

Dodaj mocy **Power BI!**

Jak za pomocą kodu w **Pythonie** i **R**
pobierać, przekształcać i wizualizować dane

Luca Zavarella
Przedmowa Francesca Lazzeri

Helion 



Tytuł oryginału: Extending Power BI with Python and R Ingest, transform, enrich, and visualize data using the power of analytical languages

Tłumaczenie: Radosław Meryk

ISBN: 978-83-283-9453-7

Copyright © Packt Publishing 2021. First published in the English language under the title 'Extending Power BI with Python and R – (9781801078207)'.

Polish edition copyright © 2022 by Helion S.A.

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 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/rozmoz>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<https://ftp.helion.pl/przyklady/rozmoz.zip>

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 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

Przedmowa	13
O autorze	15
O recenzentach	16
Wstęp	17

Część I. Najlepsze praktyki korzystania z języków R i Python w usłudze Power BI

Rozdział 1. Gdzie i jak używać w usłudze Power BI skryptów języka R i Python?	23
Wymagania techniczne	24
Wstrzykiwanie skryptów języka R lub Python do usługi Power BI	24
Ładowanie danych	24
Przekształcanie danych	27
Wizualizacja danych	31
Korzystanie z języków R i Python do interakcji z danymi	32
Ograniczenia stosowania języków R i Python dla różnych produktów usługi Power BI	34
Podsumowanie	36
Rozdział 2. Konfigurowanie języka R na potrzeby usługi Power BI	37
Wymagania techniczne	38
Dostępne silniki języka R	38
Dystrybucja CRAN R	38
Dystrybucja Microsoft R Open i MRAN	38

Klient Microsoft R	42
Stopniowe wycofywanie dystrybucji Microsoft R Open	42
Wybór silnika języka R do zainstalowania	43
Silniki języka R używane w usłudze Power BI	43
Instalowanie sugerowanych silników języka R	45
Instalowanie środowiska IDE w celu programowania w języku R	50
Instalacja programu RStudio	50
Konfigurowanie programu Power BI Desktop do pracy z językiem R	54
Konfigurowanie usługi Power BI do pracy z językiem R	56
Instalowanie lokalnej bramy danych w trybie osobistym	56
Udostępnianie raportów używających skryptów języka R w usłudze Power BI	60
Ograniczenia wizualizacji języka R	61
Podsumowanie	63
Rozdział 3. Konfigurowanie języka Python na potrzeby usługi Power BI	64
Wymagania techniczne	64
Dostępne silniki Pythona	65
Wybór silnika języka Python do zainstalowania	66
Silniki języka Python wykorzystywane w usłudze Power BI	66
Instalowanie sugerowanych silników języka Python	67
Instalowanie środowiska IDE na potrzeby programowania w języku Python	79
Konfigurowanie obsługi Pythona w środowisku programu RStudio	80
Konfigurowanie języka Python w środowisku programu Visual Studio Code	82
Konfigurowanie programu Power BI Desktop do pracy z językiem Python	86
Konfigurowanie usługi Power BI do pracy z językiem R	88
Udostępnianie raportów używających skryptów języka Python w usłudze Power BI	89
Ograniczenia wizualizacji w Pythonie	89
Podsumowanie	91
Część II. Pozyskiwanie i przekształcanie danych za pomocą języków R i Python w usłudze Power BI	93
Rozdział 4. Importowanie nieobsługiwanych obiektów danych	95
Wymagania techniczne	96
Importowanie plików RDS w języku R	96
Krótkie wprowadzenie do Tidyverse	96
Tworzenie zserializowanych obiektów w języku R	97
Korzystanie z plików RDS w usłudze Power BI	103
Importowanie plików PKL w Pythonie	113
Krótkie wprowadzenie do PyData	114
Tworzenie zserializowanego obiektu Pythona	115
Korzystanie z pliku PKL w usłudze Power BI	119
Podsumowanie	126
Bibliografia	127

Rozdział 5. Korzystanie z wyrażeń regularnych w usłudze Power BI	128
Wymagania techniczne	128
Krótkie wprowadzenie do regeksów	129
Podstawowe informacje o regeksach	130
Sprawdzanie poprawności adresów e-mail	136
Sprawdzanie poprawności dat	139
Sprawdzanie poprawności danych przy użyciu regeksów w usłudze Power BI	142
Korzystanie z regeksów w usłudze Power BI	
do sprawdzania poprawności adresów e-mail z wykorzystaniem języka Python	143
Korzystanie z regeksów w usłudze Power BI	
do sprawdzania poprawności adresów e-mail za pomocą języka R	146
Korzystanie z regeksów w usłudze Power BI	
do sprawdzania poprawności dat za pomocą języka Python	149
Korzystanie z regeksów w usłudze Power BI	
do sprawdzania poprawności dat za pomocą języka R	151
Ładowanie do usługi Power BI złożonych plików logów z wykorzystaniem regeksów	152
Logi dostępu do serwera Apache	153
Importowanie logów dostępu serwera Apache w usłudze Power BI	
za pomocą języka Python	154
Importowanie logów dostępu serwera Apache w usłudze Power BI	
za pomocą języka R	156
Wyodrębnianie wartości z tekstu w usłudze Power BI przy użyciu regeksów	157
Jeden regex do zarządzania wszystkimi danymi	159
Używanie regeksów w usłudze Power BI do wyodrębniania wartości	
z wykorzystaniem języka Python	160
Korzystanie z regeksów w usłudze Power BI do wyodrębniania wartości	
za pomocą języka R	162
Podsumowanie	164
Bibliografia	164
Rozdział 6. Anonimizacja i pseudonimizacja danych w usłudze Power BI	165
Wymagania techniczne	166
Deidentyfikacja danych	166
Techniki deidentyfikacji	166
Istota pseudonimizacji	170
Czym jest anonimizacja?	171
Anonimizacja danych w usłudze Power BI	172
Anonimizacja danych za pomocą języka Python	173
Anonimizacja danych za pomocą języka R	176
Pseudonimizacja danych w usłudze Power BI	180
Pseudonimizacja danych za pomocą języka Python	181
Pseudonimizacja danych w języku R	184
Podsumowanie	186
Bibliografia	186

Rozdział 7. Zapisywanie danych z usługi Power BI do źródeł zewnętrznych	187
Wymagania techniczne	187
Zapis danych do plików CSV	188
Zapis do plików CSV w języku Python	189
Zapis informacji do plików CSV za pomocą języka R	192
Zapisywanie informacji do plików programu Excel	195
Zapisywanie informacji do plików Excela za pomocą języka Python	196
Zapis do plików Excela za pomocą języka R	200
Zapis danych do serwera Azure SQL Server	204
Instalacja programu SQL Server Express	205
Tworzenie bazy danych w usłudze Azure SQL Database	207
Zapis danych do serwera Azure SQL Server za pomocą języka Python	212
Zapis danych na serwerze Azure SQL za pomocą języka R	219
Podsumowanie	223
Bibliografia	224
Rozdział 8. Ładowanie do usługi Power BI zbiorów danych przekraczających dostępną pamięć RAM	225
Wymagania techniczne	226
Typowy scenariusz analityczny obejmujący wykorzystanie obszernego zestawu danych	226
Importowanie dużych zestawów danych w języku Python	227
Instalowanie Dask na laptopie	228
Tworzenie obiektów DataFrame pakietu Dask	229
Wyodrębnianie informacji z obiektu DataFrame pakietu Dask	230
Importowanie dużego zestawu danych w usłudze Power BI z wykorzystaniem języka Python	235
Importowanie dużych zestawów danych za pomocą języka R	237
Instalowanie pakietu disk.frame na laptopie	238
Tworzenie egzemplarza obiektu disk.frame	238
Wyodrębnianie informacji z obiektu disk.frame	240
Importowanie dużego zestawu danych w usłudze Power BI za pomocą języka R	242
Podsumowanie	244
Bibliografia	244
Część III. Wzbogacanie danych w usłudze Power BI za pomocą języków R i Python	245
Rozdział 9. Wywoływanie zewnętrznych interfejsów API w celu wzbogacania danych	247
Wymagania techniczne	248
Czym jest usługa sieciowa?	248
Rejestrowanie się w usłudze Bing Maps Web Services	249
Geokodowanie adresów z wykorzystaniem języka Python	251
Korzystanie z jawnego żądania GET	251
Korzystanie ze współbieżnych żądań GET	255
Korzystanie z biblioteki Geocoder w trybie programowania współbieżnego	257

Geokodowanie adresów za pomocą języka R	258
Korzystanie z jawnych żądań GET	259
Korzystanie ze współbieżnych żądań GET	261
Korzystanie z pakietu tidygeocoder do współbieżnego wykonywania kodu	262
Korzystanie z usług sieciowych z poziomu usługi Power BI	263
Geokodowanie adresów w usłudze Power BI za pomocą języka Python	265
Geokodowanie adresów w usłudze Power BI za pomocą języka R	265
Podsumowanie	266
Bibliografia	267
Rozdział 10. Obliczanie kolumn przy użyciu złożonych algorytmów	268
Wymagania techniczne	269
Obliczanie odległości między dwiema lokalizacjami geograficznymi	269
Trygonometria sferyczna	269
Twierdzenie cosinusów	271
Wzór Haversinesa	272
Wzór Vincenty'ego	273
Jakiego wzoru odległości używać i kiedy?	274
Implementacja algorytmów wyznaczania odległości w Pythonie	275
Obliczanie odległości w języku Python	275
Obliczanie odległości w usłudze Power BI za pomocą języka Python	278
Implementacja algorytmów wyznaczania odległości w języku R	279
Obliczanie odległości za pomocą języka R	279
Obliczanie odległości w usłudze Power BI za pomocą języka R	282
Podstawy programowania liniowego	283
Równania i nierówności liniowe	283
Formułowanie problemu optymalizacji liniowej	285
Definicja problemu LP do rozwiązania	288
Formułowanie problemu LP	289
Rozwiązywanie problemów optymalizacji w języku Python	291
Rozwiązywanie problemu LP za pomocą języka Python	291
Rozwiązywanie problemu programowania liniowego	
za pomocą języka Python w usłudze Power BI	295
Rozwiązywanie problemów programowania liniowego za pomocą języka R	300
Rozwiązywanie problemu LP w języku R	300
Rozwiązywanie problemu programowania liniowego	
za pomocą języka R w usłudze Power BI	303
Podsumowanie	305
Bibliografia	306
Rozdział 11. Dodawanie statystyk: powiązania	307
Wymagania techniczne	307
Badanie powiązań między zmiennymi	308
Korelacja między zmiennymi liczbowymi	308
Współczynnik korelacji Karla Pearsona	310
Współczynnik korelacji Charlesa Spearmana	313
Współczynnik korelacji Maurice'a Kendalla	316
Opis przypadku	316

Implementacja obliczeń współczynników korelacji w Pythonie	317
Implementacja obliczeń współczynników korelacji w języku R	320
Implementacja obliczania współczynników korelacji w usłudze Power BI za pomocą języków Python i R	323
Korelacje między zmiennymi kategorycznymi a liczbowymi	325
Związek między dwiema zmiennymi kategorycznymi	325
Związek pomiędzy zmiennymi liczbowymi a kategorycznymi	329
Implementacja obliczania współczynników korelacji w Pythonie	332
Implementacja obliczania współczynników korelacji w języku R	334
Implementacja obliczania współczynników korelacji w usłudze Power BI za pomocą języków R i Python	337
Podsumowanie	338
Bibliografia	339
Rozdział 12. Dodawanie statystyk: wartości odstające i wartości brakujące	340
Wymagania techniczne	340
Czym są wartości odstające i jak sobie z nimi radzić?	341
Przyczyny istnienia wartości odstających	342
Obsługa wartości odstających	342
Identyfikacja wartości odstających	343
Jednowymiarowe wartości odstające	343
Wielowymiarowe wartości odstające	344
Implementacja algorytmów wykrywania wartości odstających	348
Implementacja wykrywania wartości odstających w Pythonie	349
Implementacja wykrywania wartości odstających w języku R	355
Implementacja wykrywania wartości odstających w usłudze Power BI	358
Czym są wartości brakujące i jak sobie z nimi radzić?	359
Przyczyny brakujących wartości	360
Obsługa wartości brakujących	361
Diagnozowanie brakujących wartości z wykorzystaniem języków R i Python	366
Implementacja algorytmów imputacji brakujących wartości	369
Usuwanie brakujących wartości	370
Imputacja danych tabelarycznych	370
Imputacja danych szeregów czasowych	373
Imputacja brakujących wartości w usłudze Power BI	374
Podsumowanie	376
Bibliografia	376
Rozdział 13. Korzystanie z uczenia maszynowego bez licencji Premium lub Embedded	378
Wymagania techniczne	379
Interakcje z mechanizmami uczenia maszynowego w usłudze Power BI z wykorzystaniem przepływów danych	379
Korzystanie z rozwiązań AutoML	381
PyCaret	382
Azure AutoML	383
RemixAutoML dla języka R	383

Osadzanie kodu szkoleniowego w dodatku Power Query	384
Szkolenie i korzystanie z modeli ML z wykorzystaniem biblioteki PyCaret	384
Korzystanie z biblioteki PyCaret w usłudze Power BI	387
Korzystanie z przeszkolonych modeli w dodatku Power Query	388
Ocena obserwacji w dodatku Power Query przy użyciu przeszkolonego modelu PyCaret	389
Korzystanie z przeszkolonych modeli w wizualizacjach utworzonych za pomocą skryptów	390
Ocenianie obserwacji w skrypcie wizualizacji z wykorzystaniem przeszkolonego modelu PyCaret	392
Wywoływanie usług sieciowych w dodatku Power Query	401
Korzystanie z usług Cognitive Services w dodatku Power Query	408
Podsumowanie	411
Bibliografia	411

Część IV. Wizualizacja danych za pomocą języka R w usłudze Power BI

Rozdział 14. Eksploracyjna analiza danych

Wymagania techniczne	415
Jaki jest cel EDA?	416
Zrozumienie sensu danych	416
Oczyszczanie danych	417
Odkrywanie powiązań między zmiennymi	418
Analiza EDA z wykorzystaniem języków R i Python	418
Analiza EDA w usłudze Power BI	420
Strona podsumowania zestawu danych	421
Eksploracja brakujących wartości	424
Eksploracja jednowymiarowa	425
Eksploracja wielowymiarowa	433
Powiązania zmiennych	438
Podsumowanie	442
Bibliografia	442

Rozdział 15. Zaawansowane wizualizacje

Wymagania techniczne	443
Tworzenie wykresu kołowego	443
Implementacja kołowego wykresu słupkowego w języku R	446
Implementacja kołowego wykresu słupkowego w usłudze Power BI	452
Podsumowanie	453
Bibliografia	454

Rozdział 16. Interaktywne niestandardowe wizualizacje języka R	455
Wymagania techniczne	455
Zalety stosowania interaktywnych niestandardowych wizualizacji w języku R	456
Dodawanie elementów interaktywnych za pomocą biblioteki Plotly	458
Wykorzystanie elementów interaktywnych za pośrednictwem widżetów HTML	460
Tworzenie niestandardowych wizualizacji w usłudze Power BI	460
Instalowanie pakietu pbiviz	461
Tworzenie pierwszej niestandardowej wizualizacji R HTML	462
Importowanie pakietu niestandardowej wizualizacji do usługi Power BI	471
Podsumowanie	474
Bibliografia	475
Skorowidz	476

Korzystanie z wyrażeń regularnych w usłudze Power BI

Wiele zadań czyszczenia danych obejmuje przeprowadzanie złożonych wyszukiwań i podstawień między ciągami znaków. Aby uzyskać pożądane wyniki, czasami nie wystarczą zwykłe narzędzia wyszukiwania i zastępowania. Załóżmy, że musisz dopasować ciągi znaków nie w sposób dokładny (na przykład z wykorzystaniem warunków równości), ale przy użyciu kryteriów podobieństwa między nimi. W projektach wymagających wysokiej jakości danych umiejętność korzystania z narzędzi takich jak wyrażenia regularne (znanych również jako regeksy) może mieć ogromne znaczenie. Dzięki językom R i Python możesz dodać te narzędzia do swojego arsenału.

W tym rozdziale omówię następujące tematy:

- Krótkie wprowadzenie do regeksów.
- Sprawdzanie poprawności danych przy użyciu regeksów w Power BI.
- Ładowanie złożonych plików logów przy użyciu regeksów w Power BI.
- Wyodrębnianie wartości z tekstu przy użyciu regeksów w Power BI.

Wymagania techniczne

Aby skutecznie studiować treść tego rozdziału, będziesz potrzebować połączenia z internetem i zainstalowanego na komputerze programu Power BI Desktop. Musisz poprawnie skonfigurować silniki języków R i Python oraz środowiska IDE, zgodnie z opisem w rozdziałach:

2., „Konfigurowanie języka R na potrzeby usługi Power BI” i 3., „Konfigurowanie języka Python na potrzeby usługi Power BI”.

Krótkie wprowadzenie do regeksów

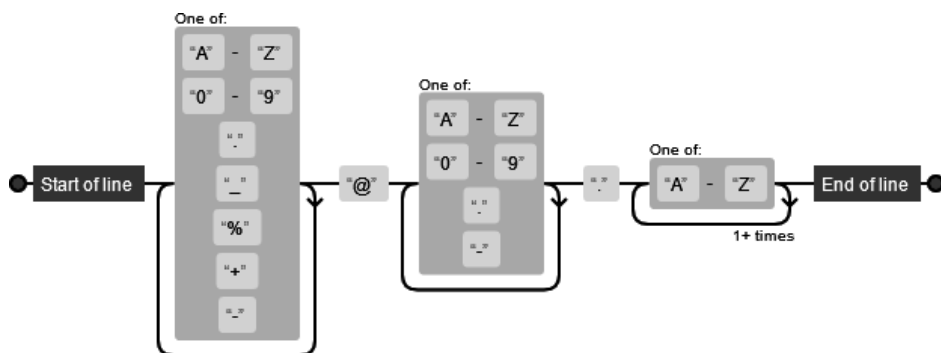
Wyrażenie regularne (zwykle określane skrótową nazwą **regex**) to ciąg znaków, które *identyfikują abstrakcyjny wzorzec wyszukiwania*. Ogólnie rzecz biorąc, jest to technika matematyczna, która została opracowana w 1951 roku przez ekspertów języka formalnego i informatyki teoretycznej. Służy do sprawdzania poprawności danych wejściowych lub wyszukiwania i wyodrębniania informacji z tekstów.

Jeśli nie znasz składni regeksów, na pierwszy rzut oka może ona sprawiać wrażenie bardzo złożonej (rysunek 5.1):

```
^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$
```

Rysunek 5.1. Przykład wzorca regeksa

Na szczęście istnieją internetowe narzędzia do wizualizacji regeksów, które ułatwiają zrozumienie tych wzorców (jeden z nich jest dostępny na stronie <https://regexper.com>). Na przykład regex wyróżniony na rysunku 5.1 można zwizualizować w następujący sposób (rysunek 5.2):



Rysunek 5.2. Wizualizacja regeksa

Na rysunku 5.2 widać, że regex z rysunku 5.1 pozwala zidentyfikować we fragmencie tekstu adresy e-mail.

Nauczenie się, jak korzystać z regeksów w sposób profesjonalny, z pewnością nie jest łatwe i nie jest celem tego podrozdziału. Wyjaśnię tutaj jedynie podstawowe zasady, które pozwolą Ci tworzyć proste, ale skuteczne wzorce wyszukiwania. Aby uzyskać więcej informacji, zapoznaj się z bibliografią na końcu tego rozdziału.

Podstawowe informacje o regeksach

Podstawowe zasady tworzenia regeksów postaramy się wyjaśnić za pomocą przykładów, co jest prawdopodobnie najbardziej bezpośrednim sposobem na rozpoczęcie ich używania. W każdym kolejnym podpunkcie wyjaśnię inną funkcję regeksów. Aby przetestować regex użyty w przykładzie, posłużymy się narzędziem dostępnym na stronie <https://www.regexpal.com/>. Zaczynamy!

Znaki literalne

Aby dołączyć jeden lub więcej znaków literalnych do regeksa, konieczne jest skorzystanie z funkcji „wyszukiwania”. Spróbujmy wyszukać ciąg *ie* wewnątrz tekstu *Niech moc rozszerzenia Power BI za pomocą Pythona i R będzie z Tobą!* (rysunek 5.3):



Rysunek 5.3. Wyszukiwanie ciągu „ie” za pomocą regeksów

Zauważ, że narzędzie domyślnie używa flagi wyszukiwania globalnego. Konkretnie flagi są oznaczane literami (w naszym przypadku g) zaraz za ogranicznikami regeksa, `/.../`. Oto lista dostępnych flag, z których możesz skorzystać:

- g (globalne): powoduje dopasowywanie wszystkich wystąpień z zachowaniem indeksu wystąpienia ostatniego.
- m (wielowierszowe — ang. *multiline*): gdy ta opcja jest włączona, kotwice ciągów (omówię je później) będą dopasowywane do początku i końca wiersza, a nie do całego ciągu.
- i (ignoruj wielkość liter): powoduje przeszukiwanie wzorca niezależnie od wielkości liter w ciągu (małych lub wielkich).

Należy zapamiętać, że nie we wszystkich językach programowania wykorzystywana jest wspomniana wcześniej składnia flag. Na przykład pakiet `re` dostępny dla Pythona (domyślny dla regeksów) pozwala na stosowanie parametrów w funkcjach `search`, `match` i `sub`:

```
re.search('test', 'TeSt', re.IGNORECASE)
re.match('test', 'TeSt', re.IGNORECASE)
re.sub('test', 'xxx', 'TeSting', flags=re.IGNORECASE)
```

To samo dotyczy funkcji `regex()` z pakietu `stringr` języka R:

```
str_detect('tEsTuj to', regex('test', ignore_case=TRUE))
```

Można także korzystać z **modyfikatorów globalnych** bezpośrednio we wzorcu regeksa (w tzw. trybie *inline*). Są to modyfikatory: `(?i)` pozwalający określić ignorowanie wielkości liter i `(?m)` oznaczający ciągi wielowierszowe. Na przykład w języku R możesz uruchomić następujący skrypt:

```
str_detect('tEsTuj to', regex("(?i)test"))
```

Warto zwrócić uwagę, że Python nie pozwala na stosowanie modyfikatorów globalnych w trybie *inline*.

Znaki specjalne

W regeksach można korzystać z 12 znaków specjalnych (zwanych również **metaznakami**). Każdy z nich ma specjalne znaczenie. Są to znaki potoku (`|`); lewego ukośnika (`\`); dolara (`$`); znaku zapytania (`?`); karetki (`^`); gwiazdki (`*`); plusa (`+`); kropki (`.`); nawiasy zwykłe (`( )`); otwierający nawias prostokątny (`[`) oraz otwierający nawias klamrowy (`{`).

Jeśli chcesz wyszukać jeden z wcześniej wymienionych znaków, musisz skorzystać z tzw. **znaku ucieczki** (ang. *escape character*), którym jest znak lewego ukośnika. Tak więc jeśli chcesz dokładnie dopasować ciąg `123$`, musisz użyć wzorca regex w postaci `123\$`.

Następnie zapoznamy się ze znaczeniem metaznaków.

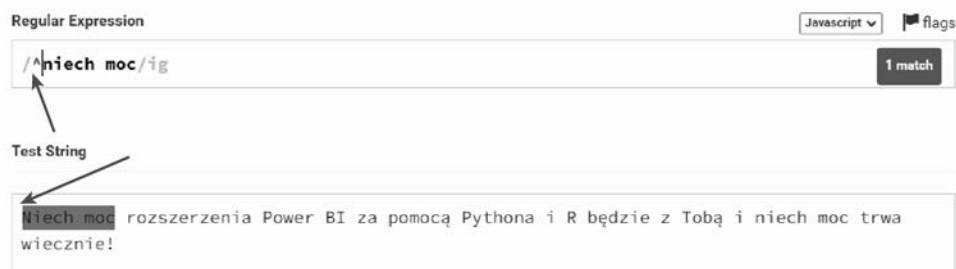
Znaki zakotwiczenia `^` i `$`

Znaki zakotwiczenia mają specjalne znaczenie, ponieważ są używane do umieszczania dopasowania regeksa w określonej pozycji w ciągu. Znak karetki (`^`) służy do wskazania początku ciągu (lub wiersza), a znak dolara (`$`) służy do wskazania końca ciągu (lub wiersza). Przykładowa ilustracja jest warta tysiąca słów (rysunek 5.4):



Rysunek 5.4. Wyszukiwanie globalne bez uwzględniania wielkości liter

W przykładzie pokazanym na rysunku 5.4 została ustawiona flaga „ignoruj wielkość liter”. Aby to zrobić, należy kliknąć ikonę *flags*, a następnie zaznaczyć opcję *ignore case*. Dzięki temu regex znajdzie oba wystąpienia dopasowania. Spróbujmy teraz dodać znak karetki (`^`) przed znakiem *n* (rysunek 5.5):



Rysunek 5.5. Wyszukiwanie globalne bez uwzględnienia wielkości liter oraz z zastosowaniem symbolu karetki (^)

W tym przypadku regex dopasuje tylko pierwsze wystąpienie (czyli to na początku ciągu).

Jeśli dodasz również znak dolara na końcu regeksa, to nie znajdzie on żadnego dopasowania. Byłoby to bowiem żądanie dopasowania ciągu *niech moc*, który jednocześnie rozpoczyna i kończy tekst.

Operatory OR

Czasami może zająć konieczność dopasowania jednego z wybranych zbiorów znaków lub ciągów. Na przykład, aby dopasować dowolny ze znaków *s* lub *t* za ciągiem *ko*, powinieneś użyć klasy znaków `[st]` wewnątrz regeksa `ko[st]`. Dzięki temu regex będzie pasował zarówno do ciągów *kos*, jak i *kot*. Klasy znaków mogą być również wykorzystywane do dopasowywania wystąpienia w zakresach znaków, które tworzymy za pomocą łącznika (-). Na przykład klasa `[0-9]` odpowiada pojedynczej cyfrze z zakresu od 0 do 9, podczas gdy klasa `[A-Z]` dopasowuje pojedynczą wielką literę z zakresu od A do Z. Możesz także łączyć wiele zakresów w jedną klasę znaków. Na przykład klasa `[A-Z0-9]` pasuje do cyfry lub do wielkiej litery.

Aby za pomocą regeksa dopasować jeden z dwóch ciągów znaków, możesz użyć symbolu potoku (`|`), za pomocą którego oddziolisz ciągi umieszczone w nawiasach okrągłych, na przykład `(ciąg1|ciąg2)`. Oto przykład (rysunek 5.6):



Rysunek 5.6. Przykład zastosowania operatorów OR

Klasę znaków można również wykorzystać do dopasowania dowolnego znaku innego niż określony znak. Można to uzyskać dzięki zanegowaniu klasy znaków.

Zanegowane klasy znaków

Karetką występująca tuż za otwierającym nawiasem kwadratowym neguje zawartość klasy znaków. Na przykład regex `[^]` pasuje do każdego znaku, który nie jest cudzysłowem.

Skróty klas znaków

Istnieją pewne klasy znaków, które są używane bardzo często. Z tego powodu zdefiniowano kilka skrótów pozwalających na ich szybkie włączenie do regeksów. Oto lista najczęściej używanych:

- `\w`: pasuje do wszystkich znaków alfanumerycznych oraz znaku podkreślenia.
- `\W`: przeciwieństwo klasy `\w`; pasuje do pojedynczego znaku niealfanumerycznego z wyłączeniem znaku podkreślenia. Na przykład pozwala dopasować spacje i znaki interpunkcyjne.
- `\d`: pasuje do pojedynczej cyfry.
- `\D`: przeciwieństwo klasy `\d`; pasuje do pojedynczego znaku niebędącego cyfrą.
- `\s`: pasuje do „znaków białych spacji”, takich jak spacja, tabulator, znak podziału wiersza i powrót karetki.

W tym rozdziale będziemy dość często korzystać ze skrótów klas znaków.

Kwantyfikatory

Kwantyfikatory wskazują, ile razy musi zostać dopasowany *znak* lub *wyrażenie*. Oto lista najczęściej używanych kwantyfikatorów:

- `+`: pasuje do ciągu poprzedzającego kwantyfikator jeden lub więcej razy. Na przykład regex `test\d+` będzie pasować do ciągu `test`, za którym występuje jedna lub więcej cyfr. Regex `test(-\d\d)+` będzie z kolei pasował do ciągu `test`, po którym występuje jeden lub więcej razy myślnik, a za nim dwie cyfry (rysunek 5.7):

Regular Expression

```
/test(-\d\d)+/g
```

Test String

```
test-01 to co innego niż test-23-456|
```

Rysunek 5.7. Powtarzanie grupy znaków za pomocą kwantyfikatora `+`

- `{n}`: pasuje do ciągu poprzedzającego ten symbol n razy. Na przykład regex `\d{4}` będzie pasować do dowolnej liczby całkowitej złożonej z 4 cyfr.
- `{n,m}`: pasuje do ciągu poprzedzającego ten kwantyfikator od n do m razy. Na przykład regex `prod-\d{2,6}` będzie pasować do ciągu *prod-*, za którym występuje liczba całkowita składająca się z od 2 do 6 cyfr.
- `{n,}`: pasuje do ciągu poprzedzającego ten kwantyfikator n lub więcej razy.
- `?`: pasuje do ciągu poprzedzającego ten kwantyfikator zero razy lub raz. Na przykład regex `Mar(zec)?` będzie pasować zarówno do ciągu *Marzec*, jak i *Mar*. Z kolei regex `pisana?nki` będzie pasował zarówno do ciągów *pisanki*, jak i *pisaki*.
- `*`: pasuje do ciągu poprzedzającego ten kwantyfikator zero lub więcej razy. Na przykład regex `kod\d*` będzie zgodny z ciągiem *kod*, *kod1*, a także *kod73846*.

Kropka

Kropka odpowiada pojedynczemu znakowi, niezależnie od tego, czym jest ten znak, z wyjątkiem znaków podziału wiersza. Jest to bardzo potężny metaznak stosowany w regeksach, który może nas rozleniwic. Właśnie dlatego trzeba uważać, aby go nie nadużywać, ponieważ czasami może to doprowadzić do uwzględnienia w dopasowaniach niezamierzonych wyników.

Dopasowania zachłanne i leniwe

Symbole `+`, `*` oraz powtórzenie symboli `{...}` to **kwantyfikatory zachłanne** (ang. *greedy quantifiers*). Zachłanność polega na tym, że kwantyfikatory te *skonsumują najdłuższy możliwy ciąg*. Załóżmy, że chcesz dopasować same tagi użyte w ciągu `<em>Power BI wymiata</em>`. Początkujący użytkownik regeksów mógłby użyć regeksa `<.+>`, który można wyrazić takimi oto słowami: „zdobądź `<`, następnie raz lub więcej razy uzyskaj dowolny znak niebędący znakiem podziału wiersza, a na koniec uzyskaj znak `>`”. Oczekiwany wynik to dwa dopasowania `<em>` i `</em>`. Rzućmy okiem na wynik (rysunek 5.8):

Regular Expression

```
/<.+>/g
```

Test String

```
<em>Power BI wymiata</em>
```

Rysunek 5.8. Chciwość symbolu `.+`

Jest oczywiste, że kombinacja `.*+` obejmuje wszystko, co zawiera się między pierwszym wystąpieniem znaku `<` a ostatnim wystąpieniem znaku `>`, stąd bierze się definicja **chciwości** kwantyfikatorów.

Czy można więc zmusić chciwy kwantyfikator do zatrzymania się przy pierwszym wykrytym wystąpieniu następnego znaku, uniemożliwiając mu „zjedzenie” czegokolwiek aż do ostatniego wystąpienia tego samego znaku? Innymi słowy, czy można zamienić chciwy kwantyfikator na **leniwy**? Odpowiedź brzmi „tak”, można to zrobić, dodając metaznak `?` bezpośrednio za znakiem `+`. Tak więc regex `<.*+` należy przekształcić na `<.*+?`. Oto wynik (rysunek 5.9):

Regular Expression

```
/<.*+?>/g
```

Test String

```
<em>Power BI wymiata</em>
```

Rysunek 5.9. Przekształcenie chciwego kwantyfikatora w leniwy dzięki metaznakowi `?`

Należy jednak pamiętać, że *leniwe kwantyfikatory są niewydajne*. O ile to możliwe, zawsze lepiej jest używać zanegowanych klas znaków. W naszym przykładzie wystarczy użyć regexa `<[^>]+>` (czyli znak `<`, dowolny znak inny niż `>` jeden lub więcej razy oraz znak `>`). Taki regex pozwala osiągnąć ten sam wynik, a jednocześnie zużyje mniej zasobów obliczeniowych (rysunek 5.10):

Regular Expression

```
/<[^>]+>/g
```

Test String

```
<em>Power BI wymiata</em>
```

Rysunek 5.10. Wykorzystanie zanegowanej klasy znaków zamiast leniwego kwantyfikatora

Dzięki temu, czego dotychczas nauczyłeś się o regeksach, masz minimalne podstawy wymagane do zrozumienia bardziej złożonych regeksów, których użyjemy w następnych kilku przykładach.

Sprawdzanie poprawności adresów e-mail

Gdyby ktoś Cię poprosił o zweryfikowanie adresu e-mail przy użyciu pojęć poznanych przed chwilą, jedna z Twoich pierwszych prób mogłaby wyglądać następująco: `^.+@.+\..+$.` Oto próba przetłumaczenia tego rekexa na język mówiony:

1. `^`: pasuje do początku ciągu lub wiersza, jeśli jest włączona flaga wielowierszowa.
2. `.+`: pasuje do dowolnego znaku, z wyjątkiem znaku podziału wiersza, występującego jeden lub więcej razy.
3. `@`: pasuje do znaku `@`.
4. `.+`: pasuje do dowolnego znaku, z wyjątkiem znaku podziału wiersza, występującego jeden lub więcej razy.
5. `\.`: Pasuje do znaku `.`
6. `.+`: pasuje do dowolnego znaku, z wyjątkiem znaku podziału wiersza, występującego jeden lub więcej razy.

Oczywiście ten regex potwierdzi poprawność adresu e-mail. Ale czy na pewno pozwoli wykryć oczywiste błędy składniowe nieprawidłowych adresów e-mail? Spróbuj przeprowadzić test na stronie <https://www.regexpal.com/> z niewłaściwym adresem e-mail `example@example.c` (domena najwyższego poziomu, czyli część domeny po kropce, musi zawierać co najmniej dwa znaki) (rysunek 5.11):



Regular Expression

`/^.+@.+\..+$/g`

Test String

`example@example.c`

Rysunek 5.11. Wykorzystanie prostego rekexa do sprawdzenia poprawności nieprawidłowego adresu e-mail

Cóż, nie tego się spodziewaliśmy: ewidentnie błędny e-mail przeszedłby jako poprawny. Z tego powodu często konieczne jest stosowanie bardziej złożonych regleksów, które są zgodne z dobrze zdefiniowanymi regułami składniowymi.

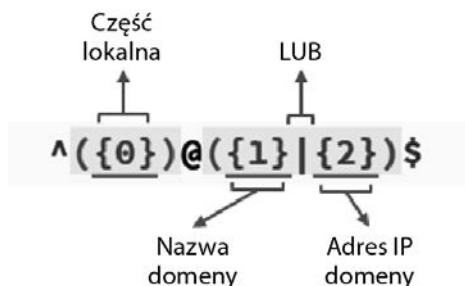
W tym przypadku użyjemy do sprawdzenia poprawności adresu e-mail konkretnego regaksa, który często stosujemy w środowisku produkcyjnym. Ten regex sprawdza również, czy używany jest adres IP domeny. Oto regex do sprawdzania poprawności adresów e-mail w pełnej postaci:

$$\wedge(([\text{^<}] [\backslash] \backslash . , ; : \backslash s @ ["] + (\backslash . [\text{^<}] ([\backslash] \backslash . , ; : \backslash s @ ["]) *) | (\text{" " } . \text{" " }) @ (([? [0-9] \{1,3\} \backslash . [0-9] \{1,3\} \backslash . [0-9] \{1,3\} \backslash . [0-9] \{1,3\}] ?) | (([a-zA-Z - 0-9] + \backslash .) + [a-zA-Z] \{2,\}) \$$$

Na pierwszy rzut oka ten regex z pewnością wprowadza w zakłopotanie. Jeśli jednak spróbujemy wydzielić z niego istotne części, staje się znacznie bardziej czytelny.

Jeśli jednak uważasz, że ten regex jest zbyt złożony, spójrz na ten, który uwzględnia wszystkie zasady składniowe opisane w dokumencie *Standard for the Format of Arpa Internet Text Messages*. Można go znaleźć pod adresem <http://www.ex-parrot.com/pdw/Mail-RFC822-Address.html>! Czyż nie robi wrażenia?

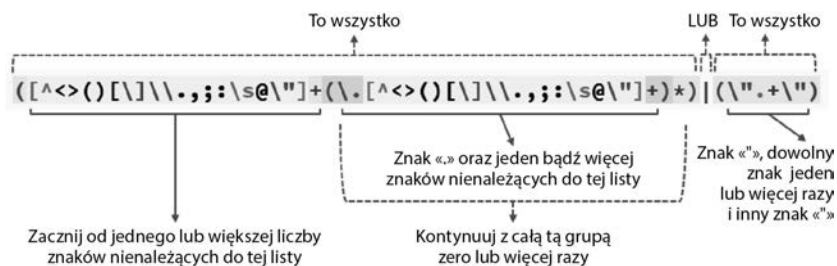
Format adresu e-mail można zdefiniować jako *część-lokalna@domena*. Pełną specyfikację można znaleźć na Wikipedii, pod adresem https://en.wikipedia.org/wiki/Email_address. Spróbujmy dopasować minimalną liczbę reguł (nie wszystkie!), które pozwolą zweryfikować znaczącą liczbę różnych adresów e-mail. Tak więc, biorąc pod uwagę, że zamierzamy dopasować nazwę domeny lub adres IP domeny, ogólna struktura regeksa jest następująca (rysunek 5.12):



Rysunek 5.12. Struktura złożonego regeksa do sprawdzania poprawności adresów e-mail

Ciągi $\{0\}$, $\{1\}$ i $\{2\}$ widoczne na rysunku 5.11 są jedynie symbolami zastępczymi, a nie faktycznymi fragmentami regleksa do dopasowania. Zaczniemy od zdefiniowania każdego z wykorzystanych tokenów:

1. Token `{0}` pasuje do regaksa dopasowujacego **lokalna część** adresu e-mail. W tym przypadku o wiele latwiej jest wyjasnic, co robi ten regex za pomoca rysunku, a nie slowami. Wszystkie szczegoly elementow skladowych tego regaksa zostaly wyjasnione na rysunku 5.13:

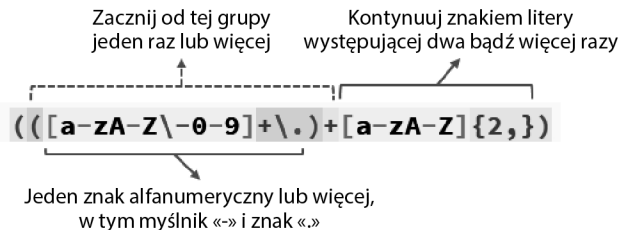


Rysunek 5.13. Szczegółowe wyjaśnienie regeksa części lokalnej

Zapamiętaj, że *nawiasy zwykle grupują poszczególne człony wzorca regeksa*. Nawiasy umożliwiają zastosowanie operatorów regeksów do całego wyrażenia grupy i uzyskanie części dopasowania jako oddzielnego elementu w tablicy wyników. Tekst odpowiadający wyrażeniu regeksa jest w nich przechwytywany jako **numerowana grupa** i może być ponownie użyty wraz z odwołaniem liczbowym. Sposób korzystania z tej własności omówię później.

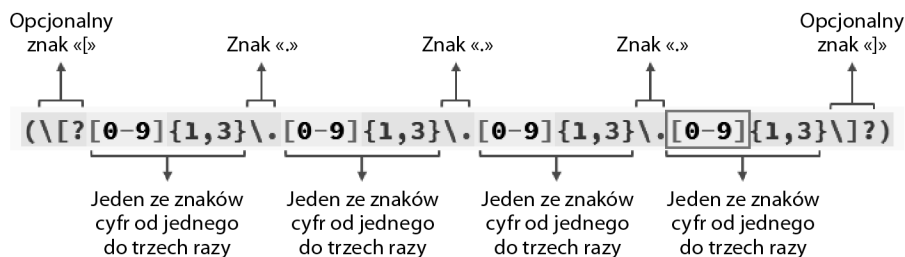
Warto zapamiętać, że aby dopasowanie metaznaku rzeczywiście reprezentowało metaznak, trzeba przed tym metaznakiem umieścić symbol ucieczki w postaci lewego ukośnika. Tak więc na przykład sekwencja `\]` oznacza znak `]`.

2. Token {1} dopasowuje nazwę domeny adresu e-mail. Aby wyjaśnić, co robi regex nazwy domeny, ponownie użyj rysunku (rysunek 5.14):



Rysunek 5.14. Szczegółowe wyjaśnienie regęksa nazwy domeny

3. Token {2} odpowiada adresowi IP domeny adresu e-mail. Jest to łatwiejszy podregex. Jego szczegółowe wyjaśnienie zamieściłem na rysunku 5.15:



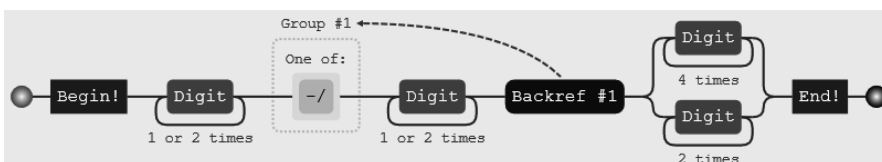
Rysunek 5.15. Szczegółowe wyjaśnienie regaksa adresu IP domeny

- \1: **odwołanie wsteczne** do grupy 1. Jak wyjaśniłem w poprzednim punkcie, nawiasy zwykle pomagają dopasować część ciągu, do którego można się odwoływać w dalszej części regeksa. W tym przypadku składnia \1 wskazuje, że w tej pozycji można oczekiwać tej samej części ciągu, jaka została dopasowana za pomocą pierwszej pary nawiasów. Pamiętaj, że w języku Python należy użyć składni \g<1> zamiast \1.
- (?: ...): jest to tak zwana *grupa nieprzechwytyjąca* (ang. *non-capturing group*). Czasami potrzebujemy nawiasów, aby poprawnie zastosować kwantyfikator, ale nie chcemy, aby ich zawartość była zgłaszana w wynikach.

Oto opis poszczególnych fragmentów powyższego regeksa:

1. ^: pasuje do początku ciągu lub wiersza, jeśli jest włączona flaga wielowierszowa.
2. \d{1,2}: pasuje do dowolnej cyfry występującej raz lub dwa razy.
3. ([\-/]): pasuje do dowolnego znaku między znakiem - a / i przechwytyje wynik jako grupę 1.
4. \d{1,2}: pasuje do dowolnej cyfry występującej raz lub dwa razy.
5. \1: odwołanie wsteczne do przechwyconej grupy 1. Pasuje więc do dowolnego znaku między znakiem - a /.
6. .+: pasuje do dowolnego znaku, z wyjątkiem znaku podziału wiersza, występującego jeden raz lub więcej razy.
7. (?: \d{4} | \d{2}): odpowiada jednej z następujących dwóch alternatyw: dowolnej cyfrze powtórzonej dokładnie cztery razy lub dowolnej cyfrze powtórzonej dokładnie dwa razy.

Powyższy regex można zwizualizować w następujący sposób (rysunek 5.17):



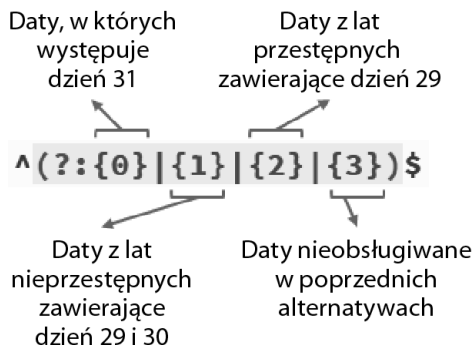
Rysunek 5.17. Wizualizacja pierwszej wersji regeksa do sprawdzania poprawności dat

Jak można się domyślić, ten regex sprawdza poprawność dat w następujących formatach: dd-mm-rrrr, dd-mm-rrrr, d-m-rrrr i d-m-rr z separatorem / lub -. Nie uwzględnia jednak błędów spowodowanych nieprawidłowymi datami, takimi jak 30 lutego lub 31 września.

Regex, który uwzględnia te błędy, ma następującą postać:

```
^(?: (? : 31 [\-/] (?: (? : 0? [13578]) | (1 [02])) [\-/] (19 | 20) ? \d \d ) | (?: (? : 29 | 30)
[\-/] (?: (? : 0? [13-9]) | (? : 1 [0-2])) [\-/] (?: 19 | 20) ? \d \d ) | (?: 29 [\-/] 0 ? 2
[\-/] (?: 19 | 20) (?: (? : [02468] [048]) | (?: [13579] [26])) ) | (?: (? : (? : 0? [1-9]) | (? : 1 \d ) | (? : 2 [0-8]))
[\-/] (?: (? : 0? [1-9]) | (? : 1 [0-2])) [\-/] (?: 19 | 20) ? \d \d ) )$
```

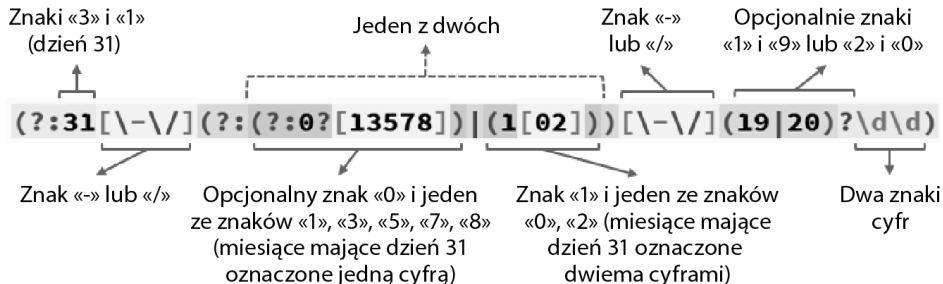
Regex zapisany w tej postaci jest bardzo trudny do interpretacji. Jednak jeśli spróbujemy przyrzeć mu się dokładniej, możemy zauważyć, że składa się on z czterech alternatyw (rysunek 5.18):



Rysunek 5.18. Struktura złożonego regeksa do walidacji daty

Ponadto w tym przypadku ciągi `{0}`, `{1}`, `{2}` i `{3}` są jedynie symbolami zastępczymi, a nie znakami do dopasowania. Spróbujmy zdefiniować kolejne tokeny, począwszy od tokena `{0}`.

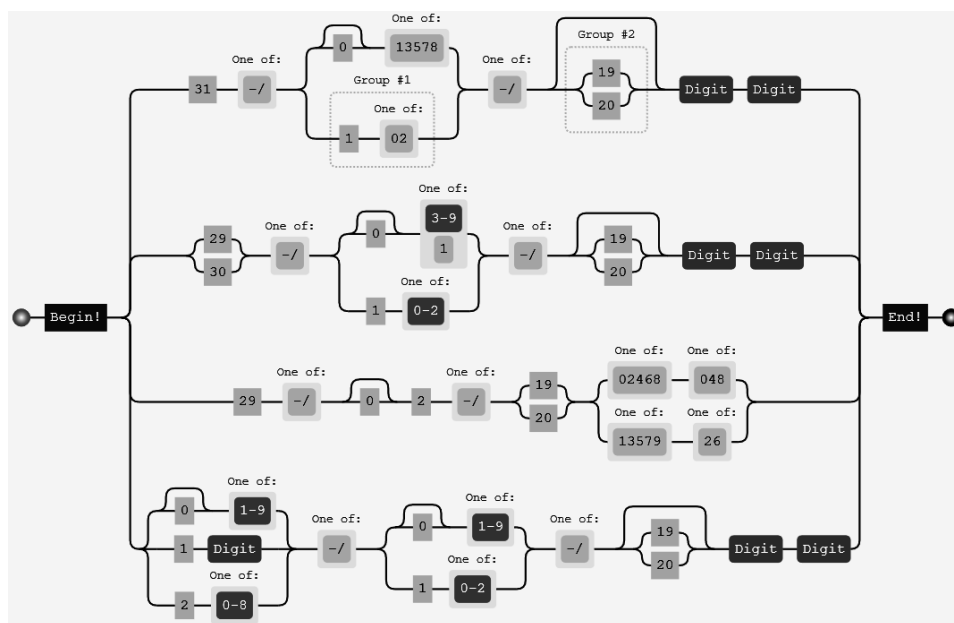
Token `{0}` odpowiada datom, które mają 31. dzień. Podobnie jak w poprzednich przypadkach, znacznie łatwiej jest wyjaśnić, co ten regex robi, na rysunku zamiast słowami. Poszczególne części regeksa zostały wyjaśnione za pomocą etykiet umieszczonych na rysunku 5.19:



Rysunek 5.19. Szczegółowy opis części regeksa dla dat, które mają dzień 31

Regexy odpowiadające innym symbolom zastępczym są bardzo podobne do tego, który właśnie wyjaśniliśmy. Dlatego pozostawiam zapoznanie się z ich wyjaśnieniem jako ćwiczenie dla Czytelnika. Jeśli chcesz zobaczyć wizualizację całego regeksa, zapoznaj się z poniższym rysunkiem pochodzącym ze strony <https://jex.im/regex> (rysunek 5.20).

Regex, który właśnie przeanalizowaliśmy, umożliwia walidację formatu `dd-mm-rrrr` ze wszystkimi jego wariantami. Sposób implementacji mechanizmu sprawdzania poprawności dat w usłudze Power BI zademonstruję w kodzie. Dodatkowo znajdziesz regeksy, które umożliwiają sprawdzanie poprawności dat w formatach `mm-dd-rrrr` i `rrrr-mm-dd` ze wszystkimi ich wariantami (gdzie rok składa się z dwóch cyfr; miesiąc składa się z jednej cyfry i tak dalej).



Rysunek 5.20. Wizualizacja całego regeksa sprawdzania poprawności dat

Teraz, gdy już wiesz, co kryje się za złożonymi regeksami przedstawionymi wcześniej, przejdźmy do ich implementacji w usłudze Power BI, gdzie zostaną wykorzystane do sprawdzania poprawności danych.

Sprawdzanie poprawności danych przy użyciu regeksów w usłudze Power BI

Do tej pory usługa Power BI nie ma natywnej funkcji w dodatku Power Query do wykonywania operacji z wykorzystaniem regeksów. Zdarzają się jednak przypadki, w których nie można uniknąć użycia regeksów do wyodrębnienia przydatnych informacji z danych w formie tekstowej. Jedynym sposobem, aby móc używać regeksów, są skrypty języka R lub skrypty języka Python. Jedyne ograniczenie w tym przypadku polega na tym, że jeśli chcesz opublikować w usłudze Power BI raport, który wymaga od dodatku Power Query korzystania z zewnętrznych silników języka R lub Python, musisz również zainstalować lokalną bramę danych w trybie osobistym.

Przejdźmy jednak do praktycznych przykładów.

Załóżmy, że pracujesz w firmie detalicznej, w której istnieje zespół zajmujący się identyfikacją nieuczciwych klientów. Gdy członek zespołu zidentyfikuje oszusta, wypełnia arkusz kalkulacyjny Excela, gdzie obok innych kolumn są zawarte kolumny Email i BannedDate. Twoim zadaniem,

w celu przeprowadzenia dokładnej analizy zakupów klienta będącego oszustem, jest załadowanie danych do usługi Power BI z tego pliku Excela oraz z innych źródeł danych i wybranie samych informacji o oszustwie.

Posiadanie poprawnych adresów e-mail oszustów w pliku Excela jest niezbędne do tego, by móc prawidłowo połączyć je z innymi danymi. Odpowiednie daty zablokowania również są ważne, aby wiedzieć, czy po tej dacie prześliznęły się dalsze zamówienia od tego oszusta. Jak wiadomo, wypełnianie pliku Excela przez kilku użytkowników odbywa się bez jakiegokolwiek walidacji wprowadzonych danych, dlatego często zdarzają się błędy. Identyfikacja wszystkich błędów podczas wypełniania niektórych pól i wyróżnianie ich pozwala zespołowi ds. oszustw je poprawić. W tym przypadku z pomocą przychodzą Ci regeksy.

Korzystanie z regeksów w usłudze Power BI do sprawdzania poprawności adresów e-mail z wykorzystaniem języka Python

W repozytorium dołączonym do tej książki, w folderze *Chapter05*, można znaleźć plik Excela *Users.xlsx*. Jego zawartość jest podobna do tej, która została przedstawiona na rysunku 5.21:

	A	B	C	D	E
1	Userid	Email	BannedDate	IsEmailValidByDefinition	IsDateValidByDefinition
2	1	@example.com	05/29/2018	1	1
3	2	example1@example.com/example2@example.com	06/07/2019	0	1
4	3	example33@example.com.	02/05/2018	0	1
5	4	firstname-lastname@example.com	06/07/2019	1	1
6	5	example@example.com --> check	02/29/18	0	0
7	6	email@example-one.com	11/06/2017	1	1

Rysunek 5.21. Zawartość pliku Users.xlsx

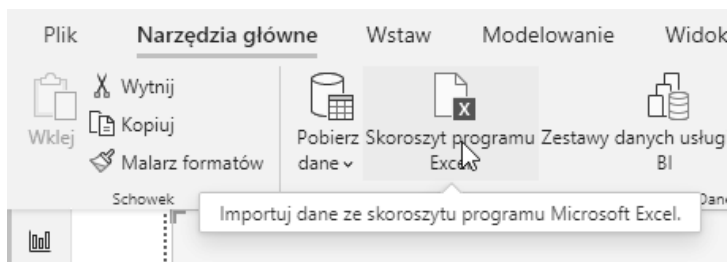
W tym punkcie skupimy się wyłącznie na kolumnie *Email*. Zawiera ona adresy e-mail oszustów, wprowadzone ręcznie przez zespół ds. oszustw wspomniany na początku podrozdziału. Nie wszystkie adresy e-mail zestawione w tej kolumnie mają prawidłową składnię. Ponadto w pliku Excel znajduje się kolumna *IsEmailValidByDefinition*, której wartości (1 = tak; 0 = nie) wskazują, czy adres e-mail odpowiadający tej wartości jest prawidłowy, czy nie.

Python zawiera wbudowany pakiet o nazwie *re*, który dostarcza potrzebnych do pracy z regeksami funkcji. Ponadto pakiet *pandas* dodaje kilka metod dla obiektów *series* lub *dataframe*, które przyjmują regeksy w roli parametrów w celu znalezienia wzorców w ciągu. Metody te działają w taki sam sposób jak te, które udostępnia moduł *re* Pythona. Wkrótce skorzystamy z metody *match*.

Pokażę również składnię `r'...'` do tworzenia ciągów znaków. W ten sposób tworzy się tzw. **surowe ciągi** (ang. *raw strings*), które pozwalają interpretować lewy ukośnik (`\`) dosłownie, a nie jako znak ucieczki.

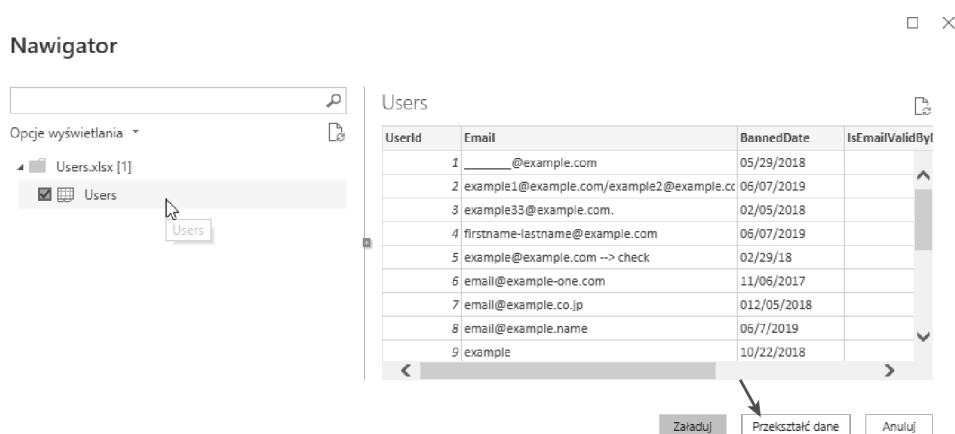
Otwórz program Power BI Desktop, upewnij się, że korzystasz z wirtualnego środowiska języka Python `pbi_powerquery_env`, i zaczynamy:

1. Aby zaimportować dane z programu Excel, kliknij na wstążce ikonę programu Excel (rysunek 5.22):



Rysunek 5.22. Importowanie danych z Excela

2. W oknie dialogowym *Otwieranie* wybierz wcześniej wymieniony plik *Users.xlsx*.
3. W oknie *Nawigator* zaznacz arkusz *Users*, a następnie kliknij *Przekształć dane* (rysunek 5.23):



Rysunek 5.23. Wybierz arkusz Users i kliknij przycisk Przekształć dane

4. Kliknij menu *Przekształć*, a następnie kliknij *Uruchom skrypt języka Python*.
5. Następnie skopiuj i wklej do edytora skryptów Pythona poniższy kod i kliknij OK:

```
import pandas as pd
import re

df = dataset

regex_local_part =
r'([^\<>() [] \\\.,;:\s@\\"]+)(\.[^\<>() [] \\\.,;:\s@\\"]+)*|(\\".+\\")'
regex_domain_name = r'([a-zA-Z\\-0-9]+\\.)+[a-zA-Z]{2,}'
```

```

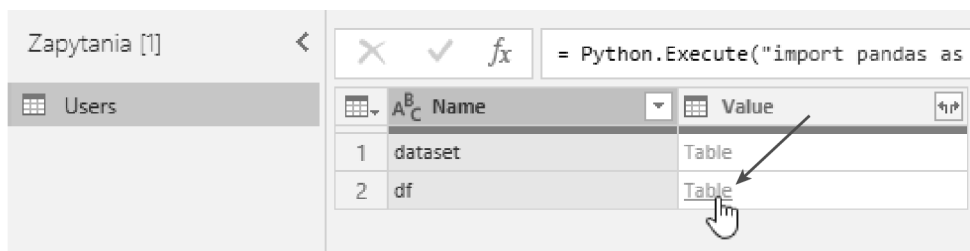
regex_domain_ip_address = r'(\?[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\?|?)'

pattern = r'^({0})@({1}|{2})$'.format(regex_local_part, regex_domain_name,
regex_domain_ip_address)
df['isEmailValidFromRegex'] = df['Email'].str.match(pattern).astype(int)

```

Ten skrypt Pythona możesz również znaleźć w pliku *01-validate-emails-withregex-with-python.py*, który znajduje się w repozytorium dołączonym do książki, w folderze *Chapter05\validating-data-using-regex*.

6. Program Power BI Desktop może wyświetlić alert o następującej treści: *Wymagane są informacje dotyczące prywatności danych*. Jeśli taki komunikat zostanie wyświetlony, kliknij *Kontynuuj* i wykonaj krok 7; w przeciwnym razie możesz przejść do kroku 8.
7. Wyświetli się okno *Poziomy prywatności*. W tym miejscu można określić poziom izolacji, który definiuje stopień, w jakim jedno źródło danych będzie odizolowane od innych źródeł danych. Możesz zignorować sprawdzanie poziomów prywatności, ale może to spowodować ujawnienie poufnych danych osobom nieupoważnionym. Zostaniesz poproszony o wybranie poziomu prywatności zarówno dla skryptu Pythona, jak i zestawu danych załadowanego z programu Excel. Jeśli wybierzesz dla obu poziom *Organizacyjne*, wszystko będzie działać poprawnie w programie Power BI Desktop. Jeśli jednak planujesz opublikować raporty w usłudze Power BI (lub osadzić je w innych miejscach), musisz użyć poziomu *Publiczne*. Aby uzyskać więcej informacji, zapoznaj się ze stroną <http://bit.ly/pbi-privacy-levels>. Na razie wybierz poziom *Organizacyjne* dla obu opcji.
8. Teraz interesuje nas tylko zestaw danych df. Kliknij więc jego wartość *Table* (rysunek 5.24):



Rysunek 5.24. Wybieranie zestawu danych df uzyskanego w wyniku transformacji za pomocą skryptu Pythona

9. Jak widać, została dodana kolumna *isEmailValidFromRegex*, która zawiera wartości logiczne wynikające ze sprawdzania poprawności adresów e-mail za pośrednictwem regaksa. Łatwo zauważyć, że wartości w tej kolumnie pokrywają się z wartościami podanymi w kolumnie *IsEmailValidByDefinition* (rysunek 5.25):

A ^B Email	A ^B BannedDate	1 ² IsEmailValidByDefinition	1 ² IsDateValidByDefinition	1 ² IsEmailValidFromRegex
@example.com	05/29/2018	1	1	1
example1@example.com/example2@exampl...	06/07/2019	0	1	0
example33@example.com.	02/05/2018	0	1	0
firstname-lastname@example.com	06/07/2019	1	1	1
example@example.com --> check	02/29/18	0	0	0
email@example-one.com	11/06/2017	1	1	1
email@example.co.jp	012/05/2018	1	0	1
email@example.name	06/7/2019	1	1	1
example	10/22/2018	0	1	0
email@example.com	9/04/2017	1	1	1
email@subdomain.example.com	11/22/2018	1	1	1
email@123.123.123.123	07/31/17	1	1	1
.@example.com	04/24/018	0	0	0
firstname.lastname@example.com	11/22/18	1	1	1
firstname+lastname@example.com	05/16/2018	1	1	1
example@example.c	6/7/2019	0	1	0
1234567890@example.com	05/16/2018	1	1	1
email@example.museum	03/26/2018	1	1	1
email@[123.123.123.123]	06/31/2017	1	0	1

Rysunek 5.25. Wyniki weryfikacji adresów e-mail za pomocą regeksa

Twój regex wykonał świetną robotę! Teraz możesz wrócić do menu głównego i kliknąć *Zamknij i zastosuj*.

Dzięki kolumnie `isEmailValidFromRegex` możesz teraz odpowiednio filtrować prawidłowe i nie-prawidłowe adresy e-mail. Możesz nawet przekazać raport w tej sprawie do zespołu ds. oszustw.

Korzystanie z regeksów w usłudze Power BI do sprawdzania poprawności adresów e-mail za pomocą języka R

Proces sprawdzania poprawności adresów e-mail za pomocą regeksa z wykorzystaniem języka R jest prawie taki sam, jak z wykorzystaniem Pythona, z wyjątkiem kilku rzeczy.

Po pierwsze, język R, począwszy od wersji 4.0.0, pozwala na korzystanie wyłącznie z surowych ciągów znaków. Ponadto składnia surowych ciągów znaków w R jest nieco inna niż w Pythonie. Zamiast zapisu `r'...'` możesz używać zapisów `r'(...)'`, `r'[...]'` lub `r'{...}'`. Ponadto zamiast używać liczbowych symboli zastępczych w nawiasach klamrowych wewnątrz ciągu, a następnie przypisywać je za pomocą funkcji `format()`, tak jak w języku Python, w R można po prostu bezpośrednio używać nazw zmiennych w nawiasach klamrowych.

Ponadto trzeba zwrócić uwagę na jeszcze jedną rzecz: w języku R nie tylko znak `]` jest interpretowany jako metaznak. W ten sam sposób jest traktowany również znak `[`. Z tego powodu, jeśli chcesz użyć obu nawiasów kwadratowych jako literalów, musisz poprzedzić obydwa znakiem ucieczki w postaci lewego ukośnika (`\`). W związku z tym część regeksa, która identyfikuje klasę znaków w lokalnej części regeksa do sprawdzania adresu e-mail, jest nieco inna (rysunek 5.26):

Python: `[^<>()\\[\]\\. , ; : \s@\"']`

R: `[^<>()\\[\]\\. , ; : \s@\"']`

↑

Rysunek 5.26. W języku R należy poprzedzić otwarty nawias kwadratowy znakiem ucieczki, aby był interpretowany jako literał

Pakiet R.Base udostępnia dwie funkcje umożliwiające korzystanie z regeksów `grep()` i `grepl()`:

- Funkcja `grepl()`, na podstawie tego, czy wzorec istnieje w ciągu znaków, zwraca wartość logiczną.
- Funkcja `grep()` zwraca indeksy wystąpień w wektorze znaków zawierającym dopasowania lub określone ciągi znaków, dla których istnieje dopasowanie.

Ponieważ chcemy zastosować paradygmat Tidyverse, użyjemy funkcji wrapperów dostarczanych przez pakiet `stringr`, odpowiednio `str_detect()` i `str_which()`.

Z wyjątkiem opisanych różnic, proces sprawdzania poprawności adresów e-mail w pliku Excela *Users.xlsx* w usłudze Power BI przy użyciu skryptu języka R jest praktycznie taki sam, jak opisany w poprzednim punkcie proces dotyczący Pythona:

1. Aby zaimportować dane z pliku *Users.xlsx*, powtórz kroki od 1. do 3. z poprzedniego punktu.
2. Kliknij menu *Przekształć*, a następnie kliknij polecenie *Uruchom skrypt języka R*.
3. Następnie skopiuj i wklej poniższy kod do edytora skryptów R i kliknij OK:

```
library(dplyr)
library(stringr)

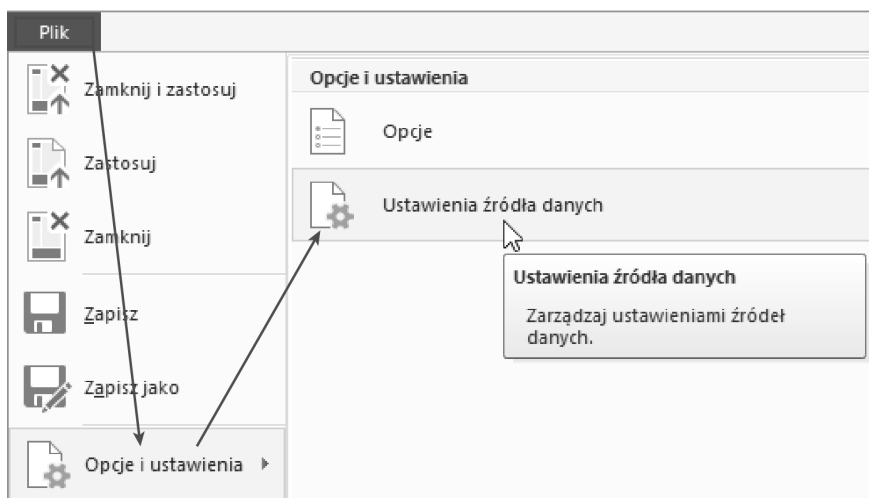
regex_local_part <-
r'(([^<>()\\[\]\\. , ; : \s@\"']+(\.[^<>()\\[\]\\. , ; : \s@\"']+)*|(\".+\"))'
regex_domain_name <- r'((([a-zA-Z\\-0-9]+\\.)+[a-zA-Z]{2,}))'
regex_domain_ip_address <- r'((\\[?0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.?)?)'

pattern <- str_glue(
  '^{regex_local_part}@({regex_domain_name}|{regex_domain_ip_address})$'
)

df <- df %>%
  mutate( isEmailValidFromRegex = as.integer(str_detect(Email, pattern)) )
```

Ten skrypt języka R możesz również znaleźć w pliku *02-validate-emails-with-regexwith-r.R*, w folderze *Chapter05\\validating-data-using-regex*, w repozytorium dołączonym do książki.

4. Program Power BI Desktop może wyświetlić powiadomienie o następującej treści: *Wymagane są informacje dotyczące prywatności danych*. Jeśli tak się stanie, kliknij *Kontynuuj* i postępuj zgodnie z instrukcjami zamieszczonymi poniżej. W przeciwnym razie możesz przejść do kroku 5. Dla skryptów języka R również wybierz poziom *Organizacyjny*. Czasami okazuje się, że poziomy zgodności zestawów danych i skryptów analitycznych nie są ze sobą zgodne. W takim przypadku usługa Power BI może wyświetlić ostrzeżenie w postaci *Formuła.Firewall: Zapytanie "XXX" (krok "YYY") próbuje uzyskać dostęp do źródeł danych z poziomami prywatności, które nie mogą być używane razem. Utwórz ponownie tę kombinację danych*. W takim przypadku wystarczy otworzyć okno *Ustawienia źródła danych* w następujący sposób (rysunek 5.27):

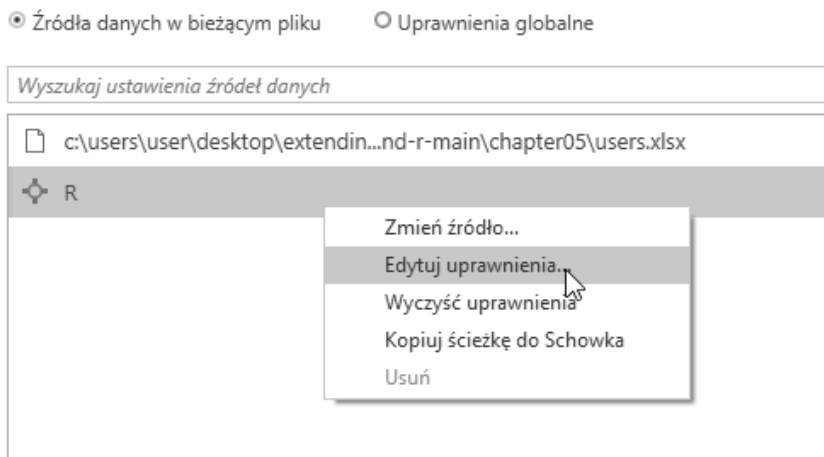


Rysunek 5.27. Otwieranie okna ustawień źródła danych dodatku Power Query

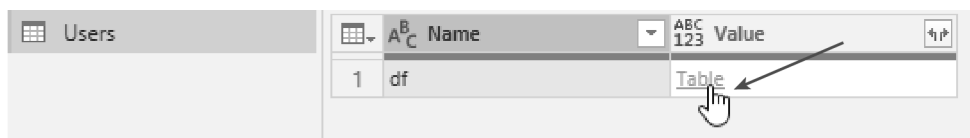
Następnie musisz upewnić się, że wszystkie źródła danych mają ten sam poziom prywatności (w naszym przypadku *Organizacyjny*). W tym celu zmień go dla każdego źródła danych, dla którego jest to konieczne, za pomocą opcji *Edit Permissions...* (rysunek 5.28).

W tym momencie możesz odświeżyć dane podglądu. Aby to zrobić, kliknij *Odśwież podgląd*.

5. W tym przypadku interesuje nas tylko zestaw danych df. Kliknij więc odpowiadającą mu wartość *Table* (rysunek 5.29).
6. Jak widać, kolumna *isEmailValidFromRegex* została dodana i zawiera wartości logiczne (przekształcone na 1 i 0) będące wynikiem walidacji adresów e-mail za pomocą regaksa. Łatwo zauważyć, że wartości w tej kolumnie pokrywają się z wartościami podanymi w kolumnie *IsEmailValidByDefinition*. Twój regex wykonał świetną robotę! Teraz możesz wrócić do menu *Narzędzia główne* i kliknąć polecenie *Zamknij i zastosuj*.



Rysunek 5.28. Edytowanie uprawnień prywatności dla źródeł danych



Rysunek 5.29. Wybieranie zestawu danych df uzyskanego w wyniku transformacji za pomocą skryptu języka R

Dzięki kolumnie `isEmailValidFromRegex` możesz teraz odpowiednio filtrować poprawne i niepoprawne adresy e-mail w swoich raportach.

Teraz przyjrzyjmy się, jak sprawdzać poprawność dat w usłudze Power BI za pomocą języka Python.

Korzystanie z regeksów w usłudze Power BI do sprawdzania poprawności dat za pomocą języka Python

Jako przykład dat do sprawdzenia poprawności użyjemy wykorzystanego już wcześniej pliku Excela *Users.xlsx*. Plik ten zawiera kolumnę `BannedDate` z tekstowymi reprezentacjami dat w formacie `mm/dd/yyyy` we wszystkich jego wariantach. Ponadto w pliku Excela znajduje się również kolumna `IsDateValidByDefinition`, której wartości (1 = tak; 0 = nie) wskazują, czy data pasująca do tej wartości jest prawidłowa, czy nie.

Funkcje Pythona potrzebne do skorzystania z regeksów poznałeś już wcześniej. Zatem zaczynamy:

1. Aby zaimportować dane z pliku *Users.xlsx*, powtórz kroki od 1. do 3. z punktu „Korzystanie z regeksów w usłudze Power BI do sprawdzania poprawności adresów e-mail za pomocą języka Python”.
2. Kliknij menu *Przekształć*, a następnie *Uruchom skrypt języka Python*.
3. Następnie skopiuj poniższy kod i wklej go do edytora skryptów Pythona, po czym kliknij *OK*:

```
import pandas as pd
import re

df = dataset

regex_dates_having_day_31 = r'(?:(?:0?[13578])|(?:1[02]))[\-\/]31[\-\/](?:19|20)?\d\d)'

regex_non_leap_dates_having_days_29_30 = r'(?:(?:0?[13-9])|(?:1[0-2]))[\-\/](?:29|30)[\-\/](?:19|20)?\d\d)'

regex_leap_dates_having_day_29 = r'(?0?2[\-\/]29[\-\/](?:19|20)?(?:0?[0468][048])|(?:[13579][26])))'

regex_remaining_dates = r'(?:(?:0?[1-9])|(?:1[0-2]))[\-\/](?:1\d)|(?:0?[1-9])|(?:2[0-8]))[\-\/](?:19|20)?\d\d)'

pattern = r'^(?:{0}|{1}|{2}|{3})$'.format(regex_dates_having_day_31,
regex_non_leap_dates_having_days_29_30, regex_leap_dates_having_day_29,
regex_remaining_dates)

df['isValidDateFromRegex'] = df['BannedDate'].str.
match(pattern).astype(int)
```

Bardziej kompletny skrypt Pythona możesz znaleźć w folderze *Chapter05\validating-data-using-regex*, w pliku *03-validate-dateswith-regex-with-python.py*, w repozytorium dołączonym do książki. Skrypt ten obsługuje daty w formatach mm-dd-rrrr, dd-mm-rrrr i rrrr-mm-dd, wraz ze wszystkimi ich odmianami, w tym zarówno z separatorem -, jak i /.

4. Jeśli usługa Power BI wymaga podania informacji o prywatności, postępuj zgodnie ze wskazówkami zamieszczonymi w poprzednich punktach.
5. Jak widać, kolumna *isValidDateFromRegex* została dodana i zawiera wartości logiczne wynikające z walidacji dat za pomocą regeksa. Łatwo zauważyć, że wartości te pokrywają się z wartościami podanymi w kolumnie *IsDateValidByDefinition* (rysunek 5.30).

Twój regex wykonał świetną robotę! Teraz możesz wrócić do menu *Narzędzia główne* i kliknąć polecenie *Zamknij i zastosuj*.

Dzięki kolumnie *isValidDateFromRegex* możesz teraz filtrować poprawne i niepoprawne daty i odpowiednio je wykorzystywać.

A ^B _C BannedDate	1 ² ₃ IsEmailValidByDefinition	1 ² ₃ IsDateValidByDefinition	1 ² ₃ isValidDateFromRegex
05/29/2018	1	1	1
06/07/2019	0	1	1
02/05/2018	0	1	1
06/07/2019	1	1	1
02/29/18	0	0	0
11/06/2017	1	1	1
012/05/2018	1	0	0
06/7/2019	1	1	1
10/22/2018	0	1	1
9/04/2017	1	1	1
11/22/2018	1	1	1
07/31/17	1	1	1
04/24/018	0	0	0
11/22/18	1	1	1
05/16/2018	1	1	1
6/7/2019	0	1	1
05/16/2018	1	1	1
03/26/2018	1	1	1
06/31/2017	1	0	0

Rysunek 5.30. Wyniki regeksa walidacji dat

Korzystanie z regeksów w usłudze Power BI do sprawdzania poprawności dat za pomocą języka R

Proces sprawdzania poprawności daty przy użyciu regeksów i języka R jest prawie taki sam, jak za pomocą Pythona. Różnice są podobne do tych, które wymieniłem w punkcie „Korzystanie z regeksów w usłudze Power BI do sprawdzania poprawności adresów e-mail za pomocą języka R”. W przykładzie skorzystamy z tego samego pliku Excela *Users.xlsx*. Oto procedura, którą należy wykonać:

1. Aby zaimportować dane z pliku *Users.xlsx*, powtórz kroki od 1. do 3. z punktu „Korzystanie z regeksów w usłudze Power BI do sprawdzania poprawności adresów e-mail za pomocą języka Python”.
2. Kliknij menu *Przekształć*, a następnie polecenie *Uruchom skrypt języka R*.
3. Skopiuj poniższy kod i wklej go do edytora skryptów języka R, po czym kliknij *OK*:

```
library(dplyr)
library(stringr)

df <- dataset

regex_dates_having_day_31 <- r'((?:(?:0?[13578])|(?:1[02]))[\-\/]31[\-\/]
(?:19|20)?\d\d)'
regex_non_leap_dates_having_days_29_30 <- r'((?:(?:0?[13-9])|(?:1[0-2]))[\-\/]
(?:29|30)[\-\/](?:19|20)?\d\d)'
```

```

regex_leap_dates_having_day_29 <- r'((?:0?2[\\-\\/])29[\\-\\/](?:19|20)?(?:?:[02468][048])|(?:[13579][26])))'
regex_remaining_dates <- r'((?:?:?:0?2[1-9])|(?:1[0-2]))[\\-\\/](?:?:1\\d)|(?:0?2[1-9])|(?:2[0-8]))[\\-\\/](?:19|20)?\\d\\d)'

pattern <-
str_glue('^(?:{regex_dates_having_day_31}|{regex_non_leap_dates_having_days_29_30}|{regex_leap_dates_having_day_29}|{regex_remaining_dates})$')

df <- df %>%
  mutate( isDateValidFromRegex = as.integer(str_
detect(BannedDate, pattern)) )

```

Bardziej kompletny skrypt języka R można znaleźć w folderze *Chapter05\\validating-data-using-regex*, w pliku *04-validate-dates-withregex-with-r.R*, w repozytorium dołączonym do tej książki. Ten skrypt obsługuje daty w formatach mm-dd-yyyy, dd-mm-yyyy i yyyy-mm-dd ze wszystkimi ich odmianami, w tym zarówno z separatorem -, jak i /.

4. Jeśli usługa Power BI wymaga podania informacji o prywatności, postępuj zgodnie ze wskazówkami zamieszczonymi w poprzednich punktach.
5. Jak widać, kolumna `isValidDateFromRegex` została dodana i zawiera wartości logiczne wynikające z walidacji dat za pomocą regeksa. Łatwo zauważyć, że wartości te pokrywają się z wartościami podanymi w kolumnie `IsDateValidByDefinition`. Twój regex wykonał świetną robotę! Teraz możesz wrócić do menu *Narzędzia główne* i kliknąć polecenie *Zamknij i zastosuj*.

Dzięki kolumnie `isEmailValidFromRegex` możesz teraz odpowiednio filtrować poprawne i niepoprawne adresy e-mail w swoich raportach.

W następnym podrozdziale dowiesz się, jak zaimportować zawartość półstrukturalnego pliku loga za pomocą języków Python i R.

Ładowanie do usługi Power BI złożonych plików logów z wykorzystaniem regeksów

Pliki logów to bardzo przydatne narzędzia dla programistów i administratorów systemów komputerowych. Rejestrują, co działo się z systemem, kiedy to się stało i który użytkownik wygenerował zdarzenie. Dzięki tym plikom można znaleźć informacje o każdej awarii systemu, co pozwala na szybszą diagnozę przyczyn występujących usterek.

Logi często zawierają **dane częściowo ustrukturyzowane**, czyli informacje, których nie można utrwalić w relacyjnej bazie danych w formacie, w którym zostały wygenerowane. Aby można było przeanalizować dane z logów za pomocą zwykłych narzędzi, dane te powinny przede wszystkim być przekształcone na bardziej odpowiedni format.

Ponieważ logi nie są danymi ustrukturyzowanymi, trudno jest je zaimportować do usługi Power BI w niezmienionej postaci bez opracowania specjalnego, niestandardowego łącznika. To właśnie w tych scenariuszach pożądane rezultaty mogą nam pomóc uzyskać regeksy użyte za pośrednictwem takich języków, jak Python lub R.

Logi dostępu do serwera Apache

Załóżmy, że Twoja firma ma stronę internetową opublikowaną za pośrednictwem serwera WWW Apache. Twój menedżer prosi Cię o przeprowadzenie analizy dotyczącej tego, na których stronach internetowych witryny użytkownicy najczęściej klikają. Jedynym sposobem uzyskania tych informacji jest przeanalizowanie pliku logu dostępu. W tym pliku są zarejestrowane dane o wszystkich żądaniach wysyłanych do serwera WWW. Oto przykład zawartości logu dostępu serwera Apache (rysunek 5.31):

```
1 83.149.9.216 - - [17/May/2015:10:05:03 +0000] "GET
/presentations/logstash-monitorama-2013/images/kibana-search.png HTTP/1.1" 200 203023
"http://semicomplete.com/presentations/logstash-monitorama-2013/" "Mozilla/5.0 (Macintosh;
Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36"
2 83.149.9.216 - - [17/May/2015:10:05:43 +0000] "GET
/presentations/logstash-monitorama-2013/images/kibana-dashboard3.png HTTP/1.1" 200 171717
"http://semicomplete.com/presentations/logstash-monitorama-2013/" "Mozilla/5.0 (Macintosh;
Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36"
```

Rysunek 5.31. Przykład logu dostępu serwera Apache

Jak widać, struktura informacji w tym logu jest dość dobrze uporządkowana. Jeśli nikt nie spersonalizował danych wyjściowych plików logów serwera Apache, domyślnie stosowany jest format CLF (ang. *Common Log Format*). Rzeczywisty przykład logu dostępu serwera Apache możesz znaleźć w folderze *Chapter05/loading-complex-log-files-using-regex*, w pliku *apache_logs.txt*, w repozytorium dołączonym do tej książki. Plik ten znajdziesz w repozytorium GitHub, pod adresem <http://bit.ly/apache-access-log> (aby go wyświetlić, kliknij *Download*).

Jeśli przeczytasz dokumentację tych plików logów, odkryjesz, że informacje zapisane w logu dostępu są zgodne z formatem NCSA. Tak więc logowane są następujące dane:

1. Nazwa zdalnego hosta (adres IP).
2. Nazwa zdalnego logu (jeśli jest pusta, w logu będzie myślnik, co oznacza, że nazwa ta nie jest używana w przykładowym pliku).
3. Zdalny użytkownik, jeśli żądanie zostało uwierzytelnione (jeśli nie ma danych użytkownika, w logu będzie myślnik).
4. Data otrzymania żądania w formacie [18/Sep/2011:19:18:28 -0400].
5. Pierwszy wiersz żądania wysłanego do serwera umieszczony w cudzysłowie.
6. Kod statusu HTTP żądania.
7. Rozmiar odpowiedzi w bajtach, z wyłączeniem nagłówków HTTP (wartość ta może być oznaczona myślnikiem, jeśli jej brakuje).
8. Nagłówek żądania HTTP Referer zawierający bezwzględny lub częściowy adres strony wysyłającej żądanie.

9. Nagłówek żądania HTTP User-Agent zawierający ciąg identyfikujący aplikację, system operacyjny, dostawcę i (lub) wersję żądającego agenta użytkownika.

Gdy znasz zarówno charakter informacji zapisanych w logu, jak i format, w jakim informacje zostały zapisane, możesz skorzystać z zaawansowanych narzędzi udostępnianych przez regeksy. Pozwala to lepiej ustrukturyzować te informacje i zaimportować do usługi Power BI.

Importowanie logów dostępu serwera Apache w usłudze Power BI za pomocą języka Python

Jak wspomniałem wcześniej, rzeczywisty przykład logu dostępu serwera Apache jest dostępny w pliku *apache_logs.txt*, w folderze *Chapter05/loading-complex-log-files-using-regex*, w repozytorium dołączonym do tej książki. Informacje zapisane w tym pliku załadujemy za pomocą skryptu języka Python, a nie łącznika usługi Power BI.

W porównaniu z tym, czego nauczyłeś się wcześniej o regeksach i Pythonie, w skrypcie Pythona *01-apacheaccess-log-parser-python.py* (który znajdziesz w wyżej wymienionym folderze), występują następujące nowe konstrukcje:

Aby czytać plik tekstowy wiersz po wierszu w Pythonie, należy użyć funkcji `open(file, mode)` i metody `readlines()`. W szczególności chcemy odczytać plik *apache_logs.txt* w trybie tylko do odczytu ('r') i zapisać na liście każdy z odczytanych wierszy.

W regeksach możliwe jest odwoływanie się do grup identyfikowanych za pomocą okrągłych nawiasów nie tylko przez indeks liczbowy, ale także *przez nazwę*. Jest to możliwe dzięki tzw. **nazwanym grupom przechwytyjącym**. Zazwyczaj składnia regeksa używana do przypisywania nazwy do grupy to `(?<nazwa-grupy>...)`. W Pythonie należy wykorzystać składnię `(?P<nazwa-grupy>...)`:

- W Pythonie można zdefiniować listę części regeksa, które można scalać (połączyć) za pomocą separatora zdefiniowanego przez regex `(\s+)`:

```
regex_parts = [
    r'(?P<hostName>\S+)',
    r'\S+',
    r'(?P<userName>\S+)',
    r'\[({P<requestDateTime>[\w:/]+\s[+-]\d{4}})\]',
    r'"(?P<requestContent>\S+\s?\S+?\s?\S+?)"',
    r'(?P<requestStatus>\d{3}| -)',
    r'(?P<responseSizeBytes>\d+|-)',
    r'"(?P<requestReferrer>[^"]*)"',
    r'"(?P<requestAgent>[^"]*)"'
]

pattern = re.compile(r'\s+'.join(regex_parts) + r'$')
```

Należy zauważyć, że w tym przypadku użyliśmy funkcji `re.compile()`, ponieważ dopasowanie musi być wykonane wiele razy we wszystkich wierszach logu. W związku z tym można skorzystać ze wstępnej kompilacji regeksa i osiągnąć zyski obliczeniowe.

- Operacja dopasowywania wzorców jest wykonywana dla każdego wiersza w logu:

```
for line in access_log_lines:
    log_data.append(pattern.match(line).groupdict())
```

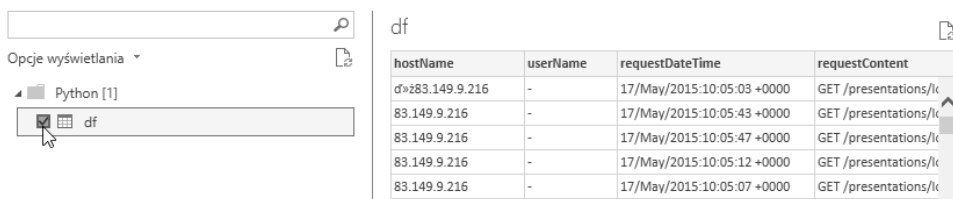
Metoda `groupdict()` zwraca słownik z kluczami w postaci nazw grup i wartościami oznaczającymi dopasowane ciągi. Wszystkie słowniki dla każdego wiersza są dołączane do listy `log_data`.

Interpretację sposobu, w jaki każda część regaksa przechwytuje pożądany ciąg, pozostawiam Czytelnikowi.

Po wyjaśnieniu kilku punktów w kodzie, spróbujemy zaimportować log do usługi Power BI:

1. W programie Power BI Desktop upewnij się, że używasz środowiska `pbi_powerquery_env`.
2. Przejdź do menu *Pobierz dane* i wybierz *Skrypt w języku Python*.
3. Skopiuj zawartość skryptu `01-apache-access-log-parser-python.py` i wklej ją do edytora skryptów Pythona, po czym kliknij *OK*.
4. Następnie w oknie *Navigator* wybierz ramkę danych `df` i kliknij *Załaduj* (rysunek 5.32):

Nawigator



Rysunek 5.32. Wybieranie ramki danych `df` zwróconej przez skrypt Pythona

5. Jeśli klikniesz ikonę *Dane*, możesz wyświetlić cały załadowany log w postaci ustrukturyzowanej tabeli (rysunek 5.33):

hostName	userName	requestDateTime	requestContent	requestStatus	responseSizeBytes	requestReferrer	requestAgent
36.38.8.174	-	17/May/2015:12:05:39 +0000	GET /favicon.ico HTTP/1.1	200	3638	-	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537
92.108.120.46	-	17/May/2015:13:05:12 +0000	GET /favicon.ico HTTP/1.1	200	3638	-	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537
75.97.9.59	-	17/May/2015:19:05:12 +0000	GET /favicon.ico HTTP/1.1	200	3638	-	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537
217.140.110.23	-	17/May/2015:19:05:40 +0000	GET /favicon.ico HTTP/1.1	200	3638	-	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537
201.76.90.77	-	17/May/2015:20:05:59 +0000	GET /favicon.ico HTTP/1.1	200	3638	-	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537

Rysunek 5.33. Log dostępu serwera Apache załadowany w usłudze Power BI za pomocą języka Python

Super! Dzięki możliwościom oferowanym przez regeksy udało Ci się zaimportować do usługi Power BI dane ze skomplikowanego pliku logu.

Importowanie logów dostępu serwera Apache w usłudze Power BI za pomocą języka R

W tym punkcie załadujemy informacje z pliku *apache_logs.txt*, ale tym razem za pomocą skryptu w języku R.

W porównaniu z tym, czego nauczyłeś się wcześniej o regeksach w języku R, w skrypcie *02-apache-access-log-parser-r* (który znajdziesz w wymienionym wcześniej folderze) napotkasz następujące nowe konstrukcje:

- Aby móc odczytywać plik tekstowy wiersz po wierszu w języku R, należy użyć funkcji `read_lines()` z pakietu `readr`. W szczególności będziemy odczytywać poszczególne wiersze pliku *apache_logs.txt* w celu utrwalenia ich w wektorze.
- Aby w pełni wykorzystać nazwane grupy przechwytyjące w języku R, należy zainstalować nowy pakiet o nazwie `namedCapture` i z niego skorzystać. Dzięki temu pakietowi dozwolone są obydwie składnie regeksów dla nazwanych grup: standardowa składnia regeksów `(?<nazwa-grupy>...)` oraz składnia `(?P<nazwa-grupy>...)`.
- Podobnie jak zrobiliśmy to w skrypcie Pythona, w skrypcie języka R zdefiniujemy wektor części regeksa, które scalimy ze sobą za pomocą funkcji `paste(..., collapse = '...')`. Ta funkcja ma za zadanie łączenie części regeksa za pomocą separatora `\s+`. Po scaleniu wszystkich części na końcu dodawany jest znak `$` za pomocą funkcji `paste0(...)`. Pamiętaj, że surowe ciągi znaków w języku R mają inną składnię niż w Pythonie. W tym przypadku użyjemy składni `r'{...}'`:

```
regex_parts <- c(
  r' {(?P<hostName>\S+)} '
  , r' {\S+} '
  , r' {(?P<userName>\S+)} '
  , r' {\[(?P<requestDateTime>[\w:./]+\s[+-]\d{4})\]} '
  , r' {"(?P<requestContent>\S+\s?\S+?\s?\S+?)"} '
  , r' {(?P<requestStatus>\d{3}| -)} '
  , r' {(?P<responseSizeBytes>\d+|-)} '
  , r' {"(?P<requestReferrer>[\"^\"*)"} '
  , r' {"(?P<requestAgent>[\"^\"*)"} '
)

pattern <- paste0( paste(regex_parts, collapse = r' {\s+} ' ), '$' )
```

- Dopasowywanie wzorców odbywa się za pomocą funkcji `str_match_named()` należącej do pakietu `namedCapture`. Funkcja ta działa na całym wektorze logu dzięki poleceniu złożonemu z jednego wiersza:

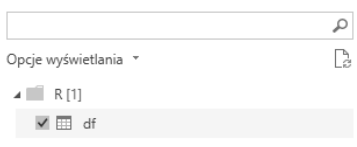
```
df <- as.data.frame( str_match_named( access_log_lines, pattern = pattern ) )
```

Interpretację sposobu, w jaki poszczególne części regeksa przechwytyują pożądany ciąg, pozostawiam Czytelnikowi.

Teraz, gdy wyjaśniłem kilka punktów w kodzie, spróbujemy zaimportować log do usługi Power BI:

1. Najpierw musisz zainstalować pakiet namedCapture. Otwórz RStudio i upewnij się, że w ustawieniach *Global Options* ustawiłeś najnowszą wersję silnika.
2. Teraz przejdź do konsoli, wprowadź poniższy kod i uruchom go:
3. `install.packages("namedCapture")`
4. Otwórz program Power BI Desktop, przejdź do pozycji *Pobierz dane* i wybierz *Skrypt języka R*.
5. Skopiuj skrypt z pliku *02-apache-access-log-parser-r.R* i wklej go do edytora skryptów języka R, a następnie kliknij przycisk *OK*.
6. Następnie w oknie *Nawigator* wybierz ramkę danych *df* i kliknij *Zaladuj* (rysunek 5.34):

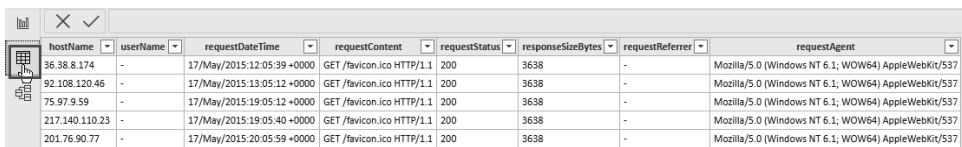
Nawigator



hostName	userName	requestDateTime	requestContent
83.149.9.216	-	17/May/2015:10:05:03 +0000	GET /presentations/k
83.149.9.216	-	17/May/2015:10:05:43 +0000	GET /presentations/k
83.149.9.216	-	17/May/2015:10:05:47 +0000	GET /presentations/k

Rysunek 5.34. Wybieranie ramki danych *df* zwróconej przez skrypt języka R

7. Jeśli klikniesz ikonę *Dane*, możesz wyświetlić cały załadowany log w postaci ustrukturyzowanej tabeli (rysunek 5.35):



hostName	username	requestDateTime	requestContent	requestStatus	responseSizeBytes	requestReferrer	requestAgent
36.38.8.174	-	17/May/2015:12:05:39 +0000	GET /favicon.ico HTTP/1.1	200	3638	-	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537
92.108.120.46	-	17/May/2015:13:05:12 +0000	GET /favicon.ico HTTP/1.1	200	3638	-	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537
75.97.9.59	-	17/May/2015:19:05:12 +0000	GET /favicon.ico HTTP/1.1	200	3638	-	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537
217.140.110.23	-	17/May/2015:19:05:40 +0000	GET /favicon.ico HTTP/1.1	200	3638	-	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537
201.76.90.77	-	17/May/2015:20:05:59 +0000	GET /favicon.ico HTTP/1.1	200	3638	-	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537

Rysunek 5.35. Log dostępu serwera Apache załadowany w usłudze Power BI za pomocą języka R

Świetna robota! Udało Ci się zaimportować częściowo ustrukturyzowany plik logu do usługi Power BI za pomocą języka R.

Wyodrębnianie wartości z tekstu w usłudze Power BI przy użyciu regeksów

Ostatni przypadek, który chcę zaprezentować, zdarza się bardzo często, gdy mamy do czynienia z wysyłką towarów do klientów. Bywa, że oszustowi udaje się ukraść towar zaadresowany do klienta; dlatego firma musi zwrócić klientowi środki. Oszukany klient kontaktuje się z działem obsługi klienta, aby poprosić o zwrot pieniędzy. Jeśli system zarządzania dostarczony operatorowi obsługi klienta zarządzającego sprawą nie pozwala na wprowadzanie informacji o zwrocie

w sposób ustrukturyzowany, operator musi skorzystać z jedynej możliwej metody: wprowadzenia *dowolnego tekstu* związanego z zamówieniem, z określeniem *kwoty* zwrotu, *powodu* zwrotu i *daty*.

Z pewnością już wiesz, że informacje wprowadzone w formacie dowolnego tekstu są koszmarem każdego analityka, zwłaszcza gdy Twój szef poprosi Cię o przeanalizowanie informacji wprowadzanych w tych niesławnych notatkach.

W repozytorium dołączonym do tej książki, w folderze *Chapter05*, można znaleźć plik Excela *OrderNotes.xlsx*. Jego zawartość jest podobna do treści pokazanej na rysunku 5.36:

	A	B
1	OrderNumber	Notes
2	ORD000001	5,00 EUR Kradzież przy dostawie opłaconej przelewem 11/02/2021
3	ORD000002	29,00 EUR Zwrot za kradzież w dostawie 04/06/2020
4	ORD000003	53,00€ Zwrot za kradzież w dostawie 24/09/2020
5	ORD000004	29/10/2020 EUR45,00 Zwrot za kradzież w dostawie
6	ORD000005	EUR 522,00 Zwrot za kradzież w dostawie 20/08/2020
7	ORD000006	€ 266,00 - Kradzież w dostawie opłaconej przelewem w dniu 10/12/2020
8	ORD000007	EUR68,50 - Zwrot za kradzież w dostawie 02/07/2020
9	ORD000008	EUR 50,00 - Zwrot za kradzież przy dostawie z dnia 30/07/2020
10	ORD000009	30/07/2020 209,00 € - Zwrot za kradzież w dostawie

Rysunek 5.36. Notatki tekstowe wprowadzone przez operatora dla niektórych zamówień

Jak widać po zawartości pliku Excela, z notatek należy wyodrębnić następujące informacje:

- Kwota zwrotu.
- Powód zwrotu.
- Data zwrotu pieniędzy.

Problem polega na tym, że operatorzy obsługi klienta wprowadzili te informacje bez zastanowienia się nad zasadą ich ustrukturyzowania. Z tego powodu w arkuszu możemy zaobserwować następujące elementy:

- Kwota zwrotu została wprowadzona jako *EUR xx,yy*, *EURxx,yy*, *xx,yy EUR*, *€ xx,yy*, *xx,yy€* i *xx,yy €*.
- „Separator” użyty między poszczególnymi fragmentami informacji może być utworzony za pomocą jednej lub kilku białych spacji, albo przez myślnik otoczony jedną białą spacją lub więcej białymi spacjami.
- Data zwrotu jest zawsze w formacie *dd/mm/yyyy* (tu mamy szczęście!).
- Powód zwrotu może zawierać dowolny tekst.

Czy możliwe jest prawidłowe wyodrębnienie informacji potrzebnych do wykonania analizy wymaganej przez szefa, biorąc pod uwagę dowolność wprowadzonych notatek? Odpowiedź brzmi „z pewnością tak”, pod warunkiem, że wiesz, jak używać reguł.

Jeden regex do zarządzania wszystkimi danymi

Dzięki doświadczeniu, które zebrałeś podczas lektury wcześniejszej części książki, natychmiast wpadniesz na rozwiązanie, które zamierzam Ci zaproponować. Rozważ następujące części regeksa:

- **Waluta:** `(?:EUR|€)`.
- **Separator:** `(?:\s+)?-(?:\s+)?`.
- **Kwota zwrotu:** `\d{1,}\,\?\d{0,2}`.
- **Powód zwrotu:** `.*?`.
- **Data zwrotu:** `\d{2}[\-\/]\d{2}[\-\/]\d{4}`.

Pamiętasz składnię **grupy nieprzechwytywającej** `(?:...)`? Dzięki tej składni wyraźnie mówisz silnikowi regeksów, że nie chcesz przechwytywać zawartości w tych nawiasach, ponieważ nie są to ważne informacje do wyodrębnienia. Pamiętaj, że końcowy regex to nic innego jak wiele alternatywnych kombinacji poszczególnych części. Może mieć on postać pokazaną na rysunku 5.37:

```
^(?:
  ({waluta}{separator}{kwota}{separator}{powód}{separator}{data})
  OR
  ({kwota}{separator}{waluta}{separator}{powód}{separator}{data})
  OR
  ({data}{separator}{waluta}{separator}{kwota}{separator}{powód})
  OR
  ({data}{separator}{kwota}{separator}{waluta}{separator}{powód})
)$
```

Rysunek 5.37. Pełna struktura regeksa do wyodrębniania informacji z notatek

Oto kompletny regex:

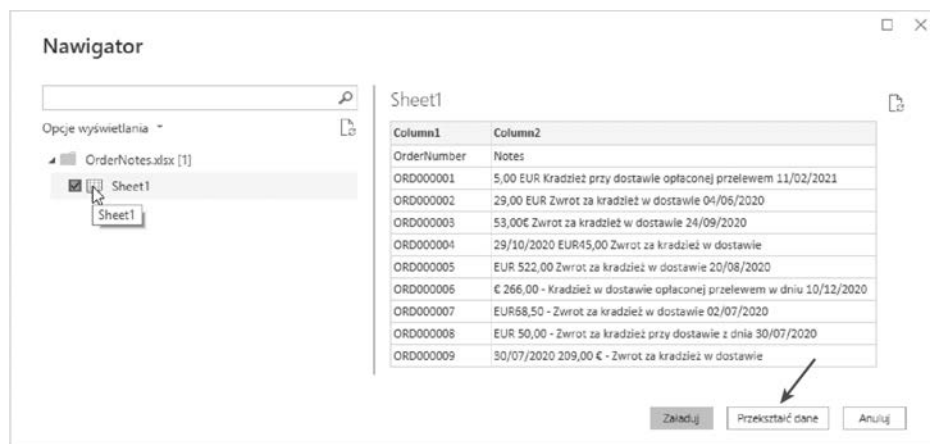
```
^(?: (?: (?: EUR|€ ) (?: \s+ )? - (?: \s+ )? (?: P<RefundAmount> \d{1,} \, \. ? \d{0,2} ) (?: \s+ )? -
  (?: \s+ )? (?: P<RefundReason> . * ? ) (?: \s+ )? - (?: \s+ )? (?: P<RefundDate> \d{2} [ \- \/ ] \d{2} [ \-
  \/ ] \d{4} ) (?: \s+ )? - (?: \s+ )? ) | (?: (?: P<RefundAmount> \d{1,} \, \. ? \d{0,2} ) (?: \s+ )? -
  (?: \s+ )? (?: EUR|€ ) (?: \s+ )? - (?: \s+ )? (?: P<RefundReason> . * ? ) (?: \s+ )? -
  (?: \s+ )? (?: P<RefundDate> \d{2} [ \- \/ ] \d{2} [ \- \/ ] \d{4} ) (?: \s+ )? -
  (?: \s+ )? ) | (?: (?: P<RefundDate> \d{2} [ \- \/ ] \d{2} [ \- \/ ] \d{4} ) (?: \s+ )? -
  (?: \s+ )? (?: EUR|€ ) (?: \s+ )? - (?: \s+ )? (?: P<RefundAmount> \d{1,} \, \. ? \d{0,2} ) (?: \s+ )? -
  (?: \s+ )? (?: P<RefundReason> . * ? ) (?: \s+ )? - (?: \s+ )? ) | (?: (?: P<RefundDate> \d{2} [ \-
  \/ ] \d{2} [ \- \/ ] \d{4} ) (?: \s+ )? - (?: \s+ )? (?: P<RefundAmount> \d{1,} \, \. ? \d{0,2} ) (?: \s+ )? -
  (?: \s+ )? (?: EUR|€ ) (?: \s+ )? - (?: \s+ )? (?: P<RefundReason> . * ? ) (?: \s+ )? - (?: \s+ )? ) ) $
```

Spróbujmy go zaimplementować w usłudze Power BI z wykorzystaniem Pythona.

Używanie regeksów w usłudze Power BI do wyodrębniania wartości z wykorzystaniem języka Python

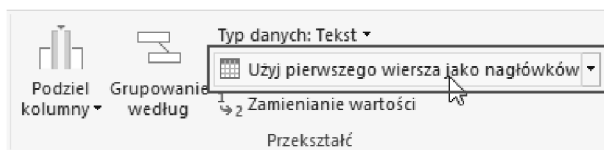
Jak widać na rysunku 5.37, regex zawiera nazwane grupy, które są w nim *używane wielokrotnie*. Niestety ponowne użycie tej samej nazwanej grupy w regeksach nie jest obsługiwane przez moduł Python re (ten sam moduł jest również używany za kulisami w pakiecie pandas). Aby skorzystać z bardziej zaawansowanych funkcji regeksów, takich jak wcześniej wymienione grupy o identycznych nazwach lub składnie *lookbehind* i *lookahead* (których dotąd nie omówiłem w tym rozdziale), musisz użyć modułu regex. Przede wszystkim trzeba go zainstalować w środowisku wirtualnym *pbi_powerquery_env*. Potem trzeba załadować do programu Power BI Desktop plik Excela *OrderNotes.xlsx*, dostępny w folderze *Chapter05*. Następnie można przekształcić ten zestaw danych za pomocą skryptu Pythona. Zaczynamy:

1. Otwórz program Anaconda Prompt, za pomocą polecenia `conda activate pbi_powerquery_env` przełącz się do środowiska wirtualnego *pbi_powerquery_env*, a następnie zainstaluj pakiet `regex` za pomocą kodu: `pip install regex`.
2. Otwórz program Power BI Desktop i sprawdź w opcjach programu, czy zostało ustawione środowisko wirtualne Pythona *pbi_powerquery_env*.
3. Aby zaimportować dane z pliku Excela *OrderNotes.xlsx* i otworzyć go, kliknij ikonę programu Excel na wstążce.
4. W oknie *Nawigator* wybierz zestaw danych *Sheet1*, po czym kliknij *Przekształć dane* (rysunek 5.38):



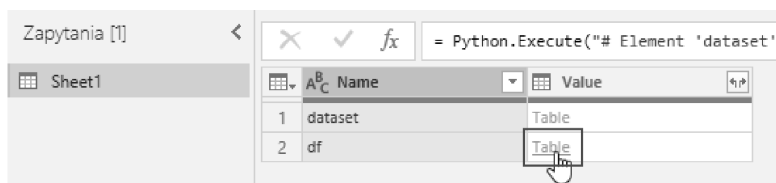
Rysunek 5.38. Ładowanie notatek dotyczących zamówień z arkusza Excela i przekształcanie danych

5. Zadeklaruj pierwszy wiersz załadowanych danych jako nagłówki kolumn. W tym celu kliknij opcję *Użyj pierwszego wiersza jako nagłówków* (rysunek 5.39):



Rysunek 5.39. Przycisk Użyj pierwszego wiersza jako nagłówków

6. Następnie przejdź do menu *Przekształć* i kliknij *Uruchom skrypt języka Python*.
7. Otwórz skrypt Pythona *01-order-notes-parserpython.py* z folderu *Chapter05\extracting-values-from-text-using-regex*. Skopiuj zawartość skryptu i wklej go do edytora skryptów Pythona, a następnie kliknij *OK*.
8. W przypadku problemów z poziomami zgodności zestawów danych otwórz okno *Ustawienia źródła danych* i za pomocą opcji *Edytuj uprawnienia...* ustaw uprawnienia wszystkich zestawów danych na *Organizacyjne*. Następnie kliknij *Odswież podgląd*.
9. Interesuje nas tylko zestaw danych *df*. Kliknij więc jego wartość *Table* (rysunek 5.40):

Rysunek 5.40. Wybieranie zestawu danych *df* otrzymanego w wyniku transformacji za pomocą skryptu Pythona

10. Łatwo zauważyć, że wynikowa tabela zawiera trzy dodatkowe kolumny, z których każda jest powiązana z nazwaną grupą (rysunek 5.41):

OrderNumber	Notes	1.2 RefundAmount	RefundReason	RefundDate
1	ORD000001 5,00 EUR Kradzież przy dostawie opłaconej przelewem 11/02/2021	5	Kradzież przy dostawie opłaconej przelewem	2021-02-11
2	ORD000002 29,00 EUR Zwrot za kradzież w dostawie 04/06/2020	29	Zwrot za kradzież w dostawie	2020-06-04
3	ORD000003 53,00€ Zwrot za kradzież w dostawie 24/09/2020	53	Zwrot za kradzież w dostawie	2020-09-24
4	ORD000004 29/10/2020 EUR45,00 Zwrot za kradzież w dostawie	45	Zwrot za kradzież w dostawie	2020-10-29
5	ORD000005 EUR 522,00 Zwrot za kradzież w dostawie 20/08/2020	522	Zwrot za kradzież w dostawie	2020-08-20
6	ORD000006 € 266,00 - Kradzież w dostawie opłaconej przelewem w dniu 10/12/20...	266	Kradzież w dostawie opłaconej przelewem w dniu	2020-12-10
7	ORD000007 EUR68,50 - Zwrot za kradzież w dostawie 02/07/2020	68,5	Zwrot za kradzież w dostawie	2020-07-02
8	ORD000008 EUR 50,00 - Zwrot za kradzież przy dostawie z dnia 30/07/2020	50	Zwrot za kradzież przy dostawie z dnia	2020-07-30
9	ORD000009 30/07/2020 209,00 € - Zwrot za kradzież w dostawie	209	Zwrot za kradzież w dostawie	2020-07-30

Rysunek 5.41. Wartości wydobyte z notatek zapisanych w formacie dowolnego tekstu za pomocą regeksa oraz kodu w Pythonie

Na koniec przejdź do menu głównego i kliknij *Zamknij i zastosuj*.

Super! Właśnie udało Ci się zreorganizować dane zawarte w notatkach dotyczących zamówień za pomocą regeksów oraz Pythona. Twój szef będzie bardzo zadowolony!

Korzystanie z regeksów w usłudze Power BI do wyodrębniania wartości za pomocą języka R

W języku R do zarządzania nazwanymi grupami, które są wielokrotnie wykorzystywane w tym samym regeksie, możemy używać pakietu `namedCapture`. Aby to zrobić, należy umieścić przed nazwaną grupą modyfikator `(?J)` (pozwala to na współdzielenie tej samej nazwy przez wiele nazwanych grup przechwytyjących). W przeciwieństwie do języka Python w języku R funkcja `str_match_named()` z pakietu `namedCapture` nie zwraca jednorazowego wyniku przechwyconego przez nazwaną grupę. Zwraca tyle kolumn, ile razy nazwana grupa została użyta (rysunek 5.42):

	A _C RefundAmount	A _C RefundReason	A _C RefundDate	A _C RefundAmount.1	A _C RefundReason.1	A _C RefundDate.1	A _C RefundDate.2	A _C RefundAmount.2
1				5,00	Kradzież przy dostawie opłaconej przelewem	11/02/2021		
2				29,00	Zwrot za kradzież w dostawie	04/06/2020		
3				53,00	Zwrot za kradzież w dostawie	24/09/2020		
4							29/10/2020	45,00
5	532,00	Zwrot za kradzież w dostawie	20/08/2020					
6	266,00	Kradzież w dostawie opłaconej przelewem w dniu	10/12/2020					
7	68,50	Zwrot za kradzież w dostawie	02/07/2020					
8	50,00	Zwrot za kradzież przy dostawie z dnia	30/07/2020					
9								

Rysunek 5.42. Funkcja `str_match_named()` zwraca tyle kolumn, ile razy została użyta nazwana grupa

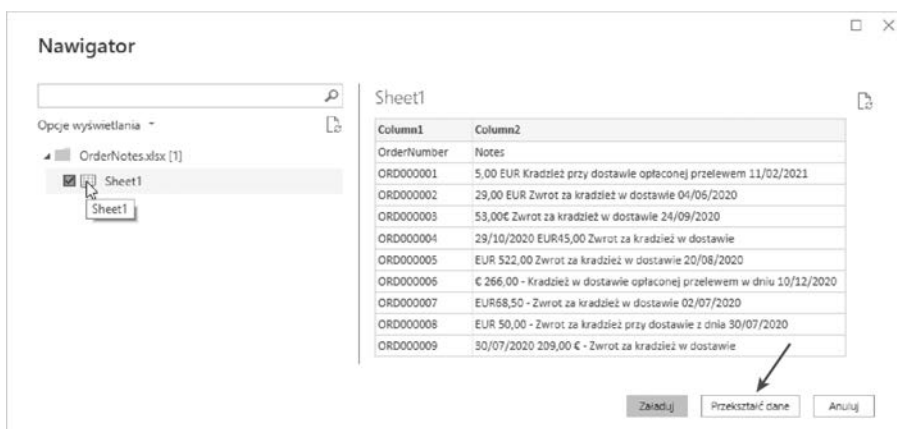
Z tego powodu trzeba nieco przetworzyć wynik. Po pierwsze, należy zastąpić puste znaki wartością pustą NA, a po drugie, zastosować funkcję `coalesce()` z pakietu `dplyr`, która scala wiele kolumn z zachowaniem wartości różnych od null.

Ważna uwaga

Zwróciłem uwagę na to ograniczenie Toby'emu Dylanowi Hockingowi, autorowi pakietu `namedCapture`, który niedawno zaimplementował funkcję wewnątrz nowej wersji pakietu o nazwie `nc`. Szczegóły implementacji można znaleźć na stronie <https://github.com/tdhock/namedCapture/issues/15>. W chwili pisania tej książki nowa wersja pakietu `nc` nie została jeszcze opublikowana w dystrybucji CRAN. Dlatego uznałem za stosowne korzystanie z pakietu `namedCapture` w przykładowym kodzie. Możesz jednak swobodnie korzystać w swoich przyszłych projektach z nowego pakietu `nc`.

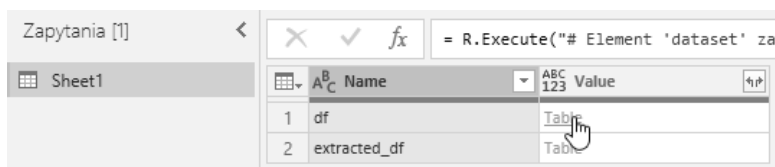
Zacznijmy wyodrębniać wartości z notatek dotyczących zamówień przy użyciu języka R w usłudze Power BI:

1. Otwórz program Power BI Desktop i upewnij się, że korzystasz z najnowszego silnika języka R (w naszym przypadku jest to dystrybucja CRAN R 4.0.2).
2. Aby zaimportować dane z programu Excel, kliknij ikonę Excela na wstążce i otwórz plik *OrderNotes.xlsx* dostępny w folderze *Chapter05*.
3. W oknie *Nawigator* wybierz zestaw danych *Sheet1*, po czym kliknij *Przekształć dane* (rysunek 5.43):



Rysunek 5.43. Ładowanie notatek dotyczących zamówienia z Excela i przekształcanie danych

4. Zadeklaruj pierwszy wiersz załadowanych danych jako nagłówki kolumn. W tym celu kliknij pozycję *Użyj pierwszego wiersza jako nagłówków*.
5. Następnie przejdź do menu *Przekształć* i kliknij *Uruchom skrypt języka R*.
6. Otwórz skrypt języka R z pliku *02-order-notes-parser-r.R* w folderze *Chapter05\extracting-values-from-text-usingregex*. Skopiuj jego zawartość i wklej ją do edytora skryptów R. Następnie kliknij *OK*.
7. Interesuje nas tylko zestaw danych *df*. Kliknij powiązaną z nim wartość *Table* (rysunek 5.44):



Rysunek 5.44. Wybieranie zestawu danych *df* uzyskanego w wyniku transformacji za pomocą skryptu języka R

8. Łatwo zauważyć, że wynikowa tabela zawiera trzy dodatkowe kolumny, a każda jest powiązana z nazwaną grupą (rysunek 5.35):

	OrderNumber	Notes	RefundAmount	RefundDate	RefundReason
1	ORD000001	5,00 EUR Kradzież przy dostawie opłaconej przelewem 11/02/2021	5,00	11/02/2021	Kradzież przy dostawie opłaconej przelewem
2	ORD000002	29,00 EUR Zwrot za kradzież w dostawie 04/06/2020	29,00	04/06/2020	Zwrot za kradzież w dostawie
3	ORD000003	53,00€ Zwrot za kradzież w dostawie 24/09/2020	53,00	24/09/2020	Zwrot za kradzież w dostawie
4	ORD000004	29/10/2020 EUR45,00 Zwrot za kradzież w dostawie	45,00	29/10/2020	Zwrot za kradzież w dostawie
5	ORD000005	EUR 522,00 Zwrot za kradzież w dostawie 20/08/2020	522,00	20/08/2020	Zwrot za kradzież w dostawie
6	ORD000006	€ 266,00 - Kradzież w dostawie opłaconej przelewem w dniu 10/12/2020	266,00	10/12/2020	Kradzież w dostawie opłaconej przelewem w dniu
7	ORD000007	EUR68,50 - Zwrot za kradzież w dostawie 02/07/2020	68,50	02/07/2020	Zwrot za kradzież w dostawie
8	ORD000008	EUR 50,00 - Zwrot za kradzież przy dostawie z dnia 30/07/2020	50,00	30/07/2020	Zwrot za kradzież przy dostawie z dnia
9	ORD000009	30/07/2020 209,00 € - Zwrot za kradzież w dostawie	209,00	30/07/2020	Zwrot za kradzież w dostawie

Rysunek 5.45. Wartości wydobyte z notatek zapisanych w formacie dowolnego tekstu za pomocą regreksa oraz kodu w R

Na koniec przejdź do menu głównego i kliknij *Zamknij i zastosuj*.

Zdumiewające! Właśnie udało Ci się zreorganizować dane zawarte w notatkach dotyczących zamówienia za pomocą regeksów oraz języka R.

Podsumowanie

W tym rozdziale zapoznałeś się z podstawami używania regeksów. Korzystając z absolutnego minimum, skutecznie zweryfikowałeś ciągi reprezentujące adresy e-mail i daty w usłudze Power BI z wykorzystaniem zarówno języka Python, jak i R.

Ponadto dowiedziałeś się, jak za pomocą regeksów wyodrębniać informacje z częściowo ustrukturyzowanych plików logów oraz jak importować wyodrębnione informacje w ustrukturyzowany sposób do usługi Power BI.

Wreszcie nauczyłeś się, jak używać regeksów w Pythonie i R do wyodrębniania informacji z pozornie nieprzetworzonego, dowolnego tekstu na przykładzie notatek związanych z zamówieniami sprzedaży.

W następnym rozdziale dowiesz się, jak używać w usłudze Power BI wybranych technik deidentyfikacji w celu anonimizacji lub pseudonimizacji zestawów danych zawierających wrażliwe dane osobowe w postaci zwykłego tekstu przed ich zaimportowaniem do usługi Power BI.

Bibliografia

Aby uzyskać dodatkowe informacje, zapoznaj się z następującymi książkami i artykułami:

1. Jan Goyvaerts, *Regular Expressions: The Complete Tutorial*, (<https://www.princeton.edu/~mlovett/reference/Regular-Expressions.pdf>).
2. *Data Privacy Settings In Power BI/Power Query, Part 1: Performance Implications* (<https://blog.crossjoin.co.uk/2017/05/24/data-privacysettings-in-power-bipower-query-part-1-performanceimplications/>).

Skorowidz

A

- adres e-mail
 - zapis do plików Excela, 202
 - zapisywanie w bazie danych, 217
 - zapisywanie w plikach CSV, 191
 - zapisywanie w plikach Excela, 199
- adres IP
 - statyczny, 210
- adresy e-mail
 - sprawdzanie poprawności, 136, 143
 - sprawdzanie poprawności, 146
- aktywacja środowiska, 72
- algorytm
 - Amelia II, 364
 - KNN, 385
 - MICE, 364
 - NER, 176
- Anaconda, 65
- Anaconda Prompt, 70, 71, 115
- analiza danych
 - eksploracyjna, 415
- analiza EDA
 - w Power BI, 420
- analiza
 - jednowymiarowa zmiennej, 430, 432
 - obrazów, 380
 - tekstu, 380
 - wariancji ANOVA, 331
 - wydzwięku, 410
 - zestawu danych, 226
- Anomaly Detection, 358
- anonimizacja, 167, 171, 172, 173

- Apache
 - logi dostępu, 153, 154, 156
- API, Application Programming Interfaces, 247
- AUC-PR, Area Under the Precision-Recall Curve, 386
- AUC-ROC, Area Under the ROC Curve, 386
- Azure AutoML, 378, 380, 383
 - konfigurowanie przebiegu, 405
 - modele o najlepszej wydajności, 406
 - potoki treningowe, 406
 - szkolenie modelu, 401
 - tworzenie eksperymentu, 404
- Azure Cognitive Services, 379
- Azure Machine Learning, 378
- Azure ML Studio, 402
- Azure SQL
 - zapisywanie danych, 219
- Azure SQL Database, 204
 - nawiązywanie połączenia, 211
 - nowa baza danych, 207
 - zapisywanie dat i adresów e-mail, 217
- Azure SQL Server
 - firewall, 210
 - zapisywanie danych, 204, 212, 220
- Azure Text Analytics, 401

B

- baza danych
 - Azure SQL Database, 207
 - tworzenie, 207
 - wybieranie, 208

BeautifulSoup, 73
 biblioteka, *Patrz także* pakiet, moduł
 charlatan, 184
 Dask, 228
 DataExplorer, 419
 D-Tale, 419
 Dython, 332
 ExPanDaR, 419
 Geocoder, 257
 ggquickeda, 419
 Lux, 419
 Matplotlib, 114, 115, 121, 174
 missingno, 366
 MKL, 39
 NumPy, 73, 114
 openpyxl, 275
 openxlsx, 200
 Pandas Profiling, 419
 Pandas, 73
 PandasGUI, 419
 Plotly.js, 458
 Plotly.R, 458
 purrr, 178, 260
 PyCaret, 382, 387
 Requests, 73
 SciPy, 73
 Seaborn, 317
 SweetVIZ, 419
 Tidyverse, 96, 193, 420
 BIFF, binary interchange file format, 195
 Bing Maps
 Locations, 252
 Web Services, 249
 brama danych, 56, 59
 Build Tools, 384

C

chmura, 204, 383
 CLF, Common Log Format, 153
 Cognitive Services, 378, 408
 Conda, 65, 70
 CRAN
 R, 27, 38
 Time Machine, 40, 41
 CRLF, Carriage Return Line Feed, 188
 CSV, comma separated values, 188

D

dane
 analiza eksploracyjna, 415
 anonimizacja, 167, 171–173, 176
 deidentyfikacja, 166
 generalizacja, 168
 haszowanie, 169
 maskowanie, 167
 oczyszczanie, 417
 pobieranie z Internetu, 324
 pseudonimizacja, 170, 180, 184
 sprawdzanie poprawności, 142
 szyfrowanie, 169
 tokenizacja, 169
 zamiana, 168
 zapisywanie
 do pliku CSV, 188, 189, 192
 do pliku Excela, 195
 na serwerze, 219
 zniekształcanie, 169
 dane osobowe, 166
 Dask, 228
 instalowanie na laptopie, 228
 tworzenie obiektów DataFrame, 229
 wyodrębnianie informacji, 230
 DataExplorer, 419, 421
 DataFrame, 229
 lista adresów do geokodowania, 253
 wyodrębnianie informacji z obiektu, 230
 daty
 sprawdzanie poprawności, 139, 149
 zapisywanie do plików CSV, 194
 DDL, Data Definition Language, 215
 debugowanie kodu wizualizacji, 56, 88
 deidentyfikacja danych, 166
 deserializowanie, 118, 123, 391
 pliku RDS, 108
 DetectLanguage, 380
 dodatek
 Build Tools, 384
 Power Pivot, 196
 Power Query, 196, 384
 dystrybucja
 Anaconda, 65
 CRAN R, 38, 45
 Microsoft R Open, 38, 42
 Miniconda, 27, 65, 68
 dziel i zwyciężaj, 233

E

EDA, Exploratory Data Analysis, 415
 edycja
 obszaru objects, 468
 sekcji dataRoles, 465
 węzła dataViewMappings, 466
 zawartości pliku, 465
 edytor
 Power Query, 104
 skryptów
 Pythona, 26, 30
 R, 27
 eksperyment AutoML, 406
 eksploracja
 jednowymiarowa zbioru, 425
 wielowymiarowa, 433
 eksploracyjna analiza danych, EDA, 415
 eksplorator obiektów RStudio, 220
 entropia, 329
 warunkowa, 329
 ETL, extract, transform, load, 187
 Excel
 zapis dat i adresów e-mail, 202
 zapisywanie danych, 195, 200
 ExtractKeyPhrases, 380

F

format
 CLF, 153
 JSON, 253
 NCSA, 153
 XLS, 195
 XLSX, 196
 ZIP, 196
 fragmentator, 124
 rozwijany, 396
 ustawianie opcji, 394
 wybieranie typu, 394
 fragmentatory strony Powiązania, 440
 framework PyData, 114
 funkcja
 distGeo(), 281
 distHaversine(), 281
 funkcja
 accuracy(), 373
 airportLatLongList(), 276
 analize_sentiment(), 410
 anonymizer.anonymize(), 173, 181
 apply(), 173, 254, 255

associations(), 333
 availableCores(), 238, 262
 bind_cols(), 260
 bing_geocode_via_address(), 256, 259, 260
 calc_corr(), 335, 336
 circular_grouped_barplot(), 446, 448
 coalesce(), 162
 collect(), 241
 compare_models(), 385, 386
 compute(), 233, 257
 connect(), 212
 coord_polar(), 450, 451
 corr(), 319
 correlate(), 320
 correlation_ratio(), 335
 csv_to_disk.frame(), 239
 curlPercentEncode(), 259
 cursor.execute(), 216
 dataframeHist(), 356
 dbAppendTable(), 221
 dbConnect(), 219
 descr(), 422
 dfSummary(), 421
 disk.frame(), 240
 distGeo(), 281
 enrich_with_geocoding(), 256
 future_map(), 261
 future_map_*, 261
 future_map_dfr(), 262
 geodistance(), 277
 get(), 252
 ggplotly(), 458, 459
 introduce(), 421
 iterrows(), 216
 karney(), 276
 list.files(), 239
 load_model(), 389
 map(), 178
 map_chr(), 178
 map_dfr(), 260
 map2(), 281
 miss_var_summary(), 366
 pandas read_excel(), 173
 paste0(), 156
 pchisq(), 357
 predict_model(), 387, 389
 pull(), 104, 260
 read_csv(), 189, 193, 256, 280
 read_csv2(), 193
 read_delim(), 193
 read_lines(), 156

read_sql(), 213
 read_tsv(), 193
 read_xls(), 201
 read_xlsx(), 201, 280
 rescale100(), 448
 reset_index(), 232
 setup(), 385, 386
 split(), 276
 srckeep(), 240
 statsNA(), 368
 str_match_named(), 156, 162
 sum_expr(), 302
 tic(), 260
 to_csv(), 190, 235
 toc(), 260
 train_test_split(), 385
 truediv(), 232
 trygonometryczna haversine, 272
 upsetplot_miss(), 367
 vis_miss(), 366
 visualize(), 231
funkcje
 AutoML, 381
 celu, 286
 RemixAutoML, 383
 sztucznej inteligencji, 379
 dla przepływów danych, 379
 usług Cognitive Services, 380
 usługi Azure Machine Learning, 380
 wektoryzowane, 277

G

galeria htmlwidgets, 460
 generator fałszywych danych, 181
 geokodowanie adresów, 251, 258, 265
 GUI, 264

H

haszowanie, 169
 histogram, 352, 353, 357, 426

I

IDE, integrated development environment,
 27, 37
 instalowanie środowiska, 50, 79
ikona
 Database Engine..., 214
 fragmentatora, 110

Raport, 110
 Sformatuj element wizualny, 110, 124
 Uruchom skrypt, 32
 Variables, 118
 Wizualizacja języka Python, 125
 wizualizacji fragmentatora, 392
implementacja
 algorytmów imputacji brakujących, 369
 wykresu kołowego, 446, 452
 imputacji wielokrotnej, 364
importowanie
 danych, 28
 dużego zestawu danych, 227, 235, 237,
 242
 logów dostępu, 154, 156
 pliku
 PKL, 113, 120
 RDS, 96, 104, 106
 z Excela, 150, 160, 162, 175
 zdeserializowanej ramki danych, 105
imputacja
 danych tabelarycznych, 370
 jednowymiarowych szeregów czasowych,
 364
 MICE, 364
 pojedyncza, 362
 ręczna, 361
 wielokrotna, 363, 364
 wielowymiarowa szeregów czasowych, 365
indeks pakietów Pythona, 65
instalacja
 Dask, 228
 dystrybucji
 CRAN R, 45, 47
 Miniconda, 68
 jądra IPython, 116
 języka
 Python, 26
 R, 27
 lokalnej bramy danych, 56, 58
 MRO, 39, 41
 pakietów PyData, 115
 pakietu, 73
 disk.frame, 238
 pbiviz, 461
 reticulate, 80
 programu
 RStudio, 50
 SQL Server Express, 205
 silnika Pythona, 67
 Tidyverse, 97

interfejs
 API Bing Maps Locations, 251
 API Text Analytics, 409, 410
 interfejsy zewnętrzne API, 247

J

jądro ipykernel, 116
 język
 DDL, 215
 M, 264
 R
 ograniczenia wizualizacji, 61
 WSDL, 248
 JVM, Java Virtual Machine, 228

K

karta
 Formatowanie, 113
 Narzędzia główne, 176, 180
 Opcje i ustawienia, 54, 86
 Transformacja, 29
 klaster
 Dask, 228
 obliczeniowy, 403
 klasy znaków, 133
 klasyfikatory, 386
 klient Microsoft R, 42
 kodowanie procentowe, 259
 kompilacja niestandardowej wizualizacji, 470
 konfigurowanie
 API Text Analytics, 408
 języka
 Python, 64, 82
 R, 37
 obsługi Pythona, 80
 programu Power BI Desktop, 54, 86
 przebiegu AutoML, 405
 środowiska
 IDE, 87
 Pythona, 87
 usługi
 Compute, 209
 Power BI, 88
 korelacja, 308
 implementacja
 obliczeń w Pythonie, 317, 323, 332
 obliczeń w R, 320, 323, 334, 337
 mapa termiczna, 334, 337

między zmiennymi kategorycznymi
 a liczbowymi, 325
 międzyklasowa, 331
 Pearsona
 rozkłady Boigelota, 311
 Spearmana
 rozkłady Boigelota, 315
 współczynnik
 Craméra, 328
 Kendalla, 316
 Pearsona, 310, 331
 Spearmana, 313
 wzór
 Craméra, 328
 Kendalla, 316
 Pearsona, 310
 Spearmana, 314
 zmienne kategoryczne, 325

kurtoza, 422
 kwantyfikatory, 133

L

leniwe wartościowanie, lazy evaluation, 230
 logi, 152
 dostępu do serwera, 153, 154, 156
 LP, Linear Programming, 283

Ł

ładowanie danych, 24
 łącznik Internet, 28

M

Machine Learning Services, 205
 macierz, 289
 funkcji celu, 290
 mapa termiczna korelacji, 439
 maskowanie danych, 167
 menedżer pakietów
 conda, 70
 pip, 70
 metody imputacji
 EMWA, 365
 imputacja
 rozłożona sezonowo, 365
 wielowymiarowa, 365
 interpolacja, 365
 LOCF, 365

MICE, 364
 Microsoft
 R Client, 42
 R Open, 27, 38
 SQL Server Management Studio, 210
 migawki, 40, 41
 MILP, Mixed-Integer Linear Programming, 287
 Miniconda, 65
 instalacja, 68
 MKL, Math Kernel Library, 39
 model uczenia maszynowego
 deserializacja, 388
 metryki klasyfikacji, 386
 o najlepszej wydajności, 406
 ocenianie
 obserwacji, 392
 w usłudze Power BI, 390
 serializacja, 388
 skrypty wizualizacji, 390
 szkolenie, 384
 w Azure AutoML, 401
 wydajność, 386
 moduł, *Patrz* pakiet
 monotoniczność powiązania, 314
 MRAN, Microsoft R Application Network, 38, 39
 migawki, 40, 41
 MRO, Microsoft R Open, 38
 instalacja, 39, 41
 sprawdzanie wersji, 47
 wielowątkowość, 39

N

narzędzia wiersza polecenia, 70
 narzędzie
 Anaconda Prompt, 71
 AutoML, 382
 CRAN Time Machine, 41
 nawiasy
 klamrowe, 116
 kwadratowe podwójne, 231
 nierówność liniowa, 286
 NumPy, 73, 114

O

obiekt
 DataFrame, 229, 234, 237, 449
 disk.frame, 238, 240
 tibble, 100

obliczanie odległości, 269
 implementacja
 w Pythonie, 275, 278
 w R, 279, 282
 między dwoma punktami, 272, 273
 prawo odległości Haversinesa, 272
 wzory Karneya, 276
 wzór Vincenty'ego, 273
 obliczenia równoległe, 234
 Object Explorer, 211
 obsługa wartości
 brakujących, 361
 odstających, 342
 obszar wykonalności, feasible region, 286, 287
 odległość
 euklidesowa, 346
 Mahalanobisa, 347, 353, 356
 odrzucanie danych, 361
 ograniczenia, 34
 wizualizacji, 89
 języka R, 61
 okno
 Change R Version, 53
 Connect to Server, 211
 Console, 300
 Installation Complete, 69
 Interactive, 117, 118, 212
 Jupyter, 117
 logowania, 59
 Nawigator, 105, 120, 144
 Opcje, 54, 86, 298
 Open file, 403
 Open Folder, 83
 Otwieranie, 144, 184
 Pobierz dane, 25
 Poziomy prywatności, 145
 Scalanie, 298
 Uruchom skrypt języka R, 31
 Ustawienia źródła danych, 161
 Wizualizacje, 31
 Włącz elementy wizualne skryptu, 112
 opcja
 ####Opublikuj w Internecie, 62
 Edytuj uprawnienia..., 161
 General Purpose, 209
 Select interpreter, 84
 Set server firewall, 209
 Skrypt, 25
 Wizualizacja języka Python, 31
 Wybór pojedynczy, 110
 Zaimportuj dane z pliku programu
 Excel, 303

opcje fragmentatora, 394
 operator
 ::, 100
 OR, 132
 optymalizacja, 291
 liniowa, 285
 Outliers Detection, 358

P

PaaS, Platform as a Service, 204
 pakiet

 BeautifulSoup, 73
 checkpoint, 41
 corr, 320
 DBI, 219
 disk.frame, 237, 238, 261
 dplyr, 41, 162, 240, 452
 ellipsoidalKarney, 276
 Faker, 181, 184
 forecast, 366, 373
 fst, 237
 furr, 261
 pakiet geographiclib, 275
 ggplot, 474
 ggpubr, 366
 htmlwidgets, 460
 imputeTS, 366, 373
 mice, 366
 miceadds, 366
 missForest, 366
 namedCapture, 156, 157, 162
 naniar, 366
 nominal, 333
 odbc, 219
 pbiviz, 461
 pickle, 115
 Presidio, 174, 176
 PyData, 115
 PyGeodesy, 275, 276
 pyodbc, 212
 PyYAML, 73
 RCurl, 259
 readr, 156, 192, 193
 readxl, 200
 requests, 252
 reticulate, 80, 180
 rlang, 452
 scales, 447
 scikit-learn, 114

 statsmodels, 332
 summarytools, 419, 421, 422
 tictoc, 259
 tidy evaluation, 452
 tidygeocoder, 262, 263
 tidyr, 99
 timetk, 101
 upsetplot, 366, 368
 pamięć RAM, 225
 Pandas, 73, 114
 panel
 Formatuj element wizualny, 467
 Global Options, 184, 219
 parametr
 what-if, 395, 397
 zmiana nazwy, 395, 397, 398
 pasek trybu, 458
 pip, 65, 70
 instalacja pakietów, 73
 plik
 Rprofile.site, 49
 YAML, 76, 78
 pliki
 .df, 239
 .fst, 239
 CSV, 116, 188
 importowanie, 403
 wyodrębnianie informacji, 226
 logów, 152
 PKL, 116, 117
 deserializacja, 123
 importowanie w Pythonie, 113
 importowanie w wizualizacjach, 121
 w usłudze Power BI, 119
 RDS, 96, 99, 101
 deserializowanie zawartości, 108
 importowanie do Power Query, 104
 importowanie w wizualizacji R, 106
 w usłudze Power BI, 103
 pobieranie danych, 324
 polecenie
 conda activate pbi_powerquery_env, 72, 160
 conda init, 116
 conda list, 73
 conda search python, 71
 install.packages, 41
 pip install, 73
 pip install openpyxl, 275
 python --version, 72
 where python, 175
 połączenia z Azure SQL Database, 211

- połączenie
 - conn, 215
 - parametry, 212
 - tworzenie, 212
 - z Azure SQL Database, 210, 213, 218, 220
 - z SQLExpress, 206, 214
 - zamknięcie, 217, 222
 - zmiana parametrów, 219
- potok, 97
- Power BI
 - analiza EDA, 420
 - anonimizacja danych, 172
 - biblioteka PyCaret, 387
 - geokodowanie adresów, 265
 - importowanie
 - logów dostępu, 154, 156
 - pliku PKL, 120
 - wizualizacji niestandardowej, 471
 - imputacja brakujących wartości, 374
 - interfejs API Text Analytics, 410
 - kołowy wykres słupkowy, 452
 - konfigurowanie usługi, 56
 - modele Azure ML, 407
 - obliczanie
 - odległości, 278, 282
 - współczynników korelacji, 323, 337
 - obsługa języka R, 56, 88
 - pliki
 - PKL, 119
 - RDS, 103
 - pseudonimizacja danych, 180
 - raport EDA, 427
 - rozwiązywanie problemu LP, 295, 303
 - sprawdzanie poprawności danych, 142
 - uczenie maszynowe, 379
 - usługi sieciowe, 263
 - wizualizacje niestandardowe, 460
 - wykrywanie wartości odstających, 358
 - wyodrębnianie wartości z tekstu, 157
 - wrażenia regularne, 128
- Power BI Desktop, 24, 34, 37, 38, 43, 45, 120, 202
 - importowanie
 - danych z Excela, 144
 - pliku RDL, 108
 - obsługa
 - języka R, 54
 - Pythona, 86
- Power BI Embedded, 34
- Power BI free, 38, 64
- Power BI Mobile, 35
- Power BI Report Server, 34
- Power BI Service, 34, 44
- Power BI Visual Project, 460
- Power BI Visual Tools, 460
- Power Pivot, 196
- Power Query, 109, 187, 196, 384
 - importowanie
 - pliku PKL, 120
 - pliku RDS, 104
 - uczenie maszynowe, 388
 - usługi
 - Cognitive Services, 408
 - sieciowe, 401
- Power Query Editor, 30
- powiązania, 438, *Patrz także* korelacja
 - forma, 309
 - kierunek, 309
 - monotoniczne, 314
 - niemonotoniczne, 314
 - siła, 310
 - odkrywanie, 418
- prawo odległości Haversinesa, 272
- problem
 - LP, 288, 289, 291
 - MILP, 287
 - optymalizacji, 291
- program
 - Object Explorer, 211
 - Power BI Desktop, 45
 - rGUI, 46
 - RStudio, 50, 55, 80
 - SQL Server Express, 205
 - Visual Studio Code, 82
 - VS Code, 212
- programowanie liniowe, LP, 283
- protokół SOAP, 248
- przekształcanie danych, 27
- przekształcenie yeo-johnsona, 431
- przepływ danych, 379
- przetwarzanie opóźnione, delayed computation, 230
- przycisk
 - Create SQL database, 208
 - Download ZIP, 77
 - Finish, 69
 - Przekształć dane, 29
 - Uruchom skrypt języka Python, 29, 297
 - Uruchom skrypt języka R, 30
 - Zamknij i zastosuj, 31, 176, 180, 298
 - Zarządzanie relacjami, 109, 123
- pseudonimizacja danych, 170, 180, 184

- punkt kontrolny, 40
 - PyCaret, 382, 387
 - modele ML, 384
 - w usłudze Power BI, 387
 - PyData, 114
 - PyPI, Python Package Index, 65
 - Python
 - aktualizacja silnika, 79
 - analiza EDA, 418
 - anonimizacja danych, 173
 - geokodowanie adresów, 251, 265
 - importowanie
 - dużego zestawu danych, 235
 - dużych zestawów danych, 227
 - logów dostępu, 154
 - plików PKL, 113
 - instalowanie
 - silników, 67
 - środowiska IDE, 79
 - konfigurowanie języka, 64
 - obliczanie
 - odległości, 275, 278
 - współczynników korelacji, 317, 323, 332, 337
 - ograniczenia wizualizacji, 89
 - Power BI Desktop, 86
 - pseudonimizacja danych, 181
 - rozwiązywanie problemu LP, 291, 295
 - serializowanie obiektu, 115, 116
 - silniki, 65
 - sprawdzanie poprawności
 - adresów e-mail, 143
 - poprawności dat, 149
 - uczenie maszynowe, 382
 - wybór silnika, 66
 - wykrywanie wartości odstających, 349
 - wyodrębnianie wartości z tekstu, 160
 - zapisywanie
 - danych do serwera, 212
 - dat i adresów e-mail, 217
 - do plików CSV, 189
 - do plików Excela, 196, 199
 - PyYAML, 73
- ## R
- R
 - analiza EDA, 418
 - anonimizacja danych, 176
 - geokodowanie adresów, 258, 265
 - importowanie
 - dużego zestawu danych, 242, 237
 - logów dostępu, 156
 - pliku RDS, 96, 106
 - instalowanie
 - silników, 45
 - środowiska IDE, 50
 - kołowy wykres słupkowy, 446
 - konfigurowanie języka, 37
 - obliczanie
 - odległości, 279, 282
 - współczynników korelacji, 320, 323, 334, 337
 - pseudonimizacja danych, 184
 - ramka danych, 106
 - rozmiar danych, 107
 - rozwiązywanie problemu LP, 300, 303
 - serializacja obiektu, 97
 - silniki, 38
 - sprawdzanie poprawności
 - adresów e-mail, 146
 - dat, 151
 - widzety HTML, 460
 - wizualizacje
 - danych, 413
 - niestandardowe, 455, 462
 - wybór silnika, 43
 - wykrywanie wartości odstających, 355
 - wyodrębnianie wartości z tekstu, 162
 - zapisywanie
 - danych na serwerze, 219
 - dat i adresów e-mail, 222
 - do plików Excela, 200, 202
 - do plików CSV, 192, 194
 - do usługi Azure SQL, 222
 - ramka danych, 61, 89, 96
 - DataFrame, 234
 - importowanie, 105
 - raport EDA, 421, 424
 - analiza
 - jednowymiarowa, 431
 - powiązań, 441
 - wielowymiarowa, 438
 - w Power BI, 427
 - raporty
 - symulacji, 400
 - udostępnianie, 60
 - RDBMS, 42
 - regeksy
 - dopasowania zachłanne i leniwe, 134
 - flagi, 130

- klasy znaków, 133
- kropka, 134
- kwantyfikatory, 133
- logi dostępu, 156
- modyfikatory globalne, 131
- nazwane grupy przechwytyjące, 154
- operatory OR, 132
- pliki logów, 152
- sprawdzanie poprawności
 - adresów e-mail, 136, 143, 146
 - danych, 142
 - dat, 139, 149, 151
- wyodrębnianie wartości z tekstu, 157, 160, 162
- zarządzanie danymi, 159
- znaki
 - literalne, 130
 - specjalne, 131
 - zakotwiczenia, 131
- regex, 129
- relacja
 - między tabelami, 109, 124
 - utworzona automatycznie, 393
- repozytorium
 - GitHub, 77, 113, 153, 173, 275
 - MRAN, 40
 - PyPI, 70
- Requests, 73
- REST, REpresentational State Transfer, 248
- RESTful, 249
- reticulate
 - instalowanie pakietu, 80
- rozkład normalny, 319
- rozkłady Boigelota, 311, 315
- równanie liniowe, 283, 284
- RStudio, 55
 - domyślny silnik języka R, 52
 - instalacja programu, 50
 - obsługa Pythona, 80
 - pierwszy skrypt Pythona, 82
 - połączenie z bazą danych, 220
 - tworzenie projektu, 98
 - wrapper języka R, 56
 - wybór silnika języka R, 53

S

- scalanie zapytań, 298
- SciPy, 73
- ScoreSentiment, 380

- serializacja obiektu, 297
 - języka R, 97, 99
 - obiektu Pythona, 115
- serwer
 - MRAN, 39
 - punktów kontrolnych, 40
- serwis Bing Maps, 250
- silnik
 - Pythona, 65, 66
 - aktualizacja, 79
 - do transformacji danych, 68
 - instalowanie, 67
 - R, 38, 43, 49
 - instalowanie, 45
- skrót klawiaturowy
 - Ctrl + Shift + N, 99
 - Ctrl + Enter, 99
 - Ctrl + Shift + P, 84, 116
 - Shift + Enter, 117
 - Win + R, 462
- skrypty
 - w wizualizacjach, 33
 - wstrzykiwanie, 33
- słownik, 116
- SOAP, Simple Object Access Protocol, 248
- sprawdzanie poprawności
 - adresów e-mail, 136, 143, 146
 - danych, 142
 - dat, 139, 149, 151
- SQL Server, 204
- SQL Server
 - Express Edition, 204, 205
 - Replication, 205
- SSMS, SQL Server Management Studio, 204
 - połączenie
 - z Azure SQL Database, 218
 - z bazą danych, 211, 214
 - wyświetlanie zawartości tabeli, 219
- sterownik ODBC, 204, 206, 219
- symbol potoku, 97
- system plików
 - FAT, 189
 - NTFS, 189
- system RDBMS, 42
- szeregi czasowe
 - imputacja danych, 373
 - metody imputacji, 365
 - wykrywanie brakujących wartości, 368
- szyfrowanie, 169

Ś

- ścieżka
 - bezwzględna, 104
 - domyślna, 77
- średni błąd
 - bezwzględny, 373
 - kwadratowy, 373
- średnie odchylenie bezwzględne, 422
- środowisko
 - pbi_powerquery_env, 84, 116, 160, 174, 212, 317
 - pycaret_env, 387, 392, 409
 - RStudio, 52
 - wirtualne, 65
 - tworzenie, 72, 79

T

- tabela
 - ręczne wprowadzanie danych, 396, 399
 - tworzenie, 215
- TagImages, 380
- Text Analytics, 408
 - w Power BI, 410
- tidyr, 99
- Tidyverse, 96
 - instalacja, 97
- Time Machine, 48
- timetk, 101
- token, 252
- tokenizacja, 169
- transformacja
 - Boxa-Coxa, 348
 - danych, 61, 70
 - Yeo-Johnsona, 348, 473
- trójkąt sferyczny, 269
- tryb
 - cichy, silent mode, 385
 - osobisty, 56
- trygonometria sferyczna, 269
- twierdzenie cosinusów, 271
- tworzenie
 - bazy danych, 207
 - klastra obliczeniowego, 403
 - migawek, 40
 - niestandardowych wizualizacji, 460, 462
 - obiektów DataFrame, 229
 - plików
 - PKL, 116
 - RDS, 99
 - połączenia, 212, 219
 - raportów
 - Power BI, 90
 - Power BI Desktop, 90
 - środowiska wirtualnego, 72, 79
 - tabeli, 215
 - wizualizacji, 33
 - wykresu kołowego, 443
 - zestawu danych, 402
 - zserializowanych obiektów, 97

- typ
 - DataFrame, 229
 - tibble, 96

U

- uczenie maszynowe, 378
 - Power Query, 388
- udostępnianie raportów, 60, 89
- usługa
 - AI Insights, 380
 - Azure AutoML, 383
 - Azure Machine Learning, 378
 - Azure ML Studio, 402
 - Bing Maps Web Services, 249
 - Compute, 209
 - RESTful, 249
- usługi
 - Cognitive Services, 408
- usługi sieciowe, web services, 248
 - punkty końcowe, 249
 - w Power Query, 401
 - wdrażanie najlepszego modelu, 407
- usuwanie
 - typu listwise, 361
 - typu pairwise, 362
 - zmiennych, 362
- uwierzytelnianie
 - programu SQL Server, 206
 - SQL Server Authentication, 210
 - Windows, 206, 220

V

- Visual Studio Code
 - obsługa Pythona, 82
 - pierwszy skrypt Pythona, 83
 - uruchamianie skryptu, 117
- VS Code
 - debugowanie kodu wizualizacji, 88
 - edycja zawartości pliku, 465
 - widok zawartości folderu, 464

W

- walidacja krzyżowa, 386
- wartości brakujące, 359
 - diagnozowanie, 366
 - eksploracja, 424
 - implementacja algorytmów, 369
 - imputacja
 - danych szeregów czasowych, 373
 - danych tabelarycznych, 370
 - jednowymiarowych szeregów
 - czasowych, 364
 - pojedyncza, 362
 - ręczna, 361
 - w Power BI, 374
 - wielokrotna, 363
 - przyczyny, 360
 - usuwanie, 370
 - typu listwise, 361
 - typu pairwise, 362
 - zmiennych, 362
 - w szeregach czasowych, 368
- wartości odstające, 341
 - identyfikacja, 343
 - implementacja
 - w Power BI, 358
 - w Pythonie, 349
 - w R, 355
 - jednowymiarowe, 343
 - sposoby postępowania, 342
 - wielowymiarowe, 344
 - wykrywanie, 349, 355, 358
- wartość oczekiwana, 319, 322
- web scraping, 73
- wersje dostępne Pythona, 71
- węzeł Databases, 211, 214, 218
- widżety HTML, 460
- wiersz poleceń, 70
- wizualizacja, 460
 - danych, 31, 413
 - fragmentatora, 111, 124, 393
 - języka Python, 125
 - limity, 90
- języka R, 47, 56, 61
 - limity, 61
- niestandardowa
 - kompilacja, 470
 - w Power BI, 471
 - wykresy pudełkowe, 473, 474, 475
- regeksa, 129
- widżetów HTML, 460
- z wykorzystaniem przeszkolonego modelu, 392
- zaawansowana, 443
- WSDL, Web Service Definition Language, 248
- współczynnik
 - determinacji, 310
 - korelacji
 - Craméra, 328
 - Kendalla, 316
 - międzyklasowej, 331
 - Pearsona, 310, 331
 - Spearmana, 313
 - wewnątrzklasowej, 331
 - nieoznaczoności Theila, 333
 - niepewności Theila, 328
 - skośności, 422
 - zmienności, 422
- wstrzykiwanie skryptów, 24
- wykres
 - chmury deszczowej, raincloud plot, 426, 435, 456
 - kołowy, 443
 - kołowy słupkowy, 454
 - korelacji, 323
 - lizakowy, lollipop plot, 424
 - pudełkowy, 330, 349, 350, 356
 - punktowy, 319, 322, 346
 - skrzypcowy, 329
 - słupkowy, 417, 427, 450
 - szeregów czasowych, 102, 119
 - typu UpSet, 368
- wykrywanie
 - anomalii, 358
 - brakujących wartości, 368
 - wartości odstających, 349, 355, 358
- wyodrębnianie informacji, 230, 240
- wrażenia regularne, *Patrz* regeksy
- wzory Karneya, 276
- wzór
 - odległość Mahalanobisa, 347
 - współczynnik korelacji
 - Craméra, 328
 - Kendalla, 316
 - międzyklasowej, 331
 - Spearmana, 314
 - współczynnik niepewności Theila, 329
 - Haversinesa, 272
 - na odległość, 272
 - Vincenty'ego, 273

X

XLS, Excel Sheet, 195
XLSX, Excel Open XML Spreadsheet, 196

Y

YAML, 76

Z

zakładka
 Additional settings, 209
 AI Insights, 379
 Prywatność, 298
 Viewer, 458
 Wizualizacja, 471
zakres międzykwartylowy, 422
zamiana danych, 168
zapis
 danych do serwera, 204, 212
 do plików CSV, 188, 189
zapisywanie
 dat i adresów e-mail, 217
 informacji, 195
zapora firewall, 210
zbiór danych
 eksploracja
 jednowymiarowa, 425
 wielowymiarowa, 433
 mapa termiczna korelacji, 439

zestaw danych, 226
 importowanie, 237, 242
 raport EDA, 421
 rozmiar, 416
 tworzenie, 402
 wyodrębnianie informacji, 240
 zbiory szkoleniowe, 385
 zbiory testowe, 385

zmienne
 badanie powiązań, 308
 decyzyjne, 286
 kategoryczne, 325, 344
 liczbowe, 329, 344
 wartości odstające, 341
 powiązania, 438

znaki
 | >, 97
 CRLF, 188
 specjalne, 131
 zakotwiczenia, 131

Ż

żądanie GET, 251
 jawne, 259
 współbieżne, 255, 261

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 

Wyzwól potężną moc Power BI!

Ważnym zadaniem inżynierów danych jest kreowanie modeli uczenia maszynowego. Używa się do tego narzędzi do analizy biznesowej, takich jak Power BI. Możliwości Power BI są imponujące, a można je dodatkowo rozbudować. Jedną z ciekawszych metod wzbogacania modelu danych i wizualizacji Power BI jest zastosowanie złożonych algorytmów zaimplementowanych w językach Python i R. W ten sposób można nie tylko tworzyć interesujące wizualizacje danych, ale także pozyskiwać dzięki nim kluczowe dla biznesu informacje.

Dzięki tej książce dowiesz się, jak to zrobić. Zaczyniesz od przygotowania środowiska Power BI do używania skryptów w Pythonie i R. Następnie będziesz importować dane z nieobsługiwanych obiektów i przekształcać je za pomocą wyrażeń regularnych i złożonych algorytmów. Nauczysz się wywoływać zewnętrzne interfejsy API i korzystać z zaawansowanych technik w celu przeprowadzenia dogłębnych analiz i wyodrębnienia cennych informacji za pomocą narzędzi statystyki i uczenia maszynowego, a także poprzez zastosowanie optymalizacji liniowej i innych algorytmów. Zapoznasz się również z głównymi cechami statystycznymi zestawów danych i z metodami tworzenia różnych wykresów ułatwiających zrozumienie relacji między zmiennymi.

Najciekawsze zagadnienia:

- złożone przekształcanie danych w Power BI za pomocą skryptów Pythona i R
- anonimizacja i pseudonimizacja danych
- praca z dużymi zestawami danych
- wartości odstające i brakujące dla danych wielowymiarowych i szeregów czasowych
- tworzenie złożonych wizualizacji danych

Luca Zavarella zdobył certyfikat Azure Data Scientist Associate i tytuł Microsoft MVP w zakresie sztucznej inteligencji. Od ponad 10 lat zajmuje się Microsoft Data Platform. Posiada praktyczną wiedzę o stosie technologicznym Microsoft Business Intelligence (SSIS, SSAS, SSRS) i technikach hurtowni danych. Ostatnio interesuje się zaawansowanymi zagadnieniami analitycznymi i technikami data science. W wolnym czasie gra na fortepianie klasycznym.

	KOD KORZYŚCI Sięgnij po więcej!  
 helion.pl  HELION SA ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl	ISBN 978-83-283-9453-7  9 788328 394537
Cena: 119,00 zł	

Packt