



Uczenie maszynowe w Pythonie Receptury

Tytuł oryginału: Machine Learning with Python Cookbook:
Practical Solutions from Preprocessing to Deep Learning

Tłumaczenie: Robert Górczyński

ISBN: 978-83-283-5046-5

© 2019 Helion S.A.

Authorized Polish translation of the English edition of Machine Learning with Python Cookbook ISBN 9781491989388 © 2018 Chris Albon

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

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 the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek 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 Helion SA 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 Helion SA nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
<ftp://ftp.helion.pl/przyklady/uczmar.zip>

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

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Wprowadzenie	11
1. Wektor, macierz i tablica	15
1.0. Wprowadzenie	15
1.1. Tworzenie wektora	15
1.2. Tworzenie macierzy	16
1.3. Tworzenie macierzy rzadkiej	17
1.4. Pobieranie elementów	18
1.5. Opisywanie macierzy	20
1.6. Przeprowadzanie operacji na elementach	20
1.7. Znajdowanie wartości maksymalnej i minimalnej	21
1.8. Obliczanie średniej, wariancji i odchylenia standardowego	22
1.9. Zmiana kształtu tablicy	23
1.10. Transponowanie wektora lub macierzy	24
1.11. Spłaszczanie macierzy	25
1.12. Znajdowanie rzędu macierzy	25
1.13. Obliczanie wyznacznika macierzy	26
1.14. Pobieranie przekątnej macierzy	27
1.15. Obliczanie śladu macierzy	27
1.16. Znajdowanie wektorów i wartości własnych	28
1.17. Obliczanie iloczynu skalarnego	29
1.18. Dodawanie i odejmowanie macierzy	30
1.19. Mnożenie macierzy	31
1.20. Odwracanie macierzy	32
1.21. Generowanie liczb losowych	33

2. Wczytywanie danych	35
2.0. Wprowadzenie	35
2.1. Wczytywanie przykładowego zbioru danych	35
2.2. Tworzenie symulowanego zbioru danych	36
2.3. Wczytywanie pliku CSV	39
2.4. Wczytywanie pliku Excela	40
2.5. Wczytywanie pliku JSON	41
2.6. Wykonywanie zapytań do bazy danych SQL	42
3. Przygotowywanie danych	45
3.0. Wprowadzenie	45
3.1. Tworzenie ramki danych	46
3.2. Opisywanie danych	47
3.3. Poruszanie się po ramce danych	49
3.4. Pobieranie wierszy na podstawie pewnych warunków	51
3.5. Zastępowanie wartości	52
3.6. Zmiana nazwy kolumny	53
3.7. Znajdowanie wartości minimalnej, maksymalnej, sumy, średniej i liczby elementów w kolumnie	54
3.8. Znajdowanie unikatowych wartości	55
3.9. Obsługa brakujących wartości	56
3.10. Usuwanie kolumn	58
3.11. Usuwanie wiersza	59
3.12. Usuwanie powielonych wierszy	60
3.13. Grupowanie wierszy	62
3.14. Grupowanie wierszy według czasu	63
3.15. Iterowanie przez kolumnę	65
3.16. Wywoływanie funkcji dla wszystkich elementów kolumny	66
3.17. Wywoływanie funkcji dla grupy	67
3.18. Konkatenacja obiektów typu DataFrame	68
3.19. Złączanie obiektów typu DataFrame	69
4. Obsługa danych liczbowych	73
4.0. Wprowadzenie	73
4.1. Przeskalowywanie cechy	73
4.2. Standaryzowanie cechy	74
4.3. Normalizowanie obserwacji	76
4.4. Generowanie cech wielomianowych i interakcji	78
4.5. Transformacja cech	79

4.6. Wykrywanie elementów odstających	80
4.7. Obsługa elementów odstających	82
4.8. Dyskretyzacja cech	84
4.9. Grupowanie obserwacji przy użyciu klastra	85
4.10. Usuwanie obserwacji, w których brakuje wartości	87
4.11. Uzupełnianie brakujących wartości	88
5. Obsługa danych kategoryzujących	91
5.0. Wprowadzenie	91
5.1. Kodowanie nominalnych cech kategoryzujących	92
5.2. Kodowanie porządkowych cech kategoryzujących	94
5.3. Kodowanie słowników cech	96
5.4. Wstawianie brakujących wartości klas	98
5.5. Obsługa niezrównoważonych klas	99
6. Obsługa tekstu	103
6.0. Wprowadzenie	103
6.1. Oczyszczanie tekstu	103
6.2. Przetwarzanie i oczyszczanie danych HTML	105
6.3. Usuwanie znaku przestankowego	105
6.4. Tokenizacja tekstu	106
6.5. Usuwanie słów o małym znaczeniu	107
6.6. Stemming słów	108
6.7. Oznaczanie części mowy	109
6.8. Kodowanie tekstu za pomocą modelu worka słów	111
6.9. Określanie wagi słów	113
7. Obsługa daty i godziny	117
7.0. Wprowadzenie	117
7.1. Konwertowanie ciągu tekstowego na datę	117
7.2. Obsługa stref czasowych	118
7.3. Pobieranie daty i godziny	120
7.4. Podział danych daty na wiele cech	121
7.5. Obliczanie różnicy między datami	122
7.6. Kodowanie dni tygodnia	123
7.7. Tworzenie cechy opóźnionej w czasie	124
7.8. Użycie okien wpływającego czasu	125
7.9. Obsługa brakujących danych w serii danych zawierających wartości daty i godziny	126

8. Obsługa obrazów	129
8.0. Wprowadzenie	129
8.1. Wczytywanie obrazu	129
8.2. Zapisywanie obrazu	132
8.3. Zmiana wielkości obrazu	133
8.4. Kadrowanie obrazu	134
8.5. Rozmywanie obrazu	135
8.6. Wyostrażanie obrazu	138
8.7. Zwiększanie kontrastu	138
8.8. Izolowanie kolorów	141
8.9. Progowanie obrazu	142
8.10. Usuwanie tła obrazu	145
8.11. Wykrywanie krawędzi	147
8.12. Wykrywanie narożników w obrazie	150
8.13. Tworzenie cech w uczeniu maszynowym	153
8.14. Użycie średniej koloru jako cechy	155
8.15. Użycie histogramu koloru jako cechy	157
9. Redukowanie wymiarowości za pomocą wyodrębniania cech	161
9.0. Wprowadzenie	161
9.1. Redukowanie cech za pomocą głównych składowych	161
9.2. Redukowanie cech, gdy dane są liniowo nierozłączne	164
9.3. Redukowanie cech przez maksymalizację rozłączności klas	166
9.4. Redukowanie cech za pomocą rozkładu macierzy	169
9.5. Redukowanie cech w rzadkich danych	170
10. Redukcja wymiarowości za pomocą wyboru cech	173
10.0. Wprowadzenie	173
10.1. Progowanie wariancji cechy liczbowej	173
10.2. Progowanie wariancji cechy binarnej	175
10.3. Obsługa wysoce skorelowanych cech	176
10.4. Usuwanie nieistotnych dla klasyfikacji cech	177
10.5. Rekurencyjne eliminowanie cech	179
11. Ocena modelu	183
11.0. Wprowadzenie	183
11.1. Modele sprawdzianu krzyżowego	183
11.2. Tworzenie modelu regresji bazowej	186
11.3. Tworzenie modelu klasyfikacji bazowej	188

11.4. Ocena prognoz klasyfikatora binarnego	189
11.5. Ocena progowania klasyfikatora binarnego	192
11.6. Ocena prognoz klasyfikatora wieloklasowego	195
11.7. Wizualizacja wydajności klasyfikatora	197
11.8. Ocena modelu regresji	199
11.9. Ocena modelu klasteryzacji	201
11.10. Definiowanie niestandardowych współczynników oceny modelu	202
11.11. Wizualizacja efektu wywieranego przez wielkość zbioru uczącego	204
11.12. Tworzenie raportu tekstowego dotyczącego współczynnika oceny	206
11.13. Wizualizacja efektu wywieranego przez zmianę wartości hiperparametrów	207
12. Wybór modelu	211
12.0. Wprowadzenie	211
12.1. Wybór najlepszych modeli przy użyciu wyczerpującego wyszukiwania	212
12.2. Wybór najlepszych modeli za pomocą przeszukiwania losowego	214
12.3. Wybór najlepszych modeli z wielu algorytmów uczenia maszynowego	216
12.4. Wybór najlepszych modeli na etapie przygotowywania danych	217
12.5. Przyspieszanie wyboru modelu za pomocą równoległości	219
12.6. Przyspieszanie wyboru modelu przy użyciu metod charakterystycznych dla algorytmu	220
12.7. Ocena wydajności po wyborze modelu	221
13. Regresja liniowa	225
13.0. Wprowadzenie	225
13.1. Wyznaczanie linii	225
13.2. Obsługa wpływu interakcji	227
13.3. Wyznaczanie zależności nieliniowej	228
13.4. Redukowanie wariancji za pomocą regularyzacji	230
13.5. Redukowanie cech za pomocą regresji metodą LASSO	232
14. Drzewa i lasy	235
14.0. Wprowadzenie	235
14.1. Trenowanie klasyfikatora drzewa decyzyjnego	235
14.2. Trenowanie regresora drzewa decyzyjnego	237
14.3. Wizualizacja modelu drzewa decyzyjnego	238
14.4. Trenowanie klasyfikatora losowego lasu	240
14.5. Testowanie regresora losowego lasu	241
14.6. Identyfikacja ważnych cech w losowych lasach	242
14.7. Wybór ważnych cech w losowym lesie	245

14.8. Obsługa niezerównoważonych klas	246
14.9. Kontrolowanie wielkości drzewa	247
14.10. Poprawa wydajności za pomocą wzmocnienia	248
14.11. Ocena losowego lasu za pomocą estymatora błędu out-of-bag	250
15. Algorytm k najbliższych sąsiadów	251
15.0. Wprowadzenie	251
15.1. Wyszukiwanie najbliższych sąsiadów obserwacji	251
15.2. Tworzenie klasyfikatora k najbliższych sąsiadów	253
15.3. Ustalanie najlepszej wielkości sąsiedztwa	255
15.4. Tworzenie klasyfikatora najbliższych sąsiadów opartego na promieniu	256
16. Regresja logistyczna	259
16.0. Wprowadzenie	259
16.1. Trenowanie klasyfikatora binarnego	259
16.2. Trenowanie klasyfikatora wieloklasowego	260
16.3. Redukcja wariancji poprzez regularyzację	262
16.4. Trenowanie klasyfikatora na bardzo dużych danych	263
16.5. Obsługa niezerównoważonych klas	264
17. Maszyna wektora nośnego	267
17.0. Wprowadzenie	267
17.1. Trenowanie klasyfikatora liniowego	267
17.2. Obsługa liniowo nierozdzielnych klas przy użyciu funkcji jądra	270
17.3. Określanie prognozowanego prawdopodobieństwa	273
17.4. Identyfikacja wektorów nośnych	275
17.5. Obsługa niezerównoważonych klas	276
18. Naiwny klasyfikator bayesowski	279
18.0. Wprowadzenie	279
18.1. Trenowanie klasyfikatora dla cech ciągłych	280
18.2. Trenowanie klasyfikatora dla cech dyskretnych lub liczebnych	282
18.3. Trenowanie naiwnego klasyfikatora bayesowskiego dla cech binarnych	283
18.4. Kalibrowanie prognozowanego prawdopodobieństwa	284
19. Klasteryzacja	287
19.0. Wprowadzenie	287
19.1. Klasteryzacja za pomocą k średnich	287
19.2. Przyspieszanie klasteryzacji za pomocą k średnich	290

19.3. Klasteryzacja za pomocą algorytmu meanshift	290
19.4. Klasteryzacja za pomocą algorytmu DBSCAN	292
19.5. Klasteryzacja za pomocą łączenia hierarchicznego	293
20. Sieci neuronowe	295
20.0. Wprowadzenie	295
20.1. Przygotowywanie danych dla sieci neuronowej	296
20.2. Projektowanie sieci neuronowej	297
20.3. Trenowanie klasyfikatora binarnego	300
20.4. Trenowanie klasyfikatora wieloklasowego	302
20.5. Trenowanie regresora	304
20.6. Generowanie prognoz	305
20.7. Wizualizacja historii trenowania	307
20.8. Redukcja nadmiernego dopasowania za pomocą regularyzacji wagi	310
20.9. Redukcja nadmiernego dopasowania za pomocą techniki wcześniejszego zakończenia procesu uczenia	311
20.10. Redukcja nadmiernego dopasowania za pomocą techniki porzucenia	313
20.11. Zapisywanie postępu modelu uczącego	315
20.12. K-krotny sprawdzian krzyżowy sieci neuronowej	316
20.13. Dostrajanie sieci neuronowej	318
20.14. Wizualizacja sieci neuronowej	320
20.15. Klasyfikacja obrazów	322
20.16. Poprawa wydajności przez modyfikację obrazu	325
20.17. Klasyfikowanie tekstu	327
21. Zapisywanie i wczytywanie wytrenowanych modeli	331
21.0. Wprowadzenie	331
21.1. Zapisywanie i wczytywanie modelu biblioteki scikit-learn	331
21.2. Zapisywanie i wczytywanie modelu biblioteki Keras	332
Skorowidz	335

Przygotowywanie danych

3.0. Wprowadzenie

Przygotowywanie danych (ang. *data wrangling*) to dość często używane pojęcie, najczęściej w celu opisanego procesu przekształcenia niezmodyfikowanych danych na postać czystego i zorganizowanego formatu informacji gotowych do użycia. Dla nas to tylko jeden — choć zarazem niezwykle ważny — krok na etapie wstępnego przetwarzania danych.

Najczęściej wykorzystywaną strukturą stosowaną do przygotowywania danych jest tzw. ramka danych, która jest intuicyjna w użyciu i jednocześnie niezwykle elastyczna. Ramka danych ma postać tabelaryczną, co oznacza, że została oparta na wierszach i kolumnach, podobnie jak dane przechowywane w arkuszu kalkulacyjnym. Oto przykład ramki danych utworzonej na podstawie informacji o pasażerach Titanica:

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie adresu URL.
url = 'https://tinyurl.com/titanic-csv'

# Umieszczenie wczytanych informacji w ramce danych.
dataframe = pd.read_csv(url)

# Wyświetlenie pierwszych pięciu wierszy.
dataframe.head(5)
```

Oto tabela pokazująca przykładowe dane wejściowe wczytane z pliku CSV:

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.00	female	1	1
1	Allison, Miss Helen Loraine	1st	2.00	female	0	1
2	Allison, Mr Hudson Joshua Creighton	1st	30.00	male	0	0
3	Allison, Mrs Hudson JC (Bessie Waldo Daniels)	1st	25.00	female	0	1
4	Allison, Master Hudson Trevor	1st	0.92	male	1	0

Trzeba zwrócić uwagę na trzy ważne kwestie dotyczące tej ramki danych.

Po pierwsze, każdy wiersz ramki danych odpowiada jednej obserwacji (tutaj jednemu pasażerowi), natomiast każda kolumna przedstawia jedną cechę — płeć, wiek itd. Na podstawie informacji zawartych w pierwszej obserwacji można powiedzieć, że panna Elisabeth Walton Allen podróżowała pierwszą klasą, w chwili katastrofy Titanica miała 29 lat, była kobietą i przeżyła katastrofę.

Po drugie, każda kolumna zawiera tytuł (na przykład `Name`, `PClass` i `Age`), zaś każdy wiersz ma numer indeksu (na przykład 0 dla szczęśliwej panny Elisabeth Walton Allen). Dzięki tym danym można pobierać, a następnie przeprowadzać operacje na obserwacjach i cechach.

Po trzecie, dwie kolumny, `Sex` i `SexCode`, zawierają te same informacje, ale zapisane w różnych formatach. W kolumnie `Sex` płeć kobiety została zapisana jako `female`, natomiast w kolumnie `SexCode` kobieta została oznaczona liczbą całkowitą 1. Ponieważ chcemy, aby wszystkie cechy były unikatowe, trzeba będzie usunąć jedną z tych kolumn.

W rozdziale przedstawię różne techniki przeznaczone do przeprowadzania operacji na ramkach danych za pomocą biblioteki `pandas`. Celem tych działań jest przygotowanie zbioru obserwacji o doskonałej strukturze, gotowego do dalszego wstępnego przetwarzania.

3.1. Tworzenie ramki danych

Problem

Chcesz utworzyć nową ramkę danych.

Rozwiązanie

Biblioteka `pandas` oferuje wiele metod przeznaczonych do tworzenia nowych ramek danych, czyli obiektów typu `DataFrame`. Jedna z nich pozwala na utworzenie pustej ramki danych, w której następnie można zdefiniować poszczególne kolumny.

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie obiektu typu DataFrame.
dataframe = pd.DataFrame()

# Dodanie kolumn.
dataframe['Name'] = ['Jacky Jackson', 'Steven Stevenson']
dataframe['Age'] = [38, 25]
dataframe['Driver'] = [True, False]

# Wyświetlenie obiektu DataFrame.
Dataframe
```

Oto dane po wykonaniu przedstawionego kodu:

	Name	Age	Driver
0	Jacky Jackson	38	True
1	Steven Stevenson	25	False

Po utworzeniu obiektu DataFrame można na jego końcu dodawać nowe dane, jak pokazałem w kolejnym fragmencie kodu.

```
# Utworzenie wiersza.  
new_person = pd.Series(['Molly Mooney', 40, True], index=['Name', 'Age', 'Driver'])  
  
# Dodanie nowego wiersza do istniejących.  
dataframe.append(new_person, ignore_index=True)
```

Oto dane po wykonaniu przedstawionego kodu:

	Name	Age	Driver
0	Jacky Jackson	38	True
1	Steven Stevenson	25	False
2	Molly Mooney	40	True

Analiza

Można odnieść wrażenie, że biblioteka pandas oferuje niemal nieograniczoną liczbę sposobów na tworzenie obiektu typu DataFrame. W rzeczywistości do utworzenia pustego obiektu DataFrame, a następnie wypełnienia go danymi niemal nigdy nie dochodzi. Zamiast tego obiekt jest tworzony na podstawie rzeczywistych danych wczytanych z innego źródła, na przykład pliku CSV lub bazy danych.

3.2. Opisywanie danych

Problem

Chcesz wyświetlić pewne cechy charakterystyczne obiektu DataFrame.

Rozwiązanie

Jedną z najprostszych czynności, którą można wykonać po wczytaniu danych, jest wyświetlenie kilku pierwszych wierszy za pomocą metody `head()`.

```
# Wczytanie biblioteki.  
import pandas as pd  
  
# Utworzenie adresu URL.  
url = 'https://tinyurl.com/titanic-csv'  
  
# Wczytanie danych.  
dataframe = pd.read_csv(url)  
  
# Wyświetlenie dwóch pierwszych wierszy.  
dataframe.head(2)
```

Oto dane po wykonaniu przedstawionego kodu:

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.00	female	1	1
1	Allison, Miss Helen Loraine	1st	2.00	female	0	1

Można też sprawdzić liczbę istniejących wierszy i kolumn.

```
# Wyświetlenie wielkości danych.  
dataframe.shape  
  
(1313, 6)
```

Metoda `describe()` dostarcza opisowe dane statystyczne dotyczące kolumn zawierających dane liczbowe.

```
# Wyświetlenie pewnych danych statystycznych.  
dataframe.describe()
```

Oto dane po wykonaniu przedstawionego polecenia:

	Age	Survived	SexCode
Count	756.000000	1313.000000	1313.000000
Mean	30.397989	0.342727	0.351866
Std	14.259049	0.474802	0.477734
Min	0.170000	0.000000	0.000000
25%	21.000000	0.000000	0.000000
50%	28.000000	0.000000	0.000000
75%	39.000000	1.000000	1.000000
Max	71.000000	1.000000	1.000000

Analiza

Po wczytaniu danych warto zapoznać się z ich strukturą i rodzajem przechowywanych w nich informacji. W idealnym świecie bezpośrednio przeglądane byłyby rzeczywiste dane. Jednak w większości sytuacji te dane mogą się składać z dziesiątek tysięcy, setek tysięcy lub nawet milionów wierszy i kolumn. Dlatego też trzeba pobrać jedynie próbki pozwalające na wyświetlanie niewielkich wycinków i obliczanie podsumowania danych statystycznych dotyczących tych informacji.

W omawianym tutaj rozwiązaniu użyłem niewielkiego zbioru danych przedstawiającego pasażerów Titanica znajdujących się na pokładzie w trakcie jego ostatniego rejsu. Metoda `head()` pozwala na przejrzenie kilku pierwszych (domyślnie pięciu) wierszy danych. Ewentualnie można użyć metody `tail()` do wyświetlenia kilku ostatnich wierszy. Wynikiem działania metody `shape()` jest liczba wierszy i kolumn przechowywanych we wskazanym obiekcie `DataFrame`. Z kolei metoda `describe()` dostarcza wybrane podstawowe dane statystyczne dotyczące wszystkich kolumn zawierających dane liczbowe.

Warto w tym miejscu dodać, że podsumowanie danych statystycznych nie zawsze przedstawia prawdziwy obraz przetwarzanych danych. Na przykład biblioteka pandas traktuje kolumny Survived i SexCode jako liczbowe, ponieważ zawierają one dane w postaci zer i jedynek. Jednak w omawianym przykładzie te wartości liczbowe przedstawiają kategorie. Dlatego też jeśli wartość kolumny Survived wynosi 1, to oznacza, że dany pasażer przeżył katastrofę Titanica. Pewne wygenerowane wartości w podsumowaniu danych statystycznych nie mają więc sensu — przykładem jest tutaj odchylenie standardowe dla kolumny SexCode określającej płeć pasażera.

3.3. Poruszanie się po ramce danych

Problem

Musisz wybrać pojedyncze dane lub wycinek ramki danych.

Rozwiązanie

Do pobrania jednego lub dwóch wierszy wartości użyj metody `loc()` lub `iloc()`.

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie adresu URL.
url = 'https://tinyurl.com//titanic-csv'

# Wczytanie danych.
dataframe = pd.read_csv(url)

# Pobranie pierwszego wiersza.
dataframe.iloc[0]

Name      Allen, Miss Elisabeth Walton
PClass    1st
Age       29
Sex       female
Survived   1
SexCode    1
Name: 0, dtype: object
```

Dwukropek może zostać wykorzystany do zdefiniowania wycinka składającego się z żądanych wierszy. W kolejnym poleceniu pokazałem przykład wycinka obejmującego wiersze od drugiego do czwartego.

```
# Pobranie trzech wierszy.
dataframe.iloc[1:4]
```

Oto dane po wykonaniu przedstawionego polecenia:

	Name	PClass	Age	Sex	Survived	SexCode
1	Allison, Miss Helen Loraine	1st	2.00	female	0	1
2	Allison, Mr Hudson Joshua Creighton	1st	30.00	male	0	0
3	Allison, Mrs Hudson JC (Bessie Waldo Daniels)	1st	25.00	female	0	1

Dwukropek może również zostać użyty do pobrania wszystkich wierszy do pewnego miejsca. W następnym przykładzie pokazałem przykład wycinka obejmującego wiersze od początku do czwartego włącznie.

```
# Pobranie wierszy do czwartego włącznie.  
dataframe.iloc[:4]
```

Oto dane po wykonaniu przedstawionego polecenia:

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.00	female	1	1
1	Allison, Miss Helen Loraine	1st	2.00	female	0	1
2	Allison, Mr Hudson Joshua Creighton	1st	30.00	male	0	0
3	Allison, Mrs Hudson JC (Bessie Waldo Daniels)	1st	25.00	female	0	1

Ramki danych nie muszą być indeksowane liczbowo. Indeksom może być dowolna wartość unikatowa w poszczególnych wierszach. Na przykład jako indeksu można użyć nazwiska pasażera, a następnie pobierać wiersze za pomocą tego nazwiska.

```
# Zdefiniowanie indeksu.  
dataframe = dataframe.set_index(dataframe['Name'])  
  
# Wyświetlenie podanego wiersza.  
dataframe.loc['Allen, Miss Elisabeth Walton']  
  
Name      Allen, Miss Elisabeth Walton  
PClass      1st  
Age         29  
Sex         female  
Survived      1  
SexCode      1  
Name: Allen, Miss Elisabeth Walton, dtype: object
```

Analiza

Wszystkie wiersze w obiekcie typu DataFrame biblioteki pandas muszą mieć indeks o unikatowej wartości. Domyślnie tym indeksem jest liczba całkowita wskazująca położenie danego wiersza w obiekcie. Można jednak zdefiniować zupełnie inny indeks. Może nim być unikatowy ciąg tekstowy lub liczba. Aby pobierać poszczególne wiersze i wycinki, biblioteka pandas oferuje dwie wymienione tutaj metody:

- `loc()` okazuje się użyteczna, gdy indeksem obiektu DataFrame jest etykieta, czyli na przykład ciąg tekstowy;
- `iloc()` działa przez wyszukanie położenia w obiekcie DataFrame. Dlatego też wywołanie `iloc[0]` zwróci pierwszy wiersz niezależnie od tego, jaką postać ma jego indeks.

Warto nabrać wprawy w pracy z obiema omówionymi tutaj metodami, ponieważ bardzo pomagają one podczas oczyszczania danych.

3.4. Pobieranie wierszy na podstawie pewnych warunków

Problem

Chcesz pobrać wiersze ramki danych na podstawie pewnych warunków.

Rozwiązanie

Ten problem można niezwykle łatwo rozwiązać za pomocą biblioteki pandas. Spójrz na przykład pokazujący, jak pobrać informacje o wszystkich kobietach znajdujących się na Titanicu.

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie adresu URL.
url = 'https://tinyurl.com/titanic-csv'

# Wczytanie danych.
dataframe = pd.read_csv(url)

# Wyświetlenie dwóch pierwszych wierszy, w których kolumna 'Sex' ma wartość 'female'.
dataframe[dataframe['Sex'] == 'female'].head(2)
```

Oto dane po wykonaniu przedstawionego kodu:

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.00	female	1	1
1	Allison, Miss Helen Loraine	1st	2.00	female	0	1

Zatrzymaj się na chwilę i przyjrzyj formatowi użytemu w rozwiązaniu. Mamy tutaj polecenie warunkowe, `dataframe['Sex'] == 'female'`, opakowane konstrukcją `dataframe[]`. W ten sposób nakazujemy bibliotece pandas pobranie wszystkich wierszy w ramce danych, których wartością `dataframe['Sex']` jest 'female'.

Równie łatwo można zastosować większą liczbę wyrażeń warunkowych. Na przykład zobacz, jak pobrać wiersze wszystkich pasażerów przedstawiające kobietę w wieku przynajmniej 65 lat.

```
# Filtrowanie wierszy.
dataframe[(dataframe['Sex'] == 'female') & (dataframe['Age'] >= 65)]
```

Oto dane po wykonaniu przedstawionego kodu:

	Name	PClass	Age	Sex	Survived	SexCode
73	Crosby, Mrs Edward Gifford (Catherine Elizabeth...	1st	69.0	female	1	1

Analiza

Warunkowe pobieranie i filtrowanie danych to jedno z podstawowych zadań wykonywanych podczas przygotowywania danych. Bardzo rzadko zależy nam na tym, by otrzymać niezmodyfikowane dane pochodzące ze źródła. Najczęściej będziesz zainteresowany jedynie ich podzbiorem.

Na przykład być może chcesz otrzymać informacje o sklepach znajdujących się w podanych województwach lub rekordy pacjentów w określonym wieku.

3.5. Zastępowanie wartości

Problem

Musisz zastąpić wartość w ramce danych.

Rozwiązanie

Oferowana przez bibliotekę pandas metoda `replace()` pozwala na bardzo łatwe wyszukiwanie i zastępowanie wartości. Spójrz na przykład pokazujący, jak prosta jest operacja zastąpienia wystąpienia słowa `female` w kolumnie `Sex` wszystkich wierszy słowem `Woman`.

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie adresu URL.
url = 'https://tinyurl.com/titanic-csv'

# Wczytanie danych.
dataframe = pd.read_csv(url)

# Zastąpienie wartości, wyświetlenie dwóch pierwszych wierszy.
dataframe['Sex'].replace("female", "Woman").head(2)

0    Woman
1    Woman
Name: Sex, dtype: object
```

Można też zastąpić wiele wartości jednocześnie.

```
# Zastąpienie słów "female" i "male" słowami "Woman" i "Man".
dataframe['Sex'].replace(["female", "male"], ["Woman", "Man"]).head(5)

0    Woman
1    Woman
2     Man
3    Woman
4     Man
Name: Sex, dtype: object
```

Operację wyszukiwania i zastępowania można przeprowadzać w całym obiekcie `DataFrame`. Wystarczy wskazać całą ramkę danych zamiast pojedynczej kolumny.

```
# Zastąpienie wartości, wyświetlenie dwóch pierwszych wierszy.
dataframe.replace(1, "One").head(2)
```

Oto dane po wykonaniu przedstawionego polecenia:

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.00	female	One	One
1	Allison, Miss Helen Loraine	1st	2.00	female	0	One

Metoda `replace()` akceptuje również wyrażenie regularne.

```
# Zastąpienie wartości, wyświetlenie dwóch pierwszych wierszy.  
dataframe.replace(r"1st", "First", regex=True).head(2)
```

Oto dane po wykonaniu przedstawionego polecenia:

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	First	29.00	female	1	1
1	Allison, Miss Helen Loraine	First	2.00	female	0	1

Analiza

Metoda `replace()` to proste, choć jednocześnie oferujące potężne możliwości narzędzie przeznaczone do zastępowania wartości. Pozwala na podanie argumentu w postaci wyrażenia regularnego.

3.6. Zmiana nazwy kolumny

Problem

Chcesz zmienić nazwę kolumny w obiekcie `DataFrame` biblioteki `pandas`.

Rozwiązanie

Aby zmienić nazwę kolumny, użyj metody `rename()`.

```
# Wczytanie biblioteki.  
import pandas as pd  
  
# Utworzenie adresu URL.  
url = 'https://tinyurl.com/titanic-csv'  
  
# Wczytanie danych.  
dataframe = pd.read_csv(url)  
  
# Zmiana nazwy kolumny, wyświetlenie dwóch pierwszych wierszy.  
dataframe.rename(columns={'PClass': 'Passenger Class'}).head(2)
```

Oto dane po wykonaniu przedstawionego kodu:

	Name	Passenger Class	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.0	female	1	1
1	Allison, Miss Helen Loraine	1st	2.0	female	0	1

Metoda `rename()` może przyjąć argument w postaci słownika, za pomocą którego istnieje możliwość jednoczesnej zmiany nazw wielu kolumn.

```
# Zmiana nazwy kolumn, wyświetlenie dwóch pierwszych wierszy.  
dataframe.rename(columns={'PClass': 'Passenger Class', 'Sex': 'Gender'}).head(2)
```

Oto dane po wykonaniu przedstawionego polecenia:

	Name	Passenger Class	Age	Gender	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.0	female	1	1
1	Allison, Miss Helen Loraine	1st	2.0	female	0	1

Analiza

Użycie metody `rename()` wraz z argumentem w postaci słownika dla parametru `columns` jest preferowanym rozwiązaniem podczas zmiany nazw kolumn, ponieważ pozwala na przeprowadzenie operacji na dowolnej liczbie kolumn. Jeżeli chcesz jednocześnie zmienić nazwy wszystkich kolumn, ten użyteczny fragment kodu tworzy słownik wraz z kluczami w postaci dotychczasowych nazw kolumn i wartościami w postaci pustych ciągów tekstowych.

```
# Wczytanie biblioteki.
import collections

# Utworzenie słownika.
column_names = collections.defaultdict(str)

# Utworzenie kluczy.
for name in dataframe.columns:
    column_names[name]

# Wyświetlenie słownika.
column_names

defaultdict(str,
            {'Age': '',
             'Name': '',
             'PClass': '',
             'Sex': '',
             'SexCode': '',
             'Survived': ''})
```

3.7. Znajdowanie wartości minimalnej, maksymalnej, sumy, średniej i liczby elementów w kolumnie

Problem

Chcesz odszukać wartość minimalną, maksymalną, sumę, średnią lub liczbę elementów w kolumnie.

Rozwiązanie

Biblioteka `pandas` zawiera pewne wbudowane metody przeznaczone do obliczania niektórych najczęściej używanych danych statystycznych.

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie adresu URL.
url = 'https://tinyurl.com/titanic-csv'
```

```
# Wczytanie danych.
dataframe = pd.read_csv(url)

# Obliczenie danych statystycznych.
print('Maksimum:', dataframe['Age'].max())
print('Minimum:', dataframe['Age'].min())
print('Średnia:', dataframe['Age'].mean())
print('Suma:', dataframe['Age'].sum())
print('Liczba elementów:', dataframe['Age'].count())

Maksimum: 71.0
Minimum: 0.17
Średnia: 30.397989417989415
Suma: 22980.879999999997
Liczba elementów: 756
```

Analiza

Poza danymi statystycznymi, które wykorzystałem w omawianym rozwiązaniu, biblioteka pandas oferuje również znalezienie wariancji (`var()`), odchylenia standardowego (`std()`), kurtozy (`kurt()`), współczynnika skośności (`skew()`), błędu standardowego średniej arytmetycznej (`sem()`), dominanty (`mode()`), mediany (`median()`) i wielu innych wartości.

Co więcej, wymienione metody można stosować dla całego obiektu `DataFrame`.

```
# Wyświetlenie liczby elementów w kolumnach.
dataframe.count()

Name      1313
PClass    1313
Age        756
Sex        1313
Survived   1313
SexCode    1313
dtype: int64
```

3.8. Znajdowanie unikatowych wartości

Problem

Chcesz pobrać wszystkie unikatowe wartości w kolumnie.

Rozwiązanie

Za pomocą metody `unique()` można wyświetlić tablicę wszystkich unikatowych wartości w kolumnie.

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie adresu URL.
url = 'https://tinyurl.com/titanic-csv'

# Wczytanie danych.
dataframe = pd.read_csv(url)
```

```
# Pobranie unikatowych wartości.
dataframe['Sex'].unique()

array(['female', 'male'], dtype=object)
```

Z kolei metoda `value_counts()` wyświetla wszystkie unikatowe wartości i podaje liczbę wystąpień każdej z nich.

```
# Wyświetlenie liczby elementów w kolumnach.
dataframe['Sex'].value_counts()

male      851
female    462
Name: Sex, dtype: int64
```

Analiza

Metody `unique()` i `value_counts()` okazują się użyteczne podczas przeprowadzania operacji i analizowania kolumn kategoryzujących. Bardzo często w takich kolumnach znajdują się klasy wymagające obsługi na etapie przygotowywania danych. Na przykład w zbiorze danych o pasażerach Titanic kolumna `PClass` określa, którą klasą podróżowała dana osoba. Na Titanicu były trzy klasy; po użyciu metody `value_counts()` będzie można dostrzec problem.

```
# Wyświetlenie liczby elementów w kolumnach.
dataframe['PClass'].value_counts()

3rd      711
1st      322
2nd      279
*          1
Name: PClass, dtype: int64
```

Wprawdzie zgodnie z oczekiwaniami praktycznie wszyscy pasażerowie należeli do jednej z tych trzech klas, ale jeden pasażer miał klasę `*`. Do obsługi takich problemów istnieje wiele strategii, które przedstawię w rozdziale 5. W tym miejscu wystarczy wiedzieć, że te „dodatkowe” klasy są często spotykane w danych kategoryzujących i nie powinny być ignorowane.

Jeżeli zachodzi potrzeba ustalenia po prostu liczby unikatowych wartości, wówczas można skorzystać z metody `nunique()`.

```
# Wyświetlenie liczby unikatowych wartości.
dataframe['PClass'].nunique()

4
```

3.9. Obsługa brakujących wartości

Problem

Chcesz ustalić, czy w obiekcie `DataFrame` brakuje jakichkolwiek wartości.

Rozwiązanie

Metody `isnull()` i `notnull()` zwracają wartość boolowską, określającą, czy brakuje danej informacji.

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie adresu URL.
url = 'https://tinyurl.com/titanic-csv'

# Wczytanie danych.
dataframe = pd.read_csv(url)

# Pobranie i wyświetlenie dwóch wierszy, w których brakuje wartości.
dataframe[dataframe['Age'].isnull()].head(2)
```

Oto dane po wykonaniu przedstawionego kodu:

	Name	PClass	Age	Sex	Survived	SexCode
12	Aubert, Mrs Leontine Pauline	1st	NaN	female	1	1
13	Barkworth, Mr Algernon H	1st	NaN	male	1	0

Analiza

Brakujące wartości to często pojawiający się problem podczas przygotowywania danych, a wielu programistów nie zdaje sobie sprawy z trudności podczas pracy z niekompletnymi danymi. Biblioteka pandas używa znanej z NumPy wartości NaN (ang. *not a number* — to nie jest liczba) do zaznaczenia braku wartości. Trzeba w tym miejscu koniecznie dodać, że wartość NaN nie została w pełni natywnie zaimplementowana w bibliotece pandas. Dlatego też jeśli chcesz zastąpić wszystkie ciągi tekstowe male wartością NaN, wynikiem takiej operacji będzie błąd.

```
# Próba zastąpienia wartości przez NaN.
dataframe['Sex'] = dataframe['Sex'].replace('male', NaN)
```

```
-----
NameError                                Traceback (most recent call last)

<ipython-input-7-5682d714f87d> in <module>()
      1 # Próba zastąpienia wartości przez NaN.
----> 2 dataframe['Sex'] = dataframe['Sex'].replace('male', NaN)

NameError: name 'NaN' is not defined
-----
```

Aby zapewnić pełną funkcjonalność NaN, najpierw trzeba zaimportować bibliotekę NumPy.

```
# Wczytanie biblioteki.
import numpy as np

# Zastąpienie wartości przez NaN.
dataframe['Sex'] = dataframe['Sex'].replace('male', np.nan)
```

Bardzo często zdarza się, że do oznaczenia brakującej obserwacji w zbiorze danych używa się pewnej wartości, na przykład NONE, -999 lub kropki. Metoda read_csv() biblioteki pandas zawiera parametr pozwalający na określenie wartości używanych do wskazania brakujących wartości.

```
# Wczytanie danych, przypisanie brakujących wartości.
dataframe = pd.read_csv(url, na_values=[np.nan, 'NONE', -999])
```

3.10. Usuwanie kolumn

Problem

Chcesz usunąć kolumnę z obiektu DataFrame.

Rozwiązanie

Najlepszym sposobem na usunięcie kolumny jest użycie metody `drop()` wraz z parametrem `axis=1`.

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie adresu URL.
url = 'https://tinyurl.com/titanic-csv'

# Wczytanie danych.
dataframe = pd.read_csv(url)

# Usunięcie kolumny.
dataframe.drop('Age', axis=1).head(2)
```

Oto dane po wykonaniu przedstawionego kodu:

	Name	PClass	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	female	1	1
1	Allison, Miss Helen Loraine	1st	female	0	1

Lista nazw kolumn może zostać użyta jako argument główny, co pozwala na jednoczesne usunięcie wielu kolumn.

```
# Usunięcie kolumn.
dataframe.drop(['Age', 'Sex'], axis=1).head(2)
```

Oto dane po wykonaniu przedstawionego polecenia:

	Name	PClass	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	1	1
1	Allison, Miss Helen Loraine	1st	0	1

Jeżeli kolumna nie ma nazwy, co się czasami zdarza, można ją usunąć przez podanie indeksu kolumny za pomocą atrybutu `dataframe.columns`, jak pokazałem w kolejnym poleceniu.

```
# Usunięcie kolumny.
dataframe.drop(dataframe.columns[1], axis=1).head(2)
```

Oto dane po wykonaniu przedstawionego polecenia:

	Name	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	29.0	female	1	1
1	Allison, Miss Helen Loraine	2.0	female	0	1

Analiza

Metoda `drop()` jest przeznaczona do usunięcia kolumny. Alternatywną metodą jest `del`, na przykład `del dataframe['Age']`, która działa w większości przypadków, ale nie jest zalecana ze względu na sposób jej wywoływania w bibliotece `pandas` (związane z tym szczegóły wykraczają poza zakres tematyczny książki).

Szczerze odradzam używanie argumentu `inplace=True`. Wiele metod biblioteki `pandas` zawiera parametr `inplace`, którego wartość `True` oznacza bezpośrednią edycję obiektu `DataFrame`. To może prowadzić do problemów w bardziej złożonych potokach przetwarzania danych, ponieważ obiekt `DataFrame` będzie traktowany jako modyfikowalny (pod względem technicznym należy go za taki uznać). Gorąco zachęcam do używania `DataFrame` jak obiektu niemodyfikowalnego. Spójrz na przedstawiony tutaj przykład.

```
# Utworzenie nowego obiektu typu DataFrame.
dataframe_name_dropped = dataframe.drop(dataframe.columns[0], axis=1)
```

W tym przykładzie nie mamy do czynienia ze zmianą obiektu typu `DataFrame`, ale z utworzeniem nowego obiektu tego typu będącego alternatywną wersją `dataframe` o nazwie `dataframe_name_dropped`. Jeżeli będziesz traktować `DataFrame` jako obiekt niemodyfikowalny, unikniesz wielu kłopotów.

3.11. Usuwanie wiersza

Problem

Chcesz usunąć jeden lub większą liczbę wierszy z obiektu `DataFrame`.

Rozwiązanie

Użyj wyrażenia boolowskiego do utworzenia nowego obiektu typu `DataFrame` niezawierającego wierszy, które chciałeś usunąć.

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie adresu URL.
url = 'https://tinyurl.com/titanic-csv'

# Wczytanie danych.
dataframe = pd.read_csv(url)

# Usunięcie wierszy, wyświetlenie dwóch pierwszych wierszy danych wyjściowych.
dataframe[dataframe['Sex'] != 'male'].head(2)
```

Oto dane po wykonaniu przedstawionego kodu:

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.0	female	1	1
1	Allison, Miss Helen Loraine	1st	2.0	female	0	1

Analiza

Mimo że z technicznego punktu widzenia możliwe jest użycie metody `drop()`, na przykład `df.drop([0, 1], axis=0)`, aby usunąć dwa pierwsze wiersze, to jednak znacznie praktyczniejszym rozwiązaniem będzie umieszczenie wyrażenia boolowskiego w wywołaniu `df[]`. Dzięki temu można wykorzystać potężne możliwości wyrażen boolowskich do usunięcia pojedynczego wiersza lub (co bardziej prawdopodobne) jednocześnie wielu wierszy.

Wyrażenia boolowskiego można używać, aby w łatwy sposób usunąć jeden wiersz dopasowany dzięki unikatowej wartości.

```
# Usunięcie wiersza, wyświetlenie dwóch pierwszych wierszy danych wyjściowych.  
dataframe[dataframe['Name'] != 'Allison, Miss Helen Loraine'].head(2)
```

Oto dane po wykonaniu przedstawionego polecenia:

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.0	female	1	1
2	Allison, Mr Hudson Joshua Creighton	1st	30.0	male	0	0

Pojedynczy wiersz można również usunąć, podając jego indeks, jak pokazałem w kolejnym poleceniu.

```
# Usunięcie wiersza, wyświetlenie dwóch pierwszych wierszy danych wyjściowych.  
dataframe[dataframe.index != 0].head(2)
```

Oto dane po wykonaniu przedstawionego polecenia:

	Name	PClass	Age	Sex	Survived	SexCode
1	Allison, Miss Helen Loraine	1st	2.0	female	0	1
2	Allison, Mr Hudson Joshua Creighton	1st	30.0	male	0	0

3.12. Usuwanie powielonych wierszy

Problem

Z obiektu `DataFrame` chcesz usunąć powielone wiersze.

Rozwiązanie

Możesz skorzystać z metody `drop_duplicates()`, ale powinieneś pamiętać o dostępnych parametrach.

```
# Wczytanie biblioteki.  
import pandas as pd  
  
# Utworzenie adresu URL.  
url = 'https://tinyurl.com/titanic-csv'  
  
# Wczytanie danych.  
dataframe = pd.read_csv(url)  
  
# Usunięcie powielonych wierszy, wyświetlenie dwóch pierwszych wierszy danych wyjściowych.  
dataframe.drop_duplicates().head(2)
```

Oto dane po wykonaniu przedstawionego kodu:

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.0	female	1	1
1	Allison, Miss Helen Loraine	1st	2.0	female	0	1

Analiza

Uważny czytelnik dostrzeże, że przedstawione tutaj rozwiązanie w rzeczywistości nie spowodowało usunięcia żadnych wierszy.

```
# Wyświetlenie liczby wierszy.  
print("Liczba wierszy w początkowym obiekcie DataFrame:", len(dataframe))  
print("Liczba wierszy po przeprowadzeniu operacji:", len(dataframe.drop_duplicates()))
```

```
Liczba wierszy w początkowym obiekcie DataFrame: 1313  
Liczba wierszy po przeprowadzeniu operacji: 1313
```

Domyślnie działanie metody `drop_duplicates()` polega na usunięciu jedynie wierszy idealnie dopasowanych we wszystkich kolumnach. W takim przypadku każdy wiersz w użytym tutaj obiekcie typu `DataFrame` jest faktycznie unikatowy. Jednak bardzo często zachodzi potrzeba sprawdzenia pod kątem powielonych wierszy tylko podzbioru kolumn. Wówczas należy skorzystać z parametru `subset`.

```
# Usunięcie powielonych wierszy.  
dataframe.drop_duplicates(subset=['Sex'])
```

Oto dane po wykonaniu przedstawionego polecenia:

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.0	female	1	1
2	Allison, Mr Hudson Joshua Creighton	1st	30.0	male	0	0

Spójrz uważnie na wygenerowane dane wyjściowe: metoda `drop_duplicates()` uznaje za duplikaty przeznaczone do usunięcia dwa dowolne wiersze o takiej samej wartości kolumny `Sex`. Po wykonaniu tej operacji mamy obiekt typu `DataFrame`, zawierający tylko dwa wiersze przedstawiające kobietę i mężczyznę. Być może zadajesz sobie pytanie, dlaczego metoda `drop_duplicates()` pozostawiła akurat te dwa wiersze. Odpowiedź okazuje się prosta: metoda pozostawia pierwsze wystąpienie powielonego wiersza i usuwa pozostałe. To zachowanie można kontrolować za pomocą parametru `keep`, jak pokazałem w kolejnym przykładzie.

```
# Usunięcie powielonych wierszy.  
dataframe.drop_duplicates(subset=['Sex'], keep='last')
```

Oto dane po wykonaniu przedstawionego polecenia:

	Name	PClass	Age	Sex	Survived	SexCode
1307	Zabour, Miss Tamini	3rd	NaN	female	0	1
1312	Zimmerman, Leo	3rd	29.0	male	0	0

Istnieje metoda o nazwie `duplicated()`, podobna do omawianej tutaj `drop_duplicates()`. Wartością zwrótną metody `duplicated()` jest seria wartości boolowskich, wskazujących, czy wiersz jest powielony. Użycie tej metody jest dobrym rozwiązaniem, jeśli nie chcesz usunąć powielonych wierszy.

3.13. Grupowanie wierszy

Problem

Chcesz pogrupować poszczególne wiersze według pewnej wartości współdzielonej.

Rozwiązanie

Metoda `groupby()` to jedna z najpotężniejszych funkcji oferowanych przez bibliotekę `pandas`.

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie adresu URL.
url = 'https://tinyurl.com/titanic-csv'

# Wczytanie danych.
dataframe = pd.read_csv(url)

# Grupowanie wierszy według wartości kolumny 'Sex'
# i obliczenie średniej w poszczególnych grupach.
dataframe.groupby('Sex').mean()
```

Oto dane po wykonaniu przedstawionego kodu:

	Age	Survived	SexCode
Sex			
Female	29.396424	0.666667	1.0
Male	31.014338	0.166863	0.0

Analiza

Użycie metody `groupby()` powoduje, że dane zaczynają nabierać kształtu. Bardzo często zdarza się, że chcemy grupować poszczególne wiersze obiektu `DataFrame` przedstawiające osobę lub zdarzenie według pewnego kryterium, aby następnie wygenerować wybrane dane statystyczne. Na przykład wyobraź sobie istnienie obiektu typu `DataFrame`, którego wiersze przedstawiają wartość pojedynczych transakcji we wszystkich lokalach danej sieci restauracji, a Twoim zadaniem jest obliczenie wartości wszystkich transakcji w poszczególnych lokalach. To zadanie można wykonać poprzez zgrupowanie wierszy określających poszczególne restauracje, a następnie obliczenie sumy całej grupy.

Początkujący mają tendencję do używania metody `groupby()` w pokazany tutaj sposób, a następnie są zdziwieni otrzymanym wynikiem.

```
# Grupowanie wierszy.
dataframe.groupby('Sex')

<pandas.core.groupby.DataFrameGroupBy object at 0x10efacf28>
```

Dlaczego przedstawione polecenie nie zwróciło bardziej użytecznego wyniku? Wywołanie metody `groupby()` trzeba po prostu połączyć z pewną operacją, która ma być przeprowadzona na każdej grupie, na przykład z obliczeniem zagregowanych danych statystycznych, takich jak średnia, mediana, suma itd. Kiedy mówi się o grupowaniu, często słyszy się określenie „grupowanie według płci”, ale to nie do końca odpowiednie stwierdzenie. Aby grupowanie było użyteczne, konieczne jest grupowanie pewnych wartości i następnie przeprowadzenie operacji na poszczególnych grupach.

```
# Grupowanie wierszy, a następnie ich zliczenie.  
dataframe.groupby('Survived')['Name'].count()
```

```
Survived  
0      863  
1      450  
Name: Name, dtype: int64
```

Zwróć uwagę na dodanie `Name` po wywołaniu metody `groupby()`. Wynika to z tego, że podsumowanie określonych danych statystycznych ma sens jedynie dla pewnych typów danych. Na przykład obliczanie średniego wieku według płci ma sens, natomiast obliczanie sumy lat według płci nie ma żadnego sensu. W omawianym przypadku zostały zgrupowane dane dotyczące osób, które przeżyły i nie przeżyły katastrofy Titanica, a następnie zliczana jest liczba pasażerów w poszczególnych grupach.

Istnieje również możliwość grupowania według pierwszej kolumny, a następnie według drugiej, jak pokazałem w kolejnym fragmencie kodu.

```
# Grupowanie wierszy, obliczenie średniej.  
dataframe.groupby(['Sex', 'Survived'])['Age'].mean()
```

```
Sex      Survived  
female  0          24.901408  
        1          30.867143  
male    0          32.320780  
        1          25.951875  
Name: Age, dtype: float64
```

3.14. Grupowanie wierszy według czasu

Problem

Musisz pogrupować pojedyncze wiersze według czasu.

Rozwiązanie

Użyj oferowanej przez bibliotekę `pandas` metody `resample()` do grupowania wierszy według czasu.

```
# Wczytanie bibliotek.  
import pandas as pd  
import numpy as np  
  
# Utworzenie przedziału czasu.  
time_index = pd.date_range('06/06/2017', periods=100000, freq='30S')  
  
# Utworzenie obiektu typu DataFrame.  
dataframe = pd.DataFrame(index=time_index)
```

```
# Utworzenie kolumny losowo wybranych wartości.
dataframe['Sale_Amount'] = np.random.randint(1, 10, 100000)

# Grupowanie wierszy według tygodnia, obliczenie sumy dla danego tygodnia.
dataframe.resample('W').sum()
```

Oto dane po wykonaniu przedstawionego kodu:

	Sale_Amount
2017-06-11	86423
2017-06-18	101045
2017-06-25	100867
2017-07-02	100894
2017-07-09	100438
2017-07-16	10297

Analiza

Wykorzystywany dość często w rozdziale zbiór danych dotyczących pasażerów Titanica nie ma kolumny z wartościami czasu. Dlatego też w tej recepturze wygenerowałem prosty obiekt typu DataFrame, którego wiersze przedstawiają pojedyncze transakcje. Znamy datę i godzinę oraz wyrażoną w złotych wartość każdej transakcji. (Te dane nie są zbyt rzeczywiste, ponieważ transakcje odbyły się w odstępach 30 sekund i wartość każdej z nich jest kwotą wyrażoną w pełnych złotych, ale na potrzeby omawianego tutaj rozwiązania jest to w zupełności wystarczające).

Niezmodyfikowane dane w trzech pierwszych wierszach przedstawiają się następująco:

```
# Wyświetlenie trzech pierwszych wierszy.
dataframe.head(3)
```

Oto dane po wykonaniu przedstawionego polecenia:

	Sale_Amount
2017-06-06 00:00:00	7
2017-06-06 00:00:30	2
2017-06-06 00:01:00	7

Zwróć uwagę na to, że data i godzina poszczególnych transakcji to indeks w obiekcie typu DataFrame. Metoda `resample()` wymaga indeksu dla wartości przedstawiających datę i godzinę.

Za pomocą tej metody można grupować wiersze według różnych przedziałów czasu, a następnie przeprowadzać pewne obliczenia w tych grupach, jak pokazałem w dwóch kolejnych poleceniach.

```
# Grupowanie według dwóch tygodni, obliczenie średniej.
dataframe.resample('2W').mean()
```

Oto dane po wykonaniu przedstawionego polecenia:

	Sale_Amount
2017-06-11	5.001331
2017-06-25	5.007738
2017-07-09	4.993353
2017-07-23	4.950481

```
# Grupowanie według miesiąca, obliczenie liczby wierszy.  
dataframe.resample('M').count()
```

Oto dane po wykonaniu przedstawionego polecenia:

	Sale_Amount
2017-06-30	72000
2017-07-31	28000

Zwróć uwagę na wygenerowanie dwóch elementów danych wyjściowych indeksu daty i godziny, pomimo że grupowanie odbywa się według — odpowiednio — tygodni i miesięcy. Domyślnie metoda `resample()` zwraca etykietę prawej „krawędzi” (ostatnia etykieta) grupy daty i godziny. To zachowanie można kontrolować za pomocą parametru `label`.

```
# Grupowanie według miesiąca, obliczenie liczby wierszy.  
dataframe.resample('M', label='left').count()
```

Oto dane po wykonaniu przedstawionego polecenia:

	Sale_Amount
2017-05-31	72000
2017-06-30	28000

Zobacz również

- Lista aliasów biblioteki `pandas` związanych z wartościami daty i godziny na stronie <http://pandas.pydata.org/pandas-docs/stable/timeseries.html#offset-aliases>.

3.15. Iterowanie przez kolumnę

Problem

Chcesz przeprowadzić iterację przez wszystkie elementy kolumny i wykonać na nich jakieś działanie.

Rozwiązanie

Kolumnę w strukturze danych biblioteki `pandas` można potraktować jak każdą inną sekwencję Pythona.

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie adresu URL.
url = 'https://tinyurl.com/titanic-csv'

# Wczytanie danych.
dataframe = pd.read_csv(url)

# Wyświetlenie wielkimi literami imienia i nazwiska dwóch pierwszych pasażerów.
for name in dataframe['Name'][0:2]:
    print(name.upper())

ALLEN, MISS ELISABETH WALTON
ALLISON, MISS HELEN LORAINÉ
```

Analiza

Poza pętlami (często określanymi mianem pętli for) można również użyć listy składanej, jak pokazałem w kolejnym przykładzie.

```
# Wyświetlenie wielkimi literami imienia i nazwiska dwóch pierwszych pasażerów.
[name.upper() for name in dataframe['Name'][0:2]]

['ALLEN, MISS ELISABETH WALTON', 'ALLISON, MISS HELEN LORAINÉ']
```

Wprawdzie rozwiązaniem awaryjnym jest pętla for, ale bardziej zgodnym ze stylem „Pythonic” podejściem jest użycie oferowanej przez bibliotekę pandas metody apply(), co przedstawię w następnej recepturze.

3.16. Wywoływanie funkcji dla wszystkich elementów kolumny

Problem

Chcesz wywołać pewną funkcję dla wszystkich elementów kolumny.

Rozwiązanie

Użyj metody apply(), aby dla każdego elementu kolumny wywołać funkcję wbudowaną lub niestandardową.

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie adresu URL.
url = 'https://tinyurl.com/titanic-csv'

# Wczytanie danych.
dataframe = pd.read_csv(url)

# Utworzenie funkcji.
def uppercase(x):
    return x.upper()
```

```
# Wywołanie funkcji, wyświetlenie dwóch pierwszych wierszy.
dataframe['Name'].apply(uppercase)[0:2]

0    ALLEN, MISS ELISABETH WALTON
1    ALLISON, MISS HELEN LORAINE
Name: Name, dtype: object
```

Analiza

Metoda `apply()` sprawdza się doskonale podczas oczyszczania i przygotowywania danych. Bardzo często tworzy się funkcje przeznaczone do przeprowadzania pewnych użytecznych operacji (rozdzielania imienia i nazwiska, konwersji ciągu tekstowego na liczbę itd.), a następnie wywołuje się tę funkcję dla każdego elementu kolumny.

3.17. Wywoływanie funkcji dla grupy

Problem

Masz utworzone za pomocą metody `groupby()` grupy wierszy i chcesz w nich wywołać pewną funkcję.

Rozwiązanie

Najlepszym rozwiązaniem będzie połączenie wywołań metod `groupby()` i `apply()`.

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie adresu URL.
url = 'https://tinyurl.com/titanic-csv'

# Wczytanie danych.
dataframe = pd.read_csv(url)

# Grupowanie wierszy, wywołanie funkcji w grupach.
dataframe.groupby('Sex').apply(lambda x: x.count())
```

Oto dane po wykonaniu przedstawionego kodu:

	Name	PClass	Age	Sex	Survived	SexCode
Sex						
female	462	462	288	462	462	462
male	851	851	468	851	851	851

Analiza

W recepturze 3.16 wspomniałem o metodzie `apply()`. Ta metoda okazuje się szczególnie użyteczna, gdy chcesz wywołać pewną funkcję w grupie elementów. Dzięki połączeniu wywołań metod `groupby()` i `apply()` można wygenerować niestandardowe dane statystyczne lub wykonywać dowolne funkcje w poszczególnych grupach.

3.18. Konkatenacja obiektów typu DataFrame

Problem

Chcesz przeprowadzić konkatenację dwóch obiektów typu DataFrame.

Rozwiązanie

Do przeprowadzenia konkatenacji obiektów typu DataFrame na podstawie wierszy użyj metody `concat()` wraz z argumentem `axis=0`.

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie obiektu typu DataFrame.
data_a = {'id': ['1', '2', '3'],
          'first': ['Alex', 'Amy', 'Allen'],
          'last': ['Anderson', 'Ackerman', 'Ali']}
dataframe_a = pd.DataFrame(data_a, columns = ['id', 'first', 'last'])

# Utworzenie obiektu typu DataFrame.
data_b = {'id': ['4', '5', '6'],
          'first': ['Billy', 'Brian', 'Bran'],
          'last': ['Bonder', 'Black', 'Balwner']}
dataframe_b = pd.DataFrame(data_b, columns = ['id', 'first', 'last'])

# Konkatenacja obiektów DataFrame na podstawie wierszy.
pd.concat([dataframe_a, dataframe_b], axis=0)
```

Oto dane po wykonaniu przedstawionego kodu:

	id	first	last
0	1	Alex	Anderson
1	2	Amy	Ackerman
2	3	Allen	Ali
0	4	Billy	Bonder
1	5	Brian	Black
2	6	Bran	Balwner

Konkatenacja na podstawie kolumny odbywa się po użyciu argumentu `axis=1` w wywołaniu metody `concat()`.

```
# Konkatenacja obiektów DataFrame na podstawie kolumn.
pd.concat([dataframe_a, dataframe_b], axis=1)
```

Oto dane po wykonaniu przedstawionego kodu:

	id	first	last	id	first	last
0	1	Alex	Anderson	4	Billy	Bonder
1	2	Amy	Ackerman	5	Brian	Black
2	3	Allen	Ali	6	Bran	Balwner

Analiza

Konkatenacja to nie jest słowo, które zbyt często pada w informatyce i programowaniu. Dlatego też nie przejmuj się, jeśli wcześniej go nie słyszałeś. Nieformalna definicja **konkatenacji** to połączenie dwóch obiektów. W omawianym tutaj rozwiązaniu pokazałem połączenie dwóch małych obiektów typu DataFrame za pomocą parametru `axis`, wskazującego, czy dwa obiekty mają być ułożone jeden na drugim, czy obok siebie.

Alternatywne rozwiązanie polega na wywołaniu metody `append()` i dodaniu za jej pomocą nowego wiersza do obiektu DataFrame.

```
# Utworzenie nowego wiersza.
row = pd.Series([10, 'Chris', 'Chillon'], index=['id', 'first', 'last'])

# Dołączenie nowego wiersza.
dataframe_a.append(row, ignore_index=True)
```

Oto dane po wykonaniu przedstawionego kodu:

	id	first	last
0	1	Alex	Anderson
1	2	Amy	Ackerman
2	3	Allen	Ali
3	10	Chris	Chillon

3.19. Złączanie obiektów typu DataFrame

Problem

Chcesz złączyć ze sobą dwa obiekty typu DataFrame.

Rozwiązanie

Do przeprowadzenia złączenia wewnętrznego należy użyć metody `merge()` wraz z parametrem `on` wskazującym kolumnę, na podstawie której zostanie przeprowadzona operacja.

```
# Wczytanie biblioteki.
import pandas as pd

# Utworzenie obiektu typu DataFrame.
employee_data = {'employee_id': ['1', '2', '3', '4'],
                 'name': ['Amy Jones', 'Allen Keys', 'Alice Bees',
                          'Tim Horton']}
dataframe_employees = pd.DataFrame(employee_data, columns = ['employee_id',
                                                           'name'])

# Utworzenie obiektu typu DataFrame.
sales_data = {'employee_id': ['3', '4', '5', '6'],
              'total_sales': [23456, 2512, 2345, 1455]}
dataframe_sales = pd.DataFrame(sales_data, columns = ['employee_id',
                                                     'total_sales'])
```

```
# Złączenie obiektów typu DataFrame.
pd.merge(dataframe_employees, dataframe_sales, on='employee_id')
```

Oto dane po wykonaniu przedstawionego kodu:

	employee_id	name	total_sales
0	3	Alice Bees	23456
1	4	Tim Horton	2512

Domyślnie metoda `merge()` przeprowadza złączenie wewnętrzne. Jeżeli chcesz przeprowadzić złączenie zewnętrzne, możesz to określić za pomocą parametru `how`.

```
# Złączenie obiektów typu DataFrame.
pd.merge(dataframe_employees, dataframe_sales, on='employee_id', how='outer')
```

Oto dane po wykonaniu przedstawionego polecenia:

	employee_id	name	total_sales
0	1	Amy Jones	NaN
1	2	Allen Keys	NaN
2	3	Alice Bees	23456.0
3	4	Tim Horton	2512.0
4	5	NaN	2345.0
5	6	NaN	1455.0

Ten sam parametr może zostać użyty do określenia złączenia lewego i prawego.

```
# Złączenie obiektów typu DataFrame.
pd.merge(dataframe_employees, dataframe_sales, on='employee_id', how='left')
```

Oto dane po wykonaniu przedstawionego polecenia:

	employee_id	name	total_sales
0	1	Amy Jones	NaN
1	2	Allen Keys	NaN
2	3	Alice Bees	23456.0
3	4	Tim Horton	2512.0

Istnieje również możliwość podania w poszczególnych obiektach typu `DataFrame` nazwy kolumny, na podstawie której odbędzie się łączenie.

```
# Złączenie obiektów typu DataFrame.
pd.merge(dataframe_employees,
         dataframe_sales,
         left_on='employee_id',
         right_on='employee_id')
```

Jeżeli zamiast złączenia na podstawie dwóch kolumn chcesz przeprowadzić złączenie na podstawie indeksów poszczególnych obiektów typu `DataFrame`, parametry `left_on` i `right_on` możesz zastąpić parametrami `right_index=True` i `left_index=True`.

Oto dane po wykonaniu przedstawionego kodu:

	employee_id	name	total_sales
0	3	Alice Bees	23456
1	4	Tim Horton	2512

Analiza

Bardzo często dane konieczne do użycia są skomplikowane i nie zawsze będą dostępne w postaci jednego bloku. W rzeczywistych projektach zwykle mamy oddzielne zbiory danych pochodzące z wielu plików bądź zapytań do różnych baz danych. Aby zebrać razem wszystkie dane, można najpierw umieścić w oddzielnych obiektach typu `DataFrame` dane pochodzące z poszczególnych źródeł, a następnie połączyć je ze sobą w jeden obiekt `DataFrame`.

Ten proces powinien być znany programistom, którzy mają doświadczenie w pracy z SQL, czyli popularnym językiem zapytań pozwalającym między innymi na operacje łączenia danych (w SQL są one określane mianem **złączeń**). Wprawdzie konkretne parametry wykorzystywane przez bibliotekę `pandas` będą inne, ale ogólna zasada działania złączenia jest taka sama jak w innych narzędziach i językach programowania.

W trakcie operacji złączenia należy zwrócić uwagę na trzy aspekty. Po pierwsze, konieczne jest podanie dwóch obiektów typu `DataFrame` przeznaczonych do złączenia. W omawianym rozwiązaniu mają one nazwy `dataframe_employees` i `dataframe_sales`. Po drugie, trzeba podać nazwę kolumny (lub kolumn), na podstawie której zostanie przeprowadzone złączenie. To będzie kolumna, której wartość jest współdzielona przez dwa obiekty `DataFrame`. Na przykład w omawianym przykładzie oba obiekty mają kolumnę o nazwie `employee_id`. Aby połączyć oba obiekty, trzeba w nich dopasować wartości wymienionej kolumny. Jeżeli obie kolumny mają tę samą nazwę, można użyć parametru `on`. Natomiast w przeciwnym razie należy skorzystać z parametrów `left_on` lub `right_on`.

Który obiekt `DataFrame` jest uznawany za lewy, a który za prawy? To proste — lewym jest obiekt `DataFrame` podany jako pierwszy w wywołaniu metody `merge()`. Ta koncepcja pojawia się ponownie w następnym zestawie parametrów.

Trzeci aspekt złączenia jest jednocześnie najtrudniejszy — to konieczność wskazania typu przeprowadzanej operacji złączenia. Do określenia typu używamy parametru `how`. Metoda `merge()` obsługuje cztery podstawowe typy złączeń.

Wewnętrzne

Zwracane są jedynie wiersze dopasowane w obu obiektach `DataFrame`. W omawianym przykładzie zwrócony będzie każdy wiersz, którego wartość `employee_id` występuje zarówno w obiekcie `dataframe_employees`, jak i `dataframe_sales`.

Zewnętrzne

Zwracane są wszystkie wiersze w obu obiektach `DataFrame`. Jeżeli wiersz istnieje tylko w jednym obiekcie, wówczas brakujące wartości są zastępowane przez `NaN`. W omawianym przykładzie zwrócone będą wszystkie wiersze znajdujące się w obiektach `dataframe_employees` i `dataframe_sales`.

Lewe

Z lewego obiektu `DataFrame` zwracane są wszystkie wiersze, natomiast z prawego tylko te, które zostały dopasowane do wierszy w lewym obiekcie. Brakujące wartości są zastępowane przez `NaN`. W omawianym przykładzie zwrócone będą wszystkie wiersze obiektu `dataframe_employees`, zaś z obiektu `dataframe_sales` tylko te, których wartość `employee_id` została dopasowana do wartości tej samej kolumny w `dataframe_employees`.

Prawe

Z prawego obiektu `DataFrame` zwracane są wszystkie wiersze, natomiast z lewego tylko te, które zostały dopasowane do wierszy w prawym obiekcie. Brakujące wartości są zastępowane przez `NaN`. W omawianym przykładzie zwrócone będą wszystkie wiersze obiektu `dataframe_sales`, zaś z obiektu `dataframe_employees` tylko te, których wartość `employee_id` została dopasowana do wartości tej samej kolumny w `dataframe_sales`.

Nie przejmuj się, jeżeli to wyjaśnienie nie jest do końca jasne. Zachęcam Cię do eksperymentów z parametrem `how` w kodzie i obserwacji efektu, jaki ten parametr ma na dane wyjściowe zwracane przez metodę `merge()`.

Zobacz również

- Artykuł *A Visual Explanation of SQL Joins* opublikowany na stronie <https://blog.codinghorror.com/a-visual-explanation-of-sql-joins/>.
- Dokumentacja biblioteki `pandas` dotycząca złączeń na stronie <http://pandas.pydata.org/pandas-docs/stable/merging.html>.

A

algorytm

- base_estimator, 249
- DBSCAN, 292
- GrabCut, 145
- k najbliższych sąsiadów, 89, 251
- meanshift, 290
- uczenia, 12
- uczenia maszynowego, 216

analiza

- dyskryminacyjna, 167
- głównych składowych, 163, 164

aproksymacja wielomianowa, 229

B

baza danych SQL, 42

bias, 228

biblioteka

- Keras, 320
- NLTK, 106
- NumPy, 15
- OpenCV, 129
- pandas, 40
- scikit-learn, 35, 73, 78

błąd

- out-of-bag, 250
- średniokwadratowy, 199

C

cechy

- binarne, 175, 283
- ciągłe, 280
- dyskretne lub liczebne, 282
- dyskretyzacja, 84

generowanie, 78

kategoryzujące

- nominalne, 92
- porządkowe, 94

liczbowe, 173

nieistotne, 177

niekompletne, 98

redukowanie, 161, 164–170

rekurencyjne eliminowanie, 179

skorelowane, 176

standaryzowanie, 74

transformacja, 79

usuwanie, 177

użycie histogramu koloru, 157

użycie średniej koloru, 155

wielomianowe, 78

chi-kwadrat, 177

CNN, convolutional neural networks, 323

CSC, compressed sparse column, 18

CSV, comma-separated values, 39

CV, cross-validation, 179

D

dane, 13

HTML, 105

data i godzina, 117

DBSCAN, 292

dodawanie macierzy, 30

dokładność, 189

dostrajanie

hiperparametru, 211

sieci neuronowej, 318

downsampling, 100

drzewa decyzyjne, 235, 237, 238

dummying, 93

dyskretyzacja cech, 84

E

elementy
 odstające, 82
 tablicy, 20
eliminowanie rekurencyjne cech, 179
epoka, 296
estymator błędu out-of-bag, 250
etap przygotowywania danych, 217

F

format
 CSV, 39
 JSON, 41
FPR, false positive rate, 194
funkcja
 aktywacji, 298
 bazowa, 271
 jądra, 270
 ReLU, 298

G

generowanie
 cech wielomianowych, 78
 liczb losowych, 33
 prognoz, 305
grupowanie
 obserwacji, 85
 wierszy, 62, 63

H

hiperparametr, 13, 207, 211
hiperpłaszczyzna, 267, 269
 liniowa, 272
histogram koloru, 157

I

IDD, independent identically distributed, 185
identyfikacja wektorów nośnych, 275
iloczyn skalarny, 29
interakcje, 227
iterowanie przez kolumnę, 65
izolowanie kolorów, 141

J

jednokierunkowa sieć neuronowa, 295

K

kalibrowanie prognozowanego
 prawdopodobieństwa, 284
klasa
 BernoulliNB, 283
 CalibratedClassifierCV, 285
 CountVectorizer, 111, 112
 DecisionTreeClassifier, 236
 DictVectorizer, 96, 97
 DummyClassifier, 189
 DummyRegressor, 187
 EllipticEnvelope, 81
 FunctionTransformer, 79
 GaussianNB, 280
 GridSearchCV, 212–214, 222
 ImageDataGenerator, 326
 LabelBinarizer, 92
 LinearDiscriminantAnalysis, 168
 LinearSVC, 268
 LogisticRegression, 259–264
 LogisticRegressionCV, 263
 MiniBatchKMeans, 290
 MinMaxScaler, 73, 74
 Normalizer, 77
 PCA, 162
 PolynomialFeatures, 227, 228
 PorterStemmer, 108
 RadiusNeighborsClassifier, 256, 257
 RandomForestClassifier, 241
 RandomForestRegressor, 242
 RandomizedSearchCV, 214
 RobustScaler, 76, 84
 StandardScaler, 75, 296
 SVC, 273
klaster, 85
klasteryzacja danych, 201, 287
 aglomeracyjna, 293
 algorytm DBSCAN, 292
 algorytm meanshift, 290
 k średnich, 287, 290
 łączenie hierarchiczne, 293
klasy niezerównoważone, 246, 264, 276

klasyfikacja

- binarna, 299
- na podstawie promienia, 257
- obrazów, 322
- wielu klas, 299

klasyfikator

- bayesowski, 279, 283
- binarny, 189, 259, 300
 - progowanie, 192
- dla cech ciągłych, 280
- dla cech dyskretnych, 282
- drzewa decyzyjnego, 235
- k najbliższych sąsiadów, 253
- liniowy, 267
- losowego lasu, 240
- najbliższych sąsiadów, 256
- wektora nośnego, 270
- wieloklasowy, 195, 260, 302

klasyfikowanie tekstu, 327

kodowanie

- dni tygodnia, 123
- gorącojedynkowe, 93
- nominalnych cech kategoryzujących, 92
- porządkowych cech kategoryzujących, 94
- słowników cech, 96
- tekstu, 111

konkatenacja obiektów, 68

konwertowanie ciągu tekstowego, 117

korelacja, 177

krzywa

- ROC, 195
- uczenia, 205

L

las

- drzew decyzyjnych, 240
- losowy, 242

LASSO, 232

LDA, linear discriminant analysis, 166

liczby pseudolosowe, 33

LSTM, 327

Ł

łączenie hierarchiczne, 293

M

macierz, 16

- redukcja wymiarowości, 168

macierze

- cech, 96
- korelacji, 177
- mnożenie, 31
- obliczanie śladu, 27
- obliczanie wyznacznika, 26
- odejmowanie, 30
- odwracanie, 32
- opisywanie, 20
- pobieranie przekątnej, 27
- pomylek, 197
- rzadkie, 17, 170
- splaszczanie, 25
- transponowanie, 24
- tworzenie, 16
- wartości własne, 28
- znajdowanie rzędu, 25
- znajdowanie wektorów, 28

maksymalizacja rozłączności klas, 166

mapa cech, 324

maszyna wektora nośnego, 267, 270

MEA, mean absolute error, 238

metoda

- `adaptiveThreshold()`, 144
- `add()`, 30
- `apply()`, 66, 67, 80
- `classification_report()`, 207
- `concat()`, 68
- `cornerHarris()`, 150, 151
- `cross_val_score()`, 189
- `describe()`, 48
- `det()`, 26
- `diagonal()`, 27
- `dot()`, 29, 31
- `drop()`, 58, 59, 60
- `drop_duplicates()`, 60
- `equalizeHist()`, 139
- `f_classif()`, 179
- `filter2D()`, 137, 138
- `fit()`, 74
- `fit_transform()`, 74
- `flatten()`, 25, 153
- `groupby()`, 62, 67

metoda

- head(), 47
- imread(), 130
- imwrite(), 132
- interpolate(), 128
- inv(), 32
- inverse_transform(), 92
- isnull(), 56
- kurt(), 55
- LASSO, 232
- linalg.eig(), 28
- linalg.inv(), 32
- loc(), 50
- make_blobs(), 37, 39
- make_circles(), 164
- make_classification(), 37–39
- make_regression(), 38, 39
- make_scorer(), 202
- matrix_rank(), 26
- max(), 22
- mean(), 22
- median(), 55
- merge(), 69, 71
- mode(), 55
- model_to_dot(), 320
- notnull(), 56
- predict(), 226, 283
- predict_proba(), 273, 275
- read_excel(), 40
- read_json(), 41
- read_sql_query(), 42
- rename(), 53
- replace(), 103
- replace(), 52, 94
- resample(), 63, 64
- reshape(), 24
- resize(), 133
- rolling(), 126
- sem(), 55
- shift(), 124
- skew(), 55
- split(), 103
- std(), 55
- strip(), 103
- subtract(), 30
- to_datetime(), 117
- toarray(), 114
- tokenizacja, 106
- trace(), 28

- transform(), 74
- translate(), 106
- unique(), 56
- validation_curve(), 209
- value_counts(), 56
- var(), 55

metody charakterystyczne dla algorytmu, 220

MLR, multinomial logistic regression, 261

mnożenie macierzy, 31

model, 13

- drzewa decyzyjnego, 238
- klasteryzacji, 201
- klasyfikacji bazowej, 188
- ocena wydajności, 221
- regresji, 199
 - bazowej, 186
 - losowego lasu, 242
- sprawdzianu krzyżowego, 183
- wczytywanie, 331
- zależności nieliniowej, 228
- zapisywanie, 331

modyfikacja obrazu, 325

MSE, mean squared error, 199, 237

N

nadmierne dopasowanie, 310–313

naiwny klasyfikator bayesowski, 279, 283

NLTK, natural language toolkit, 106

NMF, non-negative matrix factorization, 169

normalizowanie obserwacji, 76

O

obciążenie, 228

obiekt typu DataFrame, 120

obliczanie

- iloczynu skalarnego, 29
- różnicy między datami, 122

obrazy

- izolowanie kolorów, 141
- kadrowanie, 134
- klasyfikacja, 322
- konwertowanie na obserwację, 153
- modyfikacja, 325
- progowanie, 142
- rozmywanie, 135
- usuwanie tła, 145
- usuwanie zakłóceń, 144

- wczytywanie, 129
- wykrywanie krawędzi, 147
- wykrywanie narożników, 150
- wyostrzanie, 138
- zapisywanie, 132
- zmiana wielkości, 133
- zwiększanie kontrastu, 138
- obserwacja, 12
- obserwacje odstające, 80
- obsługa
 - brakujących danych, 126
 - cech skorelowanych, 176
 - danych kategoryzujących, 91
 - danych liczbowych, 73
 - daty i godziny, 117
 - elementów odstających, 82
 - niezrównoważonych klas, 99, 246, 264, 276
 - obrazów, 129
 - stref czasowych, 118
 - tekstu, 103
- ocena modelu, 183
- odwracanie macierzy, 32
- określanie wagi słów, 113
- operacje na elementach tablicy, 20
- opisywanie
 - danych, 47
 - macierzy, 20
- opóźnienie cechy, 124
- out-of-bag, 250
- OVR, one-vs-rest, 261
- oznaczanie części mowy, 109

P

- paczka, 296
- pakiet NLTK, 109
- parametr, 13
- PCA, principal component analysis, 162
- pełność, 191
- perceptron wielowarstwowy, 295
- pliki
 - CSV, 39
 - Excelsa, 40
 - JSON, 41
- pobieranie
 - daty i godziny, 120
 - elementów, 18
 - wierszy, 51
- podzbiór, 185

- podział danych daty, 121
- porzucenie, 313
- prawdopodobieństwo
 - brzegowe, 280
 - posteriori, 279
 - prognozowane, 273
 - kalibrowanie, 284
- precyzja, 191
- progowanie, 142
 - adaptacyjne, 143
 - klasyfikatora binarnego, 192
 - wariancji cechy, 173, 175
- propagacja w przód, 296
- przekątna macierzy, 27
- przeskalowywanie cechy, 73
- przestrzeń barw
 - HSV, 142
 - RGB, 131
- przeszukiwanie losowe, 214
- przygotowywanie danych, 45

R

- ramka danych, 46, 49
- raport tekstowy, 206
- redukcja
 - cech, 164–170, 232
 - nadmiernego dopasowania, 310–313
 - wariancji, 230, 262
 - wymiarowości, 161, 173
- regresja, 199, 299
 - drzewa decyzyjnego, 237
 - liniowa, 225
 - interakcje, 227
 - redukowanie cech, 232
 - redukowanie wariancji, 230
 - wyznaczanie linii, 225
 - logistyczna, 259
 - losowego lasu, 242
- regularyzacja, 230, 262
 - wagi, 310
- RFE, recursive feature elimination, 179
- RNN, radius-based nearest neighbors, 257
- RNN, recurrent neural network, 328
- rodzaje brakujących danych, 88
- rozkład
 - macierzy, 168
 - według wartości osobliwych, 170

rozstęp ćwiartkowy, 81
równoległość, 219
RSS, residual sum of squares, 231

S

SAG, stochastic average gradient, 263
sieci neuronowe, 295
sieć neuronowa
dokładność, 307
dostrajanie, 318
generowanie prognoz, 305
klasyfikator binarny, 300
klasyfikator wieloklasowy, 302
poprawa wydajności, 325
porzucenie, 313
projektowanie, 297
regularyzacja wagi, 310
sprawdzian krzyżowy, 316
strata, 307
trenowanie regresora, 304
typu LSTM, 327
typu RNN, 328
wizualizacja, 320
zakończenie procesu uczenia, 311
zapisywanie, 315
skalowanie min. – max., 74
składowe główne, 161
słownik, 96
solver, 263
sprawdzian krzyżowy, 183, 316
k-krotny, 185
SQL, structured query language, 42
standaryzowanie cechy, 74
stemming słów, 108
stopwords, 108
strata, 13
strefy czasowe, 118
SVC, support vector classifier, 267
SVD, singular value decomposition, 170

Ś

śląd macierzy, 27
średnia koloru, 155

T

tablice
dane statystyczne, 22
zmiana kształtu, 23
znajdowanie wartości maksymalnej, 21
technika
AdaBoost, 249
porzucenia, 313
tf-idf, 113
tekst
klasyfikowanie, 327
kodowanie, 111
konwertowanie na datę, 117
oczyszczanie, 103
usuwanie słów, 107
usuwanie znaku, 105
testowanie regresora losowego lasu, 241
tokenizacja tekstu, 106
TPR, true positive rate, 194
transformacja cech, 79
transponowanie macierzy, 24
trenowanie, 13
trenowanie klasyfikatora
bardzo duże dane, 263
trenowanie
klasyfikatora
binarnego, 259, 300
dla cech ciągłych, 280
dla cech dyskretnych, 282
drzewa decyzyjnego, 235
liniowego, 267
losowego lasu, 240
naiwnego bayesowskiego, 283
wieloklasowego, 260, 302
regresora, 304
drzewa decyzyjnego, 237
TSVD, truncated singular value decomposition, 170
twierdzenie Bayesa, 279
tworzenie
cech, 153
macierzy, 16
macierzy rzadkiej, 17
ramki danych, 46
raportu tekstowego, 206
wektora, 15
typy złączeń, 71

U

- uczenie, 13
 - głębokie, 295
 - maszynowe, 153
- upsampling, 100
- usuwanie
 - kolumn, 58
 - obserwacji, 87
 - powielonych wierszy, 60
 - słów, 107
 - wiersza, 59
 - znaku przestankowego, 105
- uzupełnianie brakujących wartości, 88, 89

W

- waga, 181
- wariancja, 230
- warstwy, 299
- wartość
 - maksymalna tablicy, 21
 - progowa, 194
 - progowa wariancji, 174
- wczytywanie
 - danych, 35
 - pliku CSV, 39
 - pliku Excela, 40
 - pliku JSON, 41
 - wytrenowanych modeli, 331
- wektor, 15
- wektor nośny, 267
 - identyfikacja, 275
- weryfikacja, 184
- wielkość drzewa decyzyjnego, 247
- wielomianowa regresja logistyczna, 261
- wizualizacja, 204
 - efektu, 207
 - historii trenowania, 307
 - modelu drzewa decyzyjnego, 238
 - sieci neuronowej, 320
 - wydajności klasyfikatora, 197
- wskaźnik Giniego, 236
- współczynnik determinacji, 199
 - oceny, 206
 - oceny modelu, 202
 - pomiaru odległości, 252

- wsteczna propagacja, 296
- wybór
 - cech, 173, 245
 - modelu, 211–220
- wydajność, 13
 - klasyfikatora, 197
 - modelu, 221
 - modelu regresji, 199
 - sieci neuronowej, 325
- wydziałanie, 184
- wykres dokładności zbiorów, 309
- wykrywanie krawędzi, 147
- wyodrębnianie cech, 161
- wyszukiwanie
 - najbliższych sąsiadów, 251
 - wyczerpujące, 212
- wywoływanie funkcji, 66, 67
- wyznacznik macierzy, 26
- wzmocnienie, 248

Z

- zależności nieliniowe, 228
- zapisywanie
 - postępu modelu uczącego, 315
 - wytrenowanych modeli, 331
- zapytania do bazy danych, 42
- zastępowanie wartości, 52
- zbiór
 - danych, 35
 - symulowanych danych, 36
 - testowy, 184
 - uczący, 184, 204
- złączanie obiektów, 69
- złączenia, 71
- zmiana nazwy kolumny, 53
- zmienna losowa Bernoulliego, 175
- znajdowanie
 - rzędu macierzy, 25
 - unikatowych wartości, 55
 - wartości minimalnej, 54

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 

Już dziś zacznij tworzyć systemy inteligentne!

Uczenie maszynowe jest dziś wykorzystywane w różnych dziedzinach życia: w biznesie, w polityce, w organizacjach non profit i oczywiście w nauce. Samouczące się algorytmy maszynowe stanowią wyjątkową metodę przekształcania danych w wiedzę. Powstało sporo książek wyjaśniających sposób działania tych algorytmów i prezentujących nieraz spektakularne przykłady ich wykorzystania. Do dyspozycji pozostają też narzędzia przeznaczone do tego rodzaju zastosowań, takie jak biblioteki Pythona, w tym pandas i scikit-learn. Problemem pozostaje implementacja rozwiązań codziennych problemów związanych z uczeniem maszynowym.

Z tej książki najwięcej skorzystają profesjonalści, którzy znają podstawowe koncepcje związane z uczeniem maszynowym. Osoby te potraktują ją jako przewodnik ułatwiający rozwiązywanie konkretnych problemów napotykanych podczas codziennej pracy z uczeniem maszynowym. Dzięki zawartym tu recepturom takie zadania jak wczytywanie danych, obsługa danych tekstowych i liczbowych, wybór modelu czy redukcja wymiarowości staną się o wiele łatwiejsze do wykonania. Każda receptura zawiera kod, który można wstawić do swojego programu, połączyć lub zaadaptować według potrzeb. Przedstawiono także analizy wyjaśniające poszczególne rozwiązania i ich kontekst. Z tą książką płynnie przejdziesz od rozważań teoretycznych do opracowywania działających aplikacji i praktycznego korzystania z zalet uczenia maszynowego.

Dr Chris Albon — jest analitykiem danych i politologiem. Od ponad dziesięciu lat stosuje statystykę, sztuczną inteligencję i inne zdobycze informatyki w polityce, socjologii i przy zarządzaniu akcjami humanitarnymi. Obecnie pracuje dla Devoted Health — wykorzystuje naukę o danych i maszynowe uczenie w celu rozwiązania problemów amerykańskiego systemu ochrony zdrowia. Wcześniej był głównym analitykiem danych w kenijjskim startupie BRCK.

Receptury w tej książce dotyczą:

- wektorów, macierzy i tablic
- obsługi danych liczbowych i tekstowych, obrazów, a także związanych z datą i godziną
- redukcji wymiarowości za pomocą wyodrębniania i wyboru cech
- oceny i wyboru modelu oraz regresji liniowej i logistycznej

Helion 

 **helion.pl**

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

Sprawdź nasze szkolenia!

SZKOLENIA



AKADEMIA IT & BUSINESS

WWW.SZKOLENIA.HELION.PL

KOD KORZYŚCI
Sięgnij po więcej! ▶



ISBN 978-83-283-5046-5



9 788328 350465