      NaN
1      William Gosset   Statistician  1876-06-13  1937-10-16      NaN
```

Zauważysz, że typem danych kolumny `missing` będzie `float64`. Wynika to z tego, że brakująca wartość `NaN` z biblioteki *numpy* to wartość zmiennoprzecinkowa.

```
print(scientists.dtypes)
```

```
Name            object
Occupation      object
Born            object
Died            object
missing        float64
dtype: object
```

Masz też możliwość przypisania kolumny brakujących wartości bezpośrednio do ramki danych.

```
# Utworzenie nowej ramki danych
scientists = pd.DataFrame(
  {
    "Name": ["Rosaline Franklin", "William Gosset"],
    "Occupation": ["Chemist", "Statistician"],
    "Born": ["1920-07-25", "1876-06-13"],
    "Died": ["1958-04-16", "1937-10-16"],
  }
)

# Przypisanie kolumny brakujących wartości
scientists["missing"] = nan

print(scientists)
```

```
                Name    Occupation        Born        Died  missing
0  Rosaline Franklin       Chemist  1920-07-25  1958-04-16      NaN
1     William Gosset  Statistician  1876-06-13  1937-10-16      NaN
```

9.2.4. Ponowne indeksowanie

Kolejnym sposobem na to, by pojawiły się brakujące wartości w używanych danych, jest przeprowadzenie ponownego indeksowania ramki danych. Jest to przydatne, gdy zamierzasz dodać nowe indeksy do ramki danych, lecz nadal chcesz zachować jej oryginalne wartości. Typowe tego zastosowanie ma miejsce w sytuacji, gdy indeks reprezentuje pewien interwał czasowy, a Tobie zależy na dodaniu kolejnych dat.

Jeśli wymagane byłoby sprawdzenie lat tylko z przedziału od roku 2000 do roku 2010 na wykresie zbioru danych Gapminder zamieszczonym w podrozdziale 1.5, moglibyśmy wykonać te same operacje grupowane, określić podzbiór danych, a następnie ponownie go zindeksować.

Kup książkę Poleć książkę

```
gapminder = pd.read_csv('data/gapminder.tsv', sep='\t')

life_exp = gapminder.groupby(['year'])['lifeExp'].mean()
print(life_exp)

year
1952     49.057620
1957     51.507401
1962     53.609249
1967     55.678290
1972     57.647386
           ...
1987     63.212613
1992     64.160338
1997     65.014676
2002     65.694923
2007     67.007423
Name: lifeExp, Length: 12, dtype: float64
```

W celu przeprowadzenia ponownego indeksowania należy określić podzbiór danych i skorzystać z metody `.reindex()`.

```
# Określanie podzbioru danych
y2000 = life_exp[life_exp.index > 2000]
print(y2000)

year
2002    65.694923
2007    67.007423
Name: lifeExp, dtype: float64

# Ponowne indeksowanie
print(y2000.reindex(range(2000, 2010)))

year
2000           NaN
2001           NaN
2002     65.694923
2003           NaN
2004           NaN
2005           NaN
2006           NaN
2007     67.007423
2008           NaN
2009           NaN
Name: lifeExp, dtype: float64
```

9.3. Zajmowanie się brakującymi danymi

Gdy już wiadomo, jak mogą zostać utworzone brakujące wartości, dowiedzmy się, jak one funkcjonują podczas korzystania z danych.

Kup książkę Poleć książkę

9.3.1. Znajdowanie brakujących danych i określanie ich ilości

Jednym ze sposobów uzyskania liczby brakujących wartości jest użycie metody `count()`.

```
ebola = pd.read_csv('data/country_timeseries.csv')

# Określenie liczby istniejących wartości
print(ebola.count())
```

```
Date                 122
Day                  122
Cases_Guinea          93
Cases_Liberia         83
Cases_SierraLeone     87
                     ...
Deaths_Nigeria        38
Deaths_Senegal        22
Deaths_UnitedStates   18
Deaths_Spain          16
Deaths_Mali           12
Length: 18, dtype: int64
```

Możliwe jest również odjęcie liczby wierszy istniejących danych od łącznej liczby wierszy.

```
num_rows = ebola.shape[0]
num_missing = num_rows - ebola.count()
print(num_missing)
```

```
Date                   0
Day                    0
Cases_Guinea          29
Cases_Liberia         39
Cases_SierraLeone     35
                     ...
Deaths_Nigeria        84
Deaths_Senegal       100
Deaths_UnitedStates  104
Deaths_Spain         106
Deaths_Mali          110
Length: 18, dtype: int64
```

Aby określić łączną liczbę brakujących wartości dla całych danych lub konkretnej kolumny, możesz skorzystać z funkcji `count_non zero()` biblioteki *numpy* w połączeniu z metodą `.isnull()`.

```
import numpy as np

print(np.count_nonzero(ebola.isnull()))
```

```
1214
```

```
print(np.count_nonzero(ebola['Cases_Guinea'].isnull()))
```

```
29
```


Kup książkę Poleć książkę

Innym sposobem ustalenia ilości brakujących danych jest użycie metod `.value_counts()` względem serii. W efekcie zostanie wyświetlona tabela częstotliwości występowania wartości. Jeśli zastosujesz parametr `dropna`, możesz też uzyskać liczbę brakujących wartości.

```
# Liczba wartości w kolumnie Cases_Guinea
cnts = ebola.Cases_Guinea.value_counts(dropna=False)
print(cnts)

NaN        29
86.0        3
495.0       2
112.0       2
390.0       2
           ..
1199.0      1
1298.0      1
1350.0      1
1472.0      1
49.0        1
Name: Cases_Guinea, Length: 89, dtype: int64
```

Wyniki są sortowane, dlatego możliwe jest określenie podzbioru danych wektora ilości tak, aby przyjrzeć się tylko brakującym wartościom.

```
# Wybranie wartości obiektu Series, gdzie indeks ma wartość NaN
print(cnts.loc[pd.isnull(cnts.index)])

NaN    29
Name: Cases_Guinea, dtype: int64
```

W języku Python wartość `True` odpowiada liczbie całkowitej 1, a wartość `False` jest równa wartości całkowitej 0. Możemy skorzystać z tego faktu w celu uzyskania liczby brakujących wartości przez przeprowadzenie sumowania wektora wartości boolowskich za pomocą metody `.sum()`.

```
# Sprawdzenie, czy wartość jest wartością brakującą, oraz zsumowanie wyników
print(ebola.Cases_Guinea.isnull().sum())

29
```

9.3.2. Oczyszczanie danych z brakującymi wartościami

Istnieje wiele różnych sposobów zajmowania się danymi z brakującymi wartościami. Można na przykład zastąpić takie wartości inną wartością, uzupełnić tego rodzaju dane istniejącymi danymi lub usunąć te dane ze zbioru danych.

9.3.2.1. Ponowne kodowanie lub zastępowanie

Za pomocą metody `.fillna()` można przeprowadzić ponowne kodowanie brakujących wartości do postaci innej wartości. Dla przykładu załóżmy, że brakujące wartości miałyby być ponownie kodowane jako wartość 0. Korzystając z metody `.fillna()`, można dokonać ponownego kodowania do konkretnej wartości.

Kup książkę Poleć książkę

```
# Uzupełnienie brakujących wartości za pomocą wartości 0 i wyświetlenie tylko pierwszych pięciu kolumn
print(ebola.fillna(0).iloc[:, 0:5])
```

```
           Date Day  Cases_Guinea  Cases_Liberia  Cases_SierraLeone
0      1/5/2015  289        2776.0            0.0            10030.0
1      1/4/2015  288        2775.0            0.0             9780.0
2      1/3/2015  287        2769.0         8166.0             9722.0
3      1/2/2015  286           0.0         8157.0                0.0
4    12/31/2014  284        2730.0         8115.0             9633.0
..          ...  ...           ...            ...                ...
117   3/27/2014    5         103.0            8.0                6.0
118   3/26/2014    4          86.0            0.0                0.0
119   3/25/2014    3          86.0            0.0                0.0
120   3/24/2014    2          86.0            0.0                0.0
121   3/22/2014    0          49.0            0.0                0.0

[122 rows x 5 columns]
```

9.3.2.2. Wypełnianie do przodu

Możliwe jest zastosowanie wbudowanych metod umożliwiających wypełnianie do przodu lub wstecz. W przypadku wypełniania danymi do przodu ostatnia znana wartość (w kierunku od góry do dołu) jest używana dla następnej brakującej wartości. Dzięki temu brakujące wartości są zastępowane ostatnią znaną i zarejestrowaną wartością.

```
print(ebola.fillna(method='ffill').iloc[:, 0:5])
```

```
           Date  Day  Cases_Guinea Cases_Liberia  Cases_SierraLeone
0      1/5/2015  289        2776.0           NaN            10030.0
1      1/4/2015  288        2775.0           NaN             9780.0
2      1/3/2015  287        2769.0        8166.0             9722.0
3      1/2/2015  286        2769.0        8157.0             9722.0
4    12/31/2014  284        2730.0        8115.0             9633.0
..          ...  ...           ...           ...                ...
117   3/27/2014    5         103.0           8.0                6.0
118   3/26/2014    4          86.0           8.0                6.0
119   3/25/2014    3          86.0           8.0                6.0
120   3/24/2014    2          86.0           8.0                6.0
121   3/22/2014    0          49.0           8.0                6.0

[122 rows x 5 columns]
```

Jeżeli kolumna zaczyna się od brakującej wartości, te dane pozostaną z taką wartością, gdyż nie ma żadnej wcześniejszej wartości, która może posłużyć do wypełnienia.

9.3.2.3. Wypełnianie wstecz

Biblioteka Pandas umożliwia też wypełnianie danych wstecz. W tym przypadku najnowsza wartość (w kierunku od góry do dołu) jest stosowana do zastąpienia brakującej wartości. W ten sposób brakujące wartości są zastępowane najnowszą wartością.

```
print(ebola.fillna(method='bfill').iloc[:, 0:5])
```

```
           Date  Day Cases_Guinea Cases_Liberia Cases_SierraLeone
0      1/5/2015  289       2776.0        8166.0           10030.0
```


Kup książkę Poleć książkę

```
1      1/4/2015  288       2775.0        8166.0           9780.0
2      1/3/2015  287       2769.0        8166.0           9722.0
3      1/2/2015  286       2730.0        8157.0           9633.0
4    12/31/2014  284       2730.0        8115.0           9633.0
..          ...  ...          ...           ...              ...
117    3/27/2014    5        103.0           8.0              6.0
118    3/26/2014    4         86.0           NaN              NaN
119    3/25/2014    3         86.0           NaN              NaN
120    3/24/2014    2         86.0           NaN              NaN
121    3/22/2014    0         49.0           NaN              NaN

[122 rows x 5 columns]
```

Jeżeli kolumna jest zakończona brakującą wartością, pozostanie w takiej postaci, ponieważ nie ma żadnej nowej wartości, która mogłaby zostać użyta do wypełnienia.

9.3.2.4. Interpolacja

Interpolacja polega na zastosowaniu istniejących wartości do wypełniania wartości brakujących. Dostępnych jest wiele sposobów takiego wypełniania. W przypadku biblioteki Pandas w ramach interpolacji brakujące wartości są wypełniane w sposób liniowy. Dokładniej rzecz ujmując, brakujące wartości są tutaj traktowane tak, jakby miały być rozmieszczone w równych odstępach od siebie.

```
print(ebola.interpolate().iloc[:, 0:5])
```

```
           Date  Day   Cases_Guinea   Cases_Liberia Cases_SierraLeone
0      1/5/2015  289         2776.0             NaN           10030.0
1      1/4/2015  288         2775.0             NaN            9780.0
2      1/3/2015  287         2769.0          8166.0            9722.0
3      1/2/2015  286         2749.5          8157.0            9677.5
4    12/31/2014  284         2730.0          8115.0            9633.0
..          ...  ...            ...             ...               ...
117    3/27/2014    5          103.0             8.0               6.0
118    3/26/2014    4           86.0             8.0               6.0
119    3/25/2014    3           86.0             8.0               6.0
120    3/24/2014    2           86.0             8.0               6.0
121    3/22/2014    0           49.0             8.0               6.0

[122 rows x 5 columns]
```

Zauważ, że operacja przebiega tutaj trochę w stylu wypełniania do przodu, ale zamiast przekazywania ostatniej znanej wartości do wypełniania służą różnice między wartościami.

Metoda `.interpolate()` oferuje parametr `method`, który umożliwia zmianę metody interpolacji[1]. W tabeli 9.1 zebrano wszystkie możliwe wartości, które były dostępne w czasie, gdy pisano książkę.

[1] Dokumentacja metody `.interpolate()`: *https://pandas.pydata.org/docs/reference/api/pandas.Series.interpolate.html*.

Kup książkę Poleć książkę

Tabela 9.1. *Możliwe wartości (w czasie tworzenia książki), które są przekazywane metodzie .interpolate() w ramach parametru method*

	Technika	Opis
1	`linear`	Umożliwia zignorowanie indeksu i potraktowanie wartości jako rozmieszczonych w równych odstępach. Jest to jedyna metoda obsługiwana w przypadku obiektów `MultiIndex`.
2	`time`	Pozwala przetwarzać dane z każdego dnia oraz dane o większej dokładności w celu przeprowadzenia interpolacji interwału o danej długości.
3	`index, values`	Używane są faktyczne wartości liczbowe indeksu.
4	`pad`	Wartości `NaN` są wypełniane za pomocą istniejących wartości.
5	`nearest, zero, slinear, quadratic, cubic, spline, barycentric, polynomial`	Metody są przekazywane funkcji `scipy.interpolate.interp1d`. Metody te używają wartości liczbowych indeksu.
6	`krogh, piecewise_polynomial, spline, pchip, akima, cubicspline`	Są to elementy opakowujące metody interpolacji o podobnych nazwach zawarte w bibliotece SciPy.
7	`from_derivatives`	Odwołuje się do funkcji `scipy.interpolate.BPoly`.

9.3.2.5. Pomijanie brakujących wartości

Ostatnim wariantem przetwarzania danych z brakującymi wartościami jest pomijanie obserwacji lub zmiennych z takimi danymi. Zależnie od tego, ile danych brakuje, zachowywanie danych tylko kompletnych przypadków może sprawić, że będziesz dysponować bezwartościowym zbiorem danych. Być może brakujące dane nie cechują się losowością, dlatego usunięcie brakujących wartości spowoduje uzyskanie nieobiektywnego zbioru danych. Ewentualnie utrzymanie wyłącznie kompletnych danych sprawi, że pozostaniesz z danymi niewystarczającymi do przeprowadzenia analizy.

Aby usunąć brakujące dane, można użyć metody `.dropna()` i określić jej parametry kontrolujące sposób eliminowania danych. Przykładowo parametr `how` umożliwia zdecydowanie, czy wiersz (lub kolumna) jest usuwany, gdy brakuje danych reprezentowanych przez wartość `'any'` lub `'all'`. Z kolei parametr `thresh` pozwala ustalić, ile wartości innych niż `NaN` może istnieć przed usunięciem wiersza lub kolumny.

```
print(ebola.shape)
```

```
(122, 18)
```

Jeśli w przypadku naszego zbioru danych Ebola zachowamy jedynie kompletne przypadki, pozostaniemy z tylko jednym wierszem danych.

```
ebola_dropna = ebola.dropna()
print(ebola_dropna.shape)
```

```
(1, 18)
```

```
print(ebola_dropna)
```


Kup książkę Poleć książkę

```
           Date  Day  Cases_Guinea  Cases_Liberia  Cases_SierraLeone  \
19   11/18/2014  241        2047.0         7082.0             6190.0

     Cases_Nigeria  Cases_Senegal  Cases_UnitedStates  Cases_Spain  \
19            20.0            1.0                 4.0          1.0

     Cases_Mali  Deaths_Guinea  Deaths_Liberia  Deaths_SierraLeone  \
19          6.0         1214.0          2963.0              1267.0

     Deaths_Nigeria  Deaths_Senegal  Deaths_UnitedStates  \
19              8.0             0.0                  1.0

     Deaths_Spain  Deaths_Mali
19            0.0          6.0
```

9.3.3. Obliczenia uwzględniające brakujące dane

Załóżmy, że chcielibyśmy przyjrzeć się ilości przypadków dla wielu regionów. Można dodać do siebie wiele regionów w celu uzyskania nowej kolumny zawierającej te ilości.

```
ebola["Cases_multiple"] = (
  ebola["Cases_Guinea"]
  + ebola["Cases_Liberia"]
  + ebola["Cases_SierraLeone"]
)
```

Przeanalizujmy pierwszych 10 wierszy wyników obliczeń.

```
ebola_subset = ebola.loc[
    :,
    [
        "Cases_Guinea",
        "Cases_Liberia",
        "Cases_SierraLeone",
        "Cases_multiple",
    ],
]
print(ebola_subset.head(n=10))
```

```
    Cases_Guinea  Cases_Liberia  Cases_SierraLeone  Cases_multiple
0         2776.0            NaN            10030.0            NaN
1         2775.0            NaN             9780.0            NaN
2         2769.0         8166.0             9722.0        20657.0
3            NaN         8157.0                NaN            NaN
4         2730.0         8115.0             9633.0        20478.0
5         2706.0         8018.0             9446.0        20170.0
6         2695.0            NaN             9409.0            NaN
7         2630.0         7977.0             9203.0        19810.0
8         2597.0            NaN             9004.0            NaN
9         2571.0         7862.0             8939.0        19372.0
```

Możesz zauważyć, że wartość kolumny `Cases_multiple` została obliczona tylko wtedy, gdy nie było brakującej wartości w przypadku kolumn `Cases_Guinea`, `Cases_Liberia` i `Cases_SierraLeone`. Obliczenia obejmujące brakujące wartości będą zwykle zwracać brakującą wartość, chyba że wywołana funkcja lub metoda oferuje w obrębie swoich obliczeń możliwość zignorowania brakujących wartości.

Kup książkę Poleć książkę

`.mean()` i `.sum()` to przykłady wbudowanych metod, które mogą ignorować brakujące wartości. Metody te będą zwykle zapewniać parametr `skipna`, sprawiający, że wartość będzie nadal obliczana z pominięciem brakujących wartości.

```
# Domyślna wartość True powoduje pominięcie brakujących wartości
print(ebola.Cases_Guinea.sum(skipna = True))
```

```
84729.0
```

```
print(ebola.Cases_Guinea.sum(skipna = False))
```

```
nan
```

9.4. Brakująca wartość NA wbudowana w bibliotece Pandas

W wersji 1.0 biblioteki Pandas wprowadzono wbudowaną wartość `NA` (`pd.NA`). Gdy pisano książkę, funkcjonalność ta nadal była w fazie eksperymentalnej[2]. Jej głównym celem jest zapewnienie brakującej wartości, która może być używana dla różnych typów danych.

Skorzystajmy z zastosowanego wcześniej zbioru danych `scientists` i sprawdźmy typy danych za pomocą atrybutu `.dtypes`.

```
scientists = pd.DataFrame(
  {
    "Name": ["Rosaline Franklin", "William Gosset"],
    "Occupation": ["Chemist", "Statistician"],
    "Born": ["1920-07-25", "1876-06-13"],
    "Died": ["1958-04-16", "1937-10-16"],
    "Age": [37, 61]
  }
)
```

```
print(scientists)
```

```
                Name    Occupation        Born        Died    Age
0  Rosaline Franklin       Chemist  1920-07-25  1958-04-16     37
1     William Gosset  Statistician  1876-06-13  1937-10-16     61
```

```
print(scientists.dtypes)
```

```
Name           object
Occupation     object
Born           object
Died           object
Age             int64
dtype: object
```

```
scientists.loc[1, "Name"] = pd.NA
scientists.loc[1, "Age"] = pd.NA
```

[2] Eksperymentalna wartość `NA` biblioteki Pandas: *https://pandas.pydata.org/docs/user_guide/missing_data.html#experimental-na-scalar-to-denote-missing-values*.

Kup książkę Poleć książkę

```
print(scientists)

                 Name    Occupation        Born        Died   Age
0   Rosaline Franklin       Chemist  1920-07-25  1958-04-16    37
1                <NA>  Statistician  1876-06-13  1937-10-16  <NA>

print(scientists.dtypes)

Name           object
Occupation     object
Born           object
Died           object
Age            object
dtype: object
```

Porównaj wyniki użycia atrybutu `.dtypes` w przypadku wartości `pd.NA` oraz wartości `np.NaN` podanej wcześniej w niniejszym rozdziale.

```
scientists = pd.DataFrame(
  {
    "Name": ["Rosaline Franklin", "William Gosset"],
    "Occupation": ["Chemist", "Statistician"],
    "Born": ["1920-07-25", "1876-06-13"],
    "Died": ["1958-04-16", "1937-10-16"],
    "Age": [37, 61]
  }
)

scientists.loc[1, "Name"] = np.NaN
scientists.loc[1, "Age"] = np.NaN

print(scientists.dtypes)

Name            object
Occupation      object
Born            object
Died            object
Age            float64
dtype: object
```

Ponieważ wartość `pd.NA` nadal jest w fazie eksperymentalnej, najlepiej sprawdzać w oficjalnej dokumentacji aktualne informacje o jej działaniu.

Podsumowanie

Rzadko się zdarza, żeby zbiór danych w ogóle był pozbawiony brakujących wartości. Ważne jest, aby wiedzieć, jak postępować z takimi wartościami, ponieważ nawet wtedy, gdy zajmujesz się kompletnymi danymi, brakujące wartości nadal mogą się pojawić w wyniku przeprowadzonego przez Ciebie procesu ujednolicania danych. W tym rozdziale zaprezentowane zostały niektóre z podstawowych metod stosowanych w procesie analizy danych, które mają związek z poprawnością danych. Po przyjrzeniu się swoim danym i ujęciu w tabelach brakujących wartości możesz rozpocząć proces oceniania, czy dane są wystarczająco wysokiej jakości, aby móc na ich podstawie podejmować decyzje i wyciągać wnioski.

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Skorowidz

A

agregacja, 54, 195, 196
  funkcje, 199, 202
  metody wbudowane, 198
aktualizowanie pakietów, 395
algorytm KMeans, 352
analiza
  PCA, 353
  przeżycia, 323
    dane, 324
  wariancji, ANOVA, 336
anatomia rysunku, 99
ANOVA, 336
API, Application Programming Interface, 298
argument słowa kluczowego, 158
asercje, 189
aspekty, 124
  tworzenie, 127
atrybut
  .iloc[], 45, 54
  .loc[], 43, 45, 54, 63
atrybuty obiektu Series, 65
automatyczne wyrównywanie, 69, 71, 73

B

biblioteka
  Arrow, 294
  cython, 366
  Dask, 367
  datetime, 265
  matplotlib, 94, 100, 134
  Numba, 163, 366
  NumPy, 66, 116, 162
  openpyxl, 85
  Pandas, 35
  pandas-datareader, 277
  Pandera, 368
  Polars, 367
  PyJanitor, 368
  regex, 263
  scikit-learn, 299, 302, 315
  scipy, 361
  seaborn, 105, 129
  Siuba, 367
  sklearn, 314, 439
  statsmodels, 298, 301, 303, 312, 317
  xarray, 366
biblioteki
  importowanie, 396
binarna zmienna wyjściowa, 310
błąd
  AttributeError, 300
  DeprecationWarning, 300
  ValueError, 300
brakujące dane, 206, 223, 224
  ignorowanie, 234
  interpolacja, 231
  określanie ilości, 228
  pomijanie, 232
  wypełnianie do przodu, 230
  wypełnianie wstecz, 230
  zastępowanie, 229
  znajdowanie, 228

Kup książkę Poleć książkę

C

CAS, Computer Algebra System, 366
class, 334
CSV, Comma Separated Values, 84
czas wykonywania kodu, 366, 428

D

dane
  analizy przeżycia, 324
  dwuzmienne, 110
  kategorialne, 242
    metody, 243
  tekstowe, 245
  uporządkowane, tidy data, 141
  wielozmienne, 103, 120
darmowe zasoby, 372, 425
DataFrame
  części obiektu, 72
  części przeglądowe, 171
  modyfikowanie obiektów, 75
  tworzenie obiektu, 63
  wartości boolowskie, 73
  zastosowanie funkcji, 158
daty i godziny, 265
  biblioteka Arrow, 294
  częstotliwości, 283
  ładowanie danych, 269
  obliczenia, 273
  określanie podzbioru danych, 278
  ponowne próbkowanie, 290
  przesunięcia, 284
  składniki daty, 270
  strefy czasowe, 292
  wartości przesuwające, 284
  zakresy dat, 281
diagnostyka modeli, 329
dodawanie
  kolumn, 75, 174
  wierszy, 171
dokumentacja biblioteki seaborn, 133
dopasowanie wzorca, 257
dwukropek, 50
dwuzmienność, 101
dyspersja, 319, 446

E

edytor tekstu, 385
eksportowanie danych, 81
estymacja jądrowa gęstości, 107
etykiety indeksu, 306
Excel, 84

F

filtrowanie, 195, 208
format danych
  CSV, 84
  Feather, 86
  JSON, 88
formatowanie
  liczb, 254, 431
  łańcuchów, 253, 430
funkcja, 155
  .cut(), 436
  .head(), 287
  .merge(), 184
  .reindex(), 282, 287
  .tail(), 287
  .to_numeric(), 239
  compile(), 262
  concat(), 171
  count_non zero(), 228
  countplot(), 318, 320
  DataFrame.plot.box(), 139
  DataFrame.plot.hexbin(), 138
  DataFrame.plot.kde(), 137
  DataFrame.plot.scatter(), 137
  date_range(), 282, 284
  findall(), 261
  get_dummies(), 305
  glm(), 318
  isnull(), 222
  KaplanMeierFitter(), 324, 325
  lambda, 80
  LogisticRegression(), 314
  logit(), 312
  mean(), 199
  notnull(), 222
  np.mean(), 116
  numpy.mean(), 397
  ols(), 298

Kup książkę Poleć książkę

pd.DataFrame.from_dict(), 88
pd.read_csv(), 84
pd.read_pickle(), 83
Pipeline(), 308
poisson(), 317
print(), 39, 41, 42
range(), 48, 50, 414
read_csv(), 36
scipy.mean(), 397
Series.plot.hist(), 136
sns.boxplot(), 117
sns.displot(), 108
sns.jointplot(), 112
sns.kdeplot(), 114, 115
sns.lmplot(), 112
sns.pairgrid(), 119
sns.pairplot(), 119
sns.regplot(), 111
sns.scatterplot(), 123
sns.violinplot(), 120
to_datetime(), 266, 268
to_numeric(), 240
type(), 148, 179
vectorize(), 429
funkcje, 154, 410
agregacji, 199, 202
anonimowe, 164
lambda, 164
parametry arbitralne, 412
parametry domyślne, 412
wektoryzowane, 161, 162
wyrażenie **kwargs, 413
wyrażenie *args, 413
wyrażeń regularnych, 256
zastosowanie
względem obiektu DataFrame, 158
względem obiektu Series, 157

G

generatory, 414
GLM, Generalized Linear Models, 310
grupowanie, 54, 195, 210, 217
grupy, 209
iteracja, 211
wybieranie, 211

H

histogram, 101, 106, 136
residuów, 333

I

IDE, Integrated Development Environment, 387
ilościowa zmienna wynikowa, 317
importowanie
bibliotek, 396
danych, 81
indeksowanie wierszy, 43
instalacja
języka Python, 378
pakietów, 380, 394
instrukcje warunkowe, 433
interfejs biblioteki seaborn, 135
interpolacja, 231
IPython, 386

J

jednostki obserwacyjne, 178, 190
jednozmienność, 101, 106
JSON, JavaScript Object Notation, 88
Jupyter, 386

K

katalogi robocze, 388
klasteryzacja, 351
klastrowanie
hierarchiczne, 357
ręczne ustawianie progu, 361
kompletne, 358
metodą Warda, 360
pojedyncze, 358
z centroidem, 360
ze średnią, 359
klasy, 420
kodowanie One-Hot, 307
kolory, 120
kolumny
dodawanie, 147, 174
dzielenie, 147, 149
konkatenacja, 170, 174

Kup książkę Poleć książkę

kolumny
łączenie, 149
utrwalenie, 145
utrwalenie pojedyncze, 142
zawierające wiele zmiennych, 146
zmienne, 150
kompozycje, 129
komunikat SettingWithCopyWarning, 422
konkatenacja, 170
kolumn, 174
z różnymi wierszami, 177
wierszy, 171
z różnymi kolumnami, 175
konteksty wykresów, 132
kontenery, 400
konwersja danych, 237
krotka, 401
kryterium informacyjne
Akaikego, 336
Bayesowskie, 336
krzywe Kaplana-Meiera, 324
k-średnie, 351

L

lambda, 164
LASSO, 345
lista, 400
składana, 182, 408
LOO, Leave-One-Out, 339

Ł

ładowanie
danych, 223, 269
wielu plików
lista składana, 182
pętla, 180
łańcuchy, 245
dzielenie, 246
formatowanie, 253, 430
metod, 426
ostatni znak, 247
wycinki, 248
wycinki przyrostowe, 249
łączenie
danych, 169
zbiorów danych, 169

M

mapy pojęć, 375
matplotlib, 94
anatomia rysunku, 99
dane wielozmienne, 103
dwuzmienność, 101
jednozmienność, 101
podwykresy, 95
wizualizacja danych statystycznych, 100
metoda
.agg(), 199, 202, 204
.aggregate(), 202, 209
.apply(), 76, 154, 156, 159–161, 164, 287
.assign(), 79, 80
.astype(), 238, 240
.copy(), 239
.describe(), 67, 198
.drop(), 81
.fillna(), 207, 229
.filter(), 208, 209
.first_valid_index(), 287
.format(), 431
.get(), 402
.groupby(), 55, 195, 198, 207–210, 213
.head(), 39
.info(), 303
.interpolate(), 206, 231
.isnull(), 228
.items(), 403
.join(), 184, 251
.nunique(), 58
.print_summary(), 326
.reindex(), 227
.reset_index(), 57, 214
.save(), 82
.set_xlabel(), 99
.shift(), 288
.split(), 147
.splitlines(), 251
.str.split(), 149
.sum(), 229
.summary(), 298
.tail(), 45
.to_csv(), 84
.to_dict(), 87
.to_excel(), 85
.to_feather(), 86
.to_json(), 88

Kup książkę Poleć książkę

.to_pickle, 82
.transform(), 209
.value_counts(), 58, 229
count(), 228
fig.set_tight_layout(), 97
fig.tight_layout(), 97
match(), 262
Path.glob(), 181
plt.plot(), 94
sns.histplot(), 106
str.replace(), 261
strftime(), 268
strptime(), 268
metody
agregacji, 198
danych kategorialnych, 243
eksportujące obiekty DataFrame, 91
łańcuchowe, 249–251
obiektu datetime, 274
obiektu Series, 65–67
sekwencyjne wywoływanie, 426
model proporcjonalnego hazardu Coxa, 326
testowanie założeń, 327
modele
diagnostyka, 329
GLM, 310, 321
liniowe, 334
liniowe uogólnione, 310, 321, 337
residua, 329
moduł concurrent.futures, 366
modyfikowanie
kolumn, 76, 79
podzbioru danych, 422

N

nawiasy
klamrowe, 402
kwadratowe, 39, 400
okrągłe, 77, 260
normalizacja, 190
notacja z kropką, 42
notowania giełdowe, 277
Numba
wektoryzacja, 163
NumPy, 66
wektoryzacja, 162

O

obiekt
DataFrame, 63, 72
datetime, 265
metody, 274
DatetimeIndex, 279
MultiIndex, 214
numpy.ndarray, 66
Series, 62, 63
timedelta, 273
TimedeltaIndex, 280
obiekty biblioteki
matplotlib, 134, 135
seaborn, 135
określanie podzbioru wierszy, 43, 45, 53
operator *, 334

P

pakiety
aktualizowanie, 395
instalowanie, 394
Pandas
DataFrame, 35
struktury danych, 61
tworzenie wykresów, 136
typy danych, 38, 91
panele kontrolne, 369
parametr inplace, 79
PCA, Principal Component Analysis, 353
peklowanie, pickle, 82
pętla, 180, 406
Pipenv, 392
pliki
.csv, 36, 84
.p, .pickle, .pkl, 82
Feather, 86, 87
JSON, 88
podwykresy, 95
ponowne
indeksowanie, 226
kodowanie, 229
próbkowanie, 290
potok, 307
Pipeline(), 308
potrójne apostrofy, 156

Kup książkę Poleć książkę

program
  cProfile, 366
  MATLAB, 361
  snakevis, 366
projekt Apache Arrow, 87
przekształcanie typów, 237
  do postaci obiektów łańcuchów, 237
  do postaci ramki danych, 266
  w kategorię, 242
  w wartości liczbowe, 238
przypisanie wielokrotne, 416
publikowanie, 368
pusty łańcuch, 222
Pyenv, 392
Python, 386

R

ramka danych scientists, 65
regresja
  dwumianowa ujemna, 319, 446
  grzbietowa, 346
  LASSO, 345
  liniowa, 297, 443
  logistyczna, 310, 435, 445
  Poissona, 317, 446
  wielokrotna, 301
regularyzacja, 343
relacje par, 119
replikowanie wyników, 442
residua, 329
rozgłaszanie, 69, 73
rysunki, 95, 99

S

scalanie
  typu „jedna z jedną”, 185
  typu „wiele z jedną”, 186
  typu „wiele z wieloma”, 187
  zbiorów danych, 183
seaborn, 105
  aspekty, 124
  dane dwuzmienne, 110
  dane wielozmienne, 120
  dokumentacja biblioteki, 133
  interfejs, 135
  jednozmienność, 106
  kompozycje, 129
  style, 129
Series
  atrybuty, 65
  brakujące wartości, 225
  metody, 66, 67
  modyfikowanie obiektów, 75
  tworzenie obiektu, 62
  wartości boolowskie, 67
  zastosowanie funkcji, 157
sieć elastyczna, 347
skalary, 70
składnia wycinków, 50
słownik, 87, 202, 402
słowo kluczowe class, 334
sprawdzanie kolumn, 39
strefy czasowe, 292
struktury danych biblioteki Pandas, 61
styl kodu, 398
style, 129, 130
szablony projektowe, 384

Ś

średnie grupowane, 55
środowiska systemu conda, 390

T

tablice, 418
transformacja, 195, 204
transformacje, 333
tworzenie
  aspektów, 127
  obiektu DataFrame, 63
  obiektu Series, 62
  wizualizacji danych statystycznych, 100
  wykresów, 93
    biblioteka Pandas, 136
typ danych
  DataFrame, 35
  ndarray, 65, 418
  Series, 62
typy danych, 236
  biblioteki Pandas, 38
  przekształcanie, 237
  wyjściowych, 91

Kup książkę Poleć książkę

U

uczenie maszynowe, 368
usuwanie kolumn, 81

W

walidacja krzyżowa, 349
  k-krotna, 339
wartości brakujące, *Patrz* brakujące dane
wartość
  NA, 234
  NaN, 142, 221
  NaT, 287
wektory
  o różnej długości, 70
  o tej samej długości, 69
  z liczbami całkowitymi, 70
  ze wspólnymi etykietami indeksu, 71
wektoryzacja
  biblioteka Numba, 163
  biblioteka NumPy, 162
wiersz poleceń, 382, 385
wiersze
  dodawanie, 171
wybieranie kolumn, 39, 47
wycinki, 50, 248, 249, 404
wydajność, 366
wykres, 58
  dywanikowy, 107
  gęstości, 107, 137
  gęstości dwuwymiarowy, 114
  ilościowy, 109
  k-średnich, 355, 357
  kwantylowy K-K, 332
  łączony, 112
  przedziałów sześciokątnych, 113, 138
  pudełkowy, 102, 117, 139
  punktowy, 101, 104, 110, 137
  skrzypcowy, 117
  słupkowy, 109, 116
wykresy
  rozkładu, 108
  tworzenie, 93
    biblioteka matplotlib, 94, 100
    biblioteka seaborn, 105
wynik standardowy z, 204
wyrażenia regularne, 255
  dopasowanie wzorca, 257
  kompilowanie wzorca, 262
  zastępowanie wzorca, 261
  znajdowanie wzorca, 261
wyrażenie
  **kwargs, 413
  *args, 413

Z

zakresy, 414
zastępowanie wartości, 424
zbiory danych, 36
  dzielenie, 178
  łączenie, 169
  pobieranie, 380
  scalanie, 183
zbiór danych, 36
  ACS, 317, 435
  Anscombe'a, 95
  bladder, 324
  Ebola, 266
  Gapminder, 36
  tips, 236
  titanic, 311
zintegrowane środowisko programistyczne, IDE, 387
zmienne
  dwuaspektowe, 126
  fikcyjne, 305
  jednoaspektowe, 125
  kategorialne, 303
    w bibliotece scikit-learn, 305
    kategorialne w bibliotece statsmodels, 303
znak tyldy (~), 298
znaki
  podziału wiersza, 398
  specjalne wyrażeń regularnych, 256

Kup książkę Poleć książkę

Kup książkę Poleć książkę


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

ANALIZUJ ZBIORY DANYCH I ODKRYWAJ UKRYTĄ W NICH WIEDZĘ!

Wprawny analityk potrafi się posługiwać zbiorami danych o wysokiej dynamice i różnorodności. Działanie to ułatwia biblioteka *open source* Pandas, która pozwala, przy użyciu języka Python, zrealizować niemal każde zadanie wymagające analizy danych. Pandas może pomóc w zapewnieniu wiarygodności danych, wizualizowaniu ich pod kątem efektywnego podejmowania decyzji i analizowaniu wielu zbiorów danych.

Oto drugie, zaktualizowane i uzupełnione wydanie przewodnika po bibliotece Pandas. Dzięki tej przystępnej książce nauczysz się w pełni korzystać z możliwości oferowanych przez bibliotekę, nawet jeśli dopiero zaczynasz przygodę z analizą danych w Pythonie. Naukę rozpoczniesz z użyciem rzeczywistego zbioru danych, aby wkrótce rozwiązywać złożone problemy danologii, takie jak obsługa brakujących danych, stosowanie regularyzacji czy też używanie metod nienadzorowanego uczenia maszynowego do odnajdywania podstawowej struktury w zbiorze danych. Pracę z poszczególnymi zagadnieniami ułatwia to, że zostały one zilustrowane prostymi, ale praktycznymi przykładami.

W książce:

- importowanie i eksportowanie danych, przygotowywanie ich zbiorów
- tworzenie wykresów za pomocą bibliotek *matplotlib*, *seaborn* i Pandas
- konwersja typów danych
- skalowanie operacji przetwarzania danych
- zaawansowane możliwości biblioteki Pandas powiązane z datami i czasem
- dopasowywanie modeli liniowych przy użyciu bibliotek *statsmodels* i *scikit-learn*

Dr DANIEL Y. CHEN jest wykładowcą na University of British Columbia. Prowadzi też zajęcia edukacyjne z zakresu danologii w firmie RStudio PBC. Współpracował z organizacją The Carpentries jako instruktor, prowadzący szkolenia, opiekun materiałów lekcyjnych i kierownik odpowiedzialny za utrzymanie społeczności.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-289-0151-3

Cena: 109,00 zł

Pearson
Addison-Wesley