

Python i praca z danymi

Wydanie III

Przetwarzanie, analiza, modelowanie i wizualizacja

Avinash Navlani
Armando Fandango
Ivan Idris

Helion 



Tytuł oryginału: Python Data Analysis: Perform data collection, data processing, wrangling, visualization, and model building using Python, 3rd Edition

Tłumaczenie: Krzysztof Sawka

ISBN: 978-83-283-8360-9

Copyright © Packt Publishing 2021. First published in the English language under the title 'Python Data Analysis - Third Edition' – (9781789955248).

Polish edition copyright © 2022 by Helion S.A.
All rights reserved.

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.

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)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/pyprda>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

Spis treści

Wstęp	13
--------------	-----------

Część I. Podstawy analizy danych

Rozdział 1. Wprowadzenie do bibliotek Pythona	21
Wyjaśnienie pojęcia „analiza danych”	22
Standardowy proces analizy danych	23
Proces KDD	24
Proces SEMMA	25
Proces CRISP-DM	25
Analiza danych a danetyka	27
Role analityka danych i danetyka	27
Umiejętności analityka danych oraz danetyka	28
Instalacja środowiska Python 3	30
Instalacja i konfigurowanie Pythona w systemie Windows	30
Instalacja i konfigurowanie Pythona w Linuksie	31
Instalacja i konfigurowanie Pythona w systemie macOS za pomocą instalatora graficznego	31
Instalacja i konfigurowanie Pythona w systemie macOS za pomocą menedżera Homebrew	31
Oprogramowanie używane w tej książce	32
Używanie aplikacji IPython jako powłoki	33
Odczytywanie stron podręcznika	35
Źródła informacji na temat bibliotek analizy danych w Pythonie	35
Korzystanie z aplikacji JupyterLab	36
Stosowanie aplikacji Jupyter Notebook	37

Zaawansowane funkcje aplikacji Jupyter Notebook	38
Skróty klawiszowe	38
Instalowanie innych jąder	38
Realizowanie poleceń powłoki	39
Rozszerzenia	40
Podsumowanie	44
Rozdział 2. Biblioteki NumPy i pandas	45
Wymogi techniczne	46
Tablice NumPy	46
Własności tablic	48
Wybieranie elementów tablicy	49
Numeryczne typy danych tablic NumPy	50
Obiekty dtype	52
Kody znakowe typów danych	52
Konstruktory dtype	53
Atrybuty dtype	53
Manipulowanie wymiarami tablic	54
Łączenie tablic NumPy	55
Rozdzielanie tablic NumPy	58
Zmiana typu danych tablic NumPy	60
Tworzenie widoków i kopii NumPy	61
Fragmentowanie tablic NumPy	62
Indeksowanie logiczne i indeksowanie specjalne	64
Rozgłaszanie tablic	65
Tworzenie obiektów DataFrame biblioteki pandas	67
Obiekt Series biblioteki pandas	68
Odczytywanie i kwerendowanie danych Quandl	71
Opisywanie obiektów DataFrame	74
Grupowanie i złączanie obiektów DataFrame	76
Praca z brakującymi danymi	80
Tworzenie tabel przestawnych	81
Rozwiązywanie kwestii dat	82
Podsumowanie	84
Bibliografia	85
Rozdział 3. Statystyka	86
Wymogi techniczne	87
Atrybuty i ich typy	87
Typy atrybutów	87
Atrybuty dyskretne i ciągłe	88
Pomiar tendencji centralnej	89
Średnia arytmetyczna	89
Dominanta	90
Mediana	90
Pomiar dyspersji	90
Skośność i kurtoza	93

Określanie związków za pomocą współczynników kowariancji i korelacji	94
Współczynnik korelacji Pearsona	94
Współczynnik korelacji rang Spearmana	95
Współczynnik korelacji rang Kendalla	95
Centralne twierdzenie graniczne	96
Pozyskiwanie prób	96
Przeprowadzanie testów parametrycznych	98
Przeprowadzanie testów nieparametrycznych	102
Podsumowanie	107
Rozdział 4. Algebra liniowa	108
<hr/>	
Wymogi techniczne	109
Dopasowywanie do wielomianów za pomocą biblioteki NumPy	109
Wyznacznik macierzy	111
Określanie rzędu macierzy	111
Macierz odwrotna w bibliotece NumPy	112
Rozwiązywanie równań liniowych za pomocą biblioteki NumPy	113
Rozkład macierzy za pomocą SVD	114
Wartości własne i wektory własne w bibliotece NumPy	115
Generowanie liczb losowych	116
Rozkład dwumianowy	117
Rozkład normalny	118
Testowanie normalności rozkładu danych za pomocą biblioteki SciPy	119
Tworzenie tablicy maskowanej za pomocą podpakietu numpy.ma	122
Podsumowanie	124

Część II.

Eksploracyjna analiza danych i oczyszczanie danych

Rozdział 5. Wizualizacja danych	127
<hr/>	
Wymogi techniczne	127
Wizualizacja za pomocą pakietu Matplotlib	128
Akcesoria wykresu	129
Wykres punktowy	131
Wykres liniowy	132
Wykres kołowy	133
Wykres kolumnowy	134
Histogram	135
Wykres bąbelkowy	136
Tworzenie wykresów za pomocą biblioteki pandas	137
Zaawansowana wizualizacja za pomocą pakietu seaborn	139
Wykresy lm	140
Wykresy kolumnowe	142
Wykresy rozkładu	143
Wykresy pudełkowe	143
Wykresy KDE	144
Wykresy skrzypcowe	145

Wykresy zliczeń	146
Wykresy łączone	147
Mapy cieplne	148
Wykresy macierzowe	150
Wizualizacja interaktywna za pomocą biblioteki Bokeh	151
Tworzenie prostego wykresu	151
Glify	153
Szablony	154
Wykresy wielokrotne	157
Oddziaływania	159
Adnotacje	162
Najeżdżanie kursorem	163
Widżety	165
Podsumowanie	168
Rozdział 6. Pozyskiwanie, przetwarzanie i przechowywanie danych	169
Wymogi techniczne	170
Odczyt i zapis plików CSV za pomocą biblioteki NumPy	171
Odczyt i zapis plików CSV za pomocą biblioteki pandas	172
Odczyt i zapis plików arkusza kalkulacyjnego Excel	173
Odczyt i zapis plików JSON	174
Odczyt i zapis plików HDF5	175
Odczyt i zapis danych z tabel HTML-a	176
Odczyt i zapis plików Parquet	177
Odczyt i zapis danych z obiektu pickle	178
Łatwy dostęp do danych za pomocą modułu sqlalchemy	178
Odczyt i zapis danych w bazie danych MySQL	180
Wstawianie całego obiektu DataFrame do bazy danych	182
Odczyt i zapis danych w bazie danych MongoDB	183
Odczyt i zapis danych w bazie danych Cassandra	184
Odczyt i zapis danych w bazie danych Redis	185
PonyORM	186
Podsumowanie	187
Rozdział 7. Oczyszczanie nieuporządkowanych danych	188
Wymogi techniczne	189
Eksploracja danych	189
Filtrowanie danych w celu pozbycia się szumu	192
Filtrowanie po kolumnach	193
Filtrowanie po rzędach	195
Rozwiązywanie kwestii brakujących wartości	197
Usuwanie brakujących wartości	197
Uzupełnianie brakującej wartości	198
Rozwiązywanie kwestii elementów odstających	200
Techniki kodowania cech	202
Kodowanie „gorącojedynkowe”	202
Kodowanie etykietowe	204
Koder zmiennych porządkowych	205

Skalowanie cech	206
Metody skalowania cech	206
Przekształcanie cech	208
Rozdzielanie cech	210
Podsumowanie	211
Rozdział 8. Przetwarzanie sygnałów i szeregi czasowe	212
Wymogi techniczne	213
Moduł statsmodels	213
Średnie kroczące	213
Funkcje okna czasowego	216
Kointegracja	217
Rozkład STL	220
Autokorelacja	221
Modele autoregresyjne	223
Model ARMA	225
Generowanie sygnałów okresowych	227
Analiza Fouriera	230
Filtrowanie metodą analizy widmowej	231
Podsumowanie	233

Część III. Dokładna analiza uczenia maszynowego

Rozdział 9. Uczenie nadzorowane: analiza regresyjna	237
Wymogi techniczne	238
Regresja liniowa	238
Wieloraka regresja liniowa	239
Wielospółliniowość	240
Usuwanie wielospółliniowości	240
Zmienne fikcyjne	242
Projektowanie modelu regresji liniowej	243
Ocenianie skuteczności modelu regresyjnego	245
Współczynnik determinacji R-kwadrat	245
MSE	246
MAE	246
RMSE	246
Dopasowywanie regresji wielomianowej	247
Modele regresji używane w klasyfikacji	249
Regresja logistyczna	250
Charakterystyka modelu regresji logistycznej	251
Rodzaje algorytmów regresji logistycznej	252
Mocne i słabe strony regresji logistycznej	252
Implementacja regresji logistycznej za pomocą biblioteki scikit-learn	252
Podsumowanie	254

Rozdział 10. Uczenie nadzorowane: techniki klasyfikacji	255
Wymogi techniczne	256
Klasyfikacja	256
Naiwny klasyfikator Bayesa	258
Drzewa decyzyjne	261
Algorytm KNN	264
Maszyny wektorów nośnych	266
Terminologia	266
Podział danych na zestawy uczący i testowy	268
Wydzielanie	269
k-krotny sprawdzian krzyżowy	269
Metoda samowsporna	269
Ocena skuteczności modelu klasyfikacji	270
Macierz pomyłek	270
Dokładność	273
Precyzja	273
Czułość	273
Wskaźnik F1	273
Krzywa ROC i obszar AUC	274
Podsumowanie	276
Rozdział 11. Uczenie nienadzorowane: PCA i analiza skupień	277
Wymogi techniczne	278
Uczenie nienadzorowane	278
Redukowanie wymiarowości danych	279
Analiza głównych składowych	280
Przeprowadzanie PCA	280
Analiza skupień	283
Wyznaczanie liczby skupień	284
Grupowanie danych za pomocą algorytmu centroidów	288
Hierarchiczna analiza skupień	290
Algorytm DBSCAN	294
Widmowa analiza skupień	296
Ocenianie jakości analizy skupień	298
Wewnętrzna ocena jakości	298
Zewnętrzna ocena jakości	299
Podsumowanie	303

Część IV. Przetwarzanie języka naturalnego, analiza obrazów i obliczenia równoległe

Rozdział 12. Analiza danych tekstowych	307
Wymogi techniczne	308
Instalacja bibliotek NLTK i spaCy	308
Normalizacja tekstu	309
Tokenizacja	310

Usuwanie słów nieinformatywnych	314
Rdzeniowanie słów i lematyzacja	315
Oznaczanie części mowy	317
Rozpoznawanie jednostek nazewniczych	318
Analiza zależności	319
Tworzenie chmury słów	320
„Worek słów”	321
Metoda TF-IDF	322
Analiza sentymentów za pomocą klasyfikacji tekstu	323
Klasyfikacja za pomocą „worka słów”	324
Klasyfikacja za pomocą metody TF-IDF	328
Podobieństwo tekstów	330
Indeks Jaccarda	331
Podobieństwo cosinusowe	332
Podsumowanie	333
Rozdział 13. Analiza obrazów	334
Wymogi techniczne	335
Instalacja biblioteki OpenCV	335
Omówienie danych obrazowych	336
Obrazy binarne	336
Obrazy w odcieniach szarości	337
Obrazy kolorowe	337
Modele barw	338
Rysowanie na obrazach	341
Pisanie na obrazach	345
Zmiana rozmiaru obrazu	346
Przekształcenie izometryczne obrazów	348
Zmiana jasności	350
Rozmywanie obrazu	352
Wykrywanie twarzy	355
Podsumowanie	358
Rozdział 14. Obliczenia równoległe za pomocą biblioteki Dask	359
Obliczenia równoległe za pomocą biblioteki Dask	360
Typy danych Dask	361
Tablice Dask	362
Ramki danych Dask	363
Worki Dask	368
Interfejs Dask Delayed	371
Skalowane wstępne przetwarzanie danych	373
Skalowanie cech w bibliotece Dask	373
Kodowanie cech w bibliotece Dask	374
Skalowane uczenie maszynowe	376
Obliczenia równoległe za pomocą biblioteki scikit-learn	377
Reimplementacja algorytmów uczenia maszynowego na potrzeby biblioteki Dask	378
Podsumowanie	382

Wprowadzenie do bibliotek Pythona

Jak już wiesz, Python stał się jednym z najpopularniejszych języków standardowych i zawiera wszystkie narzędzia niezbędne do przeprowadzania obliczeń naukowych. W jego ramach dostępne są liczne biblioteki, takie jak NumPy, pandas, SciPy, scikit-learn, Matplotlib, seaborn czy Plotly. Tworzą one kompletne środowisko analizy danych, wykorzystywane przez analityków danych, **danetyków**¹ (ang. *data scientist*) i analityków biznesowych. Pythona wyróżniają także inne cechy. Jest m.in.: elastyczny, przystępny, dynamicznie rozwijany, zawiera olbrzymią społeczność użytkowników, a także umożliwia pracę nad skomplikowanymi zadaniami numerycznymi, naukowymi i badawczymi. Z powodu tych cech stanowi pierwszy wybór w przypadku analizy danych.

W tym rozdziale skoncentrujemy się na różnych procesach analizy danych, takich jak KDD, SEMMA czy CRISP-DM. Następnie zajmiemy się porównaniem analizy danych i danetyki, a także rolami i umiejętnościami cechującymi dobrego analityka danych oraz danetyka. Na koniec zmienimy tematykę i pokażemy sposób instalacji różnych bibliotek Pythona, a także aplikacji IPython, JupyterLab i Jupyter Notebook. Przyjrzymy się także zaawansowanym funkcjom tej ostatniej aplikacji.

Rozdział ten jest poświęcony następującym zagadnieniom:

- wyjaśnienie pojęcia „analiza danych”,
- standardowy proces analizy danych,
- proces KDD,
- proces SEMMA,
- proces CRISP-DM,

¹ Etymologię słowa **danetyka** (ang. *data science*) znajdziesz pod adresem <https://www.kodolamacz.pl/blog/data-science-po-polsku/> — *przyj. tłum.*

- analiza danych a danetyka,
- umiejętności analityka danych oraz danetyka,
- instalacja środowiska Python 3,
- oprogramowanie używane w tej książce,
- używanie aplikacji IPython jako powłoki,
- korzystanie z aplikacji JupyterLab,
- stosowanie aplikacji Jupyter Notebook,
- zaawansowane funkcje aplikacji Jupyter Notebook.

Zaczynamy!

Wyjaśnienie pojęcia „analiza danych”

Wiek XXI jest wiekiem informacji. Żyjemy w erze informacji, co oznacza, że niemal każdy aspekt naszego życia wytwarza jakieś dane. Ponadto olbrzymie ilości danych są generowane przez operacje biznesowe, działania rządów oraz wpisy w serwisach społecznościowych. Codziennie gromadzonych jest coraz więcej danych, gdyż bez przerwy powstają w wyniku działań biznesowych, rządowych, naukowych, inżynierskich, zdrowotnych, społecznych, klimatycznych i środowiskowych. W tych wszystkich dziedzinach wymagających podejmowania decyzji potrzebny jest usystematyzowany, uogólniony, skuteczny i elastyczny system realizujący procesy analityczne i naukowe, dzięki którym możemy wyciągać wnioski z generowanych danych.

We współczesnym, inteligentnym świecie analiza danych zapewnia skuteczny proces decyzyjny w zastosowaniach biznesowych i rządowych. Analiza danych jest procesem sprawdzania, przetwarzania, przeglądania, opisywania i wizualizowania dostarczanych zestawów danych. Głównym celem procesu analizy danych jest odkrywanie informacji niezbędnych do podejmowania decyzji. Na analizę danych składa się wiele różnych metod, narzędzi i technik, które można stosować w tak rozbieżnych dziedzinach jak biznes, socjologia czy nauki fundamentalne.

Przyjrzyjmy się podstawowym bibliotekom analizy danych dostępnymi w środowisku Pythona:

- **NumPy** — nazwę tę możemy rozwinąć jako „numeryczny Python”. Jest to najbardziej zaawansowana biblioteka naukowa Pythona, obsługująca wielowymiarowe tablice i macierze oraz zawierająca metody umożliwiające przeprowadzanie wydajnych obliczeń matematycznych.
- **SciPy** — jest to również zaawansowana biblioteka obliczeń naukowych realizująca operacje naukowe, matematyczne i inżynierskie.
- **pandas** — biblioteka służąca do eksplorowania danych i manipulowania nimi, zawierająca tabelaryczne struktury danych, takie jak obiekty DataFrame, a także różne metody analizowania i przetwarzania danych.

- **scikit-learn** — nazwa ta oznacza *Scientific Toolkit for Machine learning*, czyli „zestaw narzędzi naukowych przeznaczonych do uczenia maszynowego”. Jest to biblioteka uczenia maszynowego, zawierająca wiele algorytmów uczenia nadzorowanego i nienadzorowanego, takich jak algorytmy regresji, klasyfikacji, redukcji wymiarowości, analizy skupień i wykrywania anomalii.
- **Matplotlib** — jest to kluczowa biblioteka wizualizacji danych, a także podstawowa biblioteka dla wszystkich pozostałych bibliotek wizualizacji w Pythonie. Umożliwia generowanie wykresów dwu- i trójwymiarowych, a także rysunków służących do wzrokowej analizy danych. Współpracuje z bibliotekami NumPy oraz SciPy.
- **seaborn** — biblioteka ta bazuje na bibliotece Matplotlib i pozwala na łatwe tworzenie dokładniejszych, interaktywnych i bardziej zorganizowanych wykresów.
- **Plotly** — jest to biblioteka wizualizacji danych. Zawiera wiele typów dobrej jakości wykresów, takich jak wykresy punktowe, liniowe, słupkowe, pudełkowe, histogramy, mapy cieplne i wykresy podrzędne.

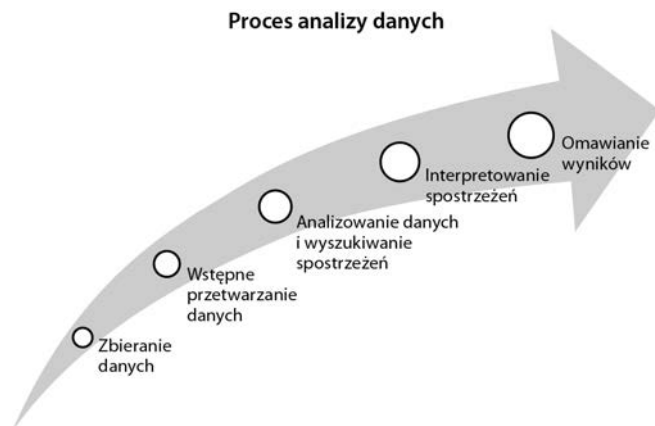
Tam, gdzie będzie konieczna instalacja poszczególnych bibliotek, opiszemy instrukcję ich instalacji krok po kroku. Tymczasem zajmijmy się omówieniem różnych procesów analizy danych, takich jak standardowy proces, KDD, SEMMA i CRISP-DM.

Standardowy proces analizy danych

Proces analizy danych polega na badaniu danych, uzyskiwaniu na ich podstawie spostrzeżeń i wyciąganiu z nich wniosków. Głównym celem tego procesu jest zbieranie, filtrowanie, oczyszczanie, przekształcanie, przeglądanie, opisywanie i wizualizowanie danych oraz przekazywanie tych spostrzeżeń w celu odkrywania informacji pomagających w podejmowaniu decyzji. Zasadniczo proces analizowania danych składa się z następujących etapów:

1. **Zbieranie danych** — pozyskiwanie i gromadzenie danych z różnych źródeł.
2. **Wstępne przetwarzanie danych** — filtrowanie, oczyszczanie i przekształcanie danych w wymagany format.
3. **Analizowanie danych i wyszukiwanie spostrzeżeń** — przeglądanie, opisywanie i wizualizowanie danych, a także znajdowanie spostrzeżeń i wyciąganie na ich podstawie wniosków.
4. **Intepretowanie spostrzeżeń** — zrozumienie znaczenia spostrzeżeń i określenie wpływu każdej zmiennej na dany system.
5. **Omawianie wyników (ang. *storytelling*)** — przekazywanie rezultatów w formie zrozumiałej dla laika historyjki.

Możemy podsumować przebieg poszczególnych etapów procesu analizy danych za pomocą następującego diagramu:



W tym podrozdziale przyjrzelśmy się standardowemu procesowi analizy danych, w którym główny nacisk jest kładziony na wyszukiwanie łatwych do interpretacji spostrzeżeń i przekształcanie ich w historijkę użytkownika. W następnym podrozdziale przejdziemy do procesu KDD.

Proces KDD

Skrót KDD rozwijamy jako **knowledge discovery from data** (czyli „odkrywanie wiedzy z danych”) lub **Knowledge Discovery in Databases** („odkrywanie wiedzy w bazach danych”). Dla wielu osób proces KDD jest synonimem wydobywania danych (ang. *data mining*). Wydobywaniem danych nazywamy proces odkrywania wiedzy z interesujących wzorów danych. Głównym celem procesu KDD jest wydobywanie lub odkrywanie ukrytych, interesujących wzorów z dużych baz danych, hurtowni danych oraz innych repozytoriów informacji. Proces KDD składa się z siedmiu głównych etapów:

1. **Oczyszczanie danych** — w tej pierwszej fazie realizowane jest wstępne przetwarzanie danych. Tu właśnie jest usuwany szum, rozwiązywany problem brakujących wartości, a także wykrywane są elementy odstające.
2. **Integracja danych** — na tym etapie dane pochodzące z różnych źródeł są łączone oraz integrowane za pomocą narzędzi migracji danych i ETL.
3. **Dobór danych** — wyznaczone są tutaj dane istotne z perspektywy analizy.
4. **Przekształcanie danych** — dane są tutaj przygotowywane do analizy.
5. **Wydobywanie danych** — używane są tu techniki wydobywania danych w celu odkrywania przydatnych i nieznanych wzorów.
6. **Ocena wzorów** — w tej fazie oceniane są wydobyte wzory.
7. **Prezentacja wiedzy** — po przeprowadzeniu oceny wzorów należy zwizualizować odkrytą wiedzę i zaprezentować ją właściwym osobom odpowiedzialnym za podejmowanie decyzji.

Cały proces KDD został zaprezentowany na poniższym schemacie:



KDD jest procesem iteracyjnym służącym poprawie jakości danych, a także ich integracji i przekształcaniu w celu usprawnienia systemu. Przejdźmy do procesu SEMMA.

Proces SEMMA

Skrót SEMMA rozwijamy jako **S**ample, **E**xplore, **M**odify, **M**odel, **A**ssess (czyli „próbuj, eksploruj, modyfikuj, modeluj, oceniaj”). Ten sekwencyjny proces wydobywania danych został opracowany przez instytut SAS. Jest on pięciostopowy:

1. **Próbkowanie** — w tej fazie rozpoznajemy różne bazy danych i scalamy je. Następnie możemy wybrać próbę danych wystarczającą do procesu modelowania.
2. **Eksploracja** — na tym etapie staramy się zrozumieć dane, odkrywamy relacje pomiędzy zmiennymi, wizualizujemy dane i uzyskujemy wstępną interpretację danych.
3. **Modyfikacja** — tutaj przygotowujemy dane do modelowania. W tej fazie rozwiązujemy kwestię brakujących wartości, wykrywamy elementy odstające, przekształcamy cechy i tworzymy dodatkowe cechy.
4. **Modelowanie** — głównym zadaniem na tym etapie jest dobór i stosowanie różnych technik modelowania, takich jak regresja liniowa lub logistyczna, sieci przeprowadzające propagację wsteczną, algorytm k-najbliższych sąsiadów, maszyny wektorów nośnych, drzewa decyzyjne czy lasy losowe.
5. **Ocena** — w ostatniej fazie oceniane są uzyskane modele predykcyjne za pomocą miar oceny wydajności.

Proces ten został ukazany na następującym schemacie:



Powyższy schemat prezentuje kolejne etapy procesu SEMMA. Kluczowymi etapami są tutaj tworzenie modelu i jego ocena. Spójrzmy teraz na proces CRISP-DM.

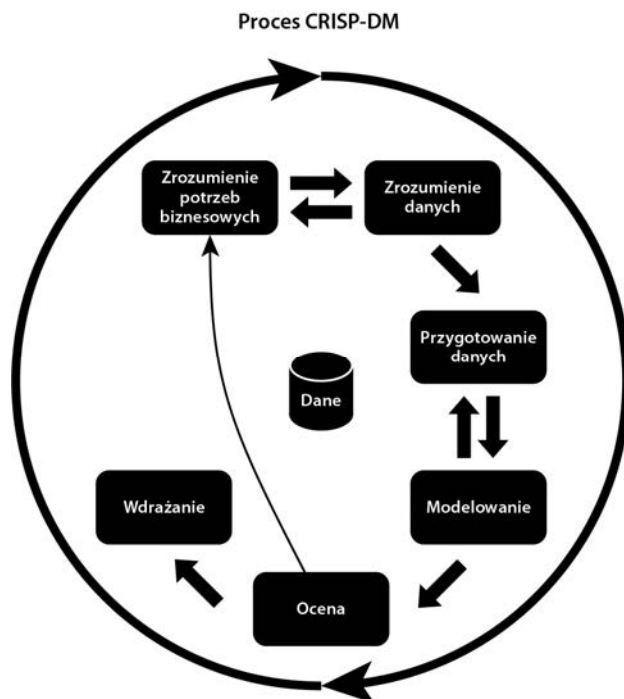
Proces CRISP-DM

Skrót CRISP-DM rozwijamy jako **C**ross-**I**ndu**S**try **P**rocess for **D**ata **M**ining (czyli „międzybranżowy proces wydobywania danych”). Jest to precyzyjnie zdefiniowany, ustrukturyzowany i skuteczny proces stosowany w projektach uczenia maszynowego, wydobywania danych i analizy biznesowej. Stanowi on solidną, elastyczną, cykliczną, przydatną i praktyczną metodologię

rozwiązywania problemów biznesowych. Proces ten odkrywa niewidoczne, a przydatne informacje lub wzory z kilku baz danych. Na CRISP-DM składa się sześć głównych etapów:

- 1. Zrozumienie potrzeb biznesowych** — głównym celem w tej pierwszej fazie jest zdefiniowanie zadania biznesowego i wymagań, aby można było zaprojektować cel analityczny i wstępny plan działania.
- 2. Zrozumienie danych** — tutaj głównymi zadaniami są zrozumienie danych i procesu ich pozyskiwania, testowanie jakości danych, a także uzyskiwanie wstępnych spostrzeżeń.
- 3. Przygotowanie danych** — ten etap polega na przygotowaniu danych do analizy. Wiąże się z tym rozwiązanie kwestii brakujących wartości, wykrywanie i odpowiednie przetwarzanie elementów odstających, normalizowanie danych, a także inżynieria cech. Faza ta jest najbardziej czasochłonna dla analityków danych/danetyków.
- 4. Modelowanie** — jest to najciekawsza część całego procesu, ponieważ to właśnie tutaj projektujemy model predykcyjny. Analityk musi najpierw wybrać technikę modelowania, a następnie opracować modele na podstawie danych.
- 5. Ocena** — po przygotowaniu modelu należy ocenić i przetestować jego skuteczność na danych walidacyjnych i testowych za pomocą miar oceny modelu takich jak MSE, RMSE, R-kwadrat w przypadku regresji, a także wskaźników dokładności, precyzji, pełności i F1.
- 6. Wdrażanie** — w tej ostatniej fazie model, który został wybrany na poprzednim etapie, zostanie wdrożony do środowiska produkcyjnego. Wymaga to pracy zespołowej danetyków, programistów, ekspertów DevOps i zawodowych biznesmenów.

Na poniższym schemacie widzimy pełen cykl procesu CRISP-DM:



Standardowy proces kładzie nacisk na odkrywanie spostrzeżeń i przygotowywanie ich interpretacji w formie historyjek, natomiast proces KDD koncentruje się na odkrywaniu wzorów danych i ich wizualizowaniu. W procesie SEMMA skupiamy się na zadaniach tworzenia modelu, natomiast w procesie CRISP-DM najważniejsze są zrozumienie potrzeb biznesowych oraz wdrażanie. Wiemy już co nieco o procesach powiązanych z analizą danych, możemy więc teraz porównać analizę danych z danetyką, by określić występujące między nimi relacje i definiujące je różnice.

Analiza danych a danetyka

Analiza danych (ang. *data analysis*) jest procesem poznawania danych w celu odkrycia wzorów pomagających w podejmowaniu decyzji biznesowych. Stanowi ona jedną z dziedzin **danetyki** (ang. *data science*). Metody i narzędzia analizy danych są powszechnie stosowane w różnych kategoriach biznesowych przez analityków biznesowych, analityków danych i badaczy. Jej głównym celem jest zwiększanie produktywności i zysków. Analityk danych wydobywa i sprawdza dane z różnych źródeł, realizuje eksploracyjną analizę danych, wizualizuje dane, przygotowuje raporty i prezentuje je kadrom kierowniczym.

Z kolei danetyka stanowi interdyscyplinarną dziedzinę wykorzystującą metody naukowe do wydobywania spostrzeżeń z danych ustrukturyzowanych i nieustrukturyzowanych. Składa się na nią wiele dziedzin, takich jak analiza danych, wydobywanie danych, uczenie maszynowe itp. Danetyka nie ogranicza się jedynie do eksploracyjnej analizy danych i jest stosowana w tworzeniu modeli i algorytmów predykcyjnych w takich kategoriach jak prognozy cen akcji, pogody, epidemii czy oszustw, a także rekomendacje filmów, książek czy muzyki.

Role analityka danych i danetyka

Analityk danych zbiera, filtruje, przetwarza i wykorzystuje wymagane pojęcia statystyczne do wyłapywania wzorów, trendów i spostrzeżeń z danych, a także przygotowuje raporty przeznaczone dla decydentów. Głównym zadaniem analityka danych jest pomaganie firmom w rozwiązywaniu problemów biznesowych za pomocą odkrytych wzorów i trendów. Analityk danych ocenia także jakość danych i rozwiązuje problemy związane z ich pozyskiwaniem. Osoba taka powinna być biegła w tworzeniu kwerend SQL-a, wyszukiwaniu wzorów, korzystaniu z narzędzi wizualizacji, a także stosowaniu narzędzi raportujących, takich jak Microsoft Power BI, IBM Cognos, Tableau, QlikView, Oracle BI itd.

Danetycy są bardziej zorientowani matematycznie i technicznie od analityków danych. Są raczej nastawieni na kwestie badawcze i akademickie, natomiast analitycy danych są ukierunkowani użytkowo. Po danetyku oczekuje się, że będzie przewidywał przyszłe zdarzenia, z kolei analityk danych ma wydobywać przydatne spostrzeżenia z danych. Danetycy sami określają pytania, na jakie chcą odpowiedzieć, podczas gdy analitycy znajdują odpowiedzi na zadane pytania. Ostatnia różnica jest taka, że danetycy koncentrują się na tym, **co się wydarzy**, a analitycy

danych skupiają się na tym, **co już się wydarzyło**. Możemy podsumować obydwie role za pomocą poniższej tabeli:

Cechy	Danetyk	Analitik danych
Zadanie	Przewidywanie przyszłych zdarzeń i scenariuszy na podstawie danych	Odkrywanie przydatnych spostrzeżeń w danych
Rola	Formułowanie pytań, które mogą być korzystne dla firmy	Uzyskiwanie odpowiedzi na zadane pytania w celu podejmowania decyzji
Typ danych	Pracuje zarówno na danych ustrukturyzowanych, jak i nieustrukturyzowanych	Pracuje jedynie na danych ustrukturyzowanych
Programowanie	Zaawansowane	Podstawowe
Umiejętności	Znajomość statystyki, algorytmów uczenia maszynowego, przetwarzania języka naturalnego i uczenia głębokiego	Znajomość statystyki, baz danych SQL i wizualizacji danych
Narzędzia	R, Python, SAS, Hadoop, Spark, TensorFlow i Keras	Excel, SQL, R, Tableau i QlikView

Poznaliśmy definicję analityka danych i danetyka, a także występujące między nimi różnice, możemy więc przyjrzeć się teraz różnym umiejętnościom wymaganim do tego, aby zostać jednym z nich.

Umiejętności analityka danych oraz danetyka

Analityk danych odkrywa spostrzeżenia w danych i tworzy z nich wartość. Dzięki temu osoby podejmujące decyzje poznają stan firmy. Analityk danych musi posiadać następujące umiejętności:

- **Eksploracyjna analiza danych (ang. *Exploratory Data Analysis* — EDA)** — EDA jest kluczową umiejętnością analityka danych. Pomaga ona podczas wyszukiwania wzorów w danych, testowania hipotez i potwierdzania założeń.
- **Relacyjne bazy danych** — obowiązkowa jest znajomość przynajmniej jednego narzędzia relacyjnych baz danych, takiego jak MySQL czy Postgre. Niezbędna jest znajomość języka SQL podczas pracy na relacyjnych bazach danych.
- **Narzędzia wizualizacji i analityki biznesowej** — obraz mówi więcej niż tysiąc słów. Obrazy mają olbrzymi wpływ na ludzi i za ich pomocą można łatwo oraz precyzyjnie przekazywać spostrzeżenia. Takie narzędzia jak Tableau, QlikView, MS Power BI czy IBM Cognos pomagają analitykom wizualizować dane i przygotowywać raporty.
- **Arkusze kalkulacyjny** — znajomość aplikacji MS Excel, WPS, Libra lub Google Sheets jest wymagana do przechowywania danych i zarządzania nimi w formie tabelarycznej.

- **Umiejętność opowiadania historyjek i przygotowywania prezentacji** — sztuka opowiadania historyjek jest kolejną obowiązkową umiejętnością. Analityk danych powinien potrafić po mistrzowsku łączyć fakty z ideą lub zdarzeniem i przekształcać je w historyjki.

Z kolei głównym zadaniem danetyka jest rozwiązywanie problemów za pomocą danych. W tym celu musi on znać potrzeby klienta, badany obszar, przestrzeń problemów i upewnić się, że klient otrzyma dokładnie to, czego rzeczywiście potrzebuje. Zadania realizowane przez danetyka różnią się w poszczególnych firmach. W niektórych firmach przydaje się analityk danych, a stanowisko danetyka jest tworzone jedynie na pokaz. Pewne firmy pod nazwą tego stanowiska łączą zadania analityka danych z zadaniami inżyniera danych; w innych natomiast przydzielają osoby na tym stanowisku do zadań uczenia maszynowego i wizualizacji danych.

Zadania danetyka różnią się pomiędzy firmami. Niektóre przedsiębiorstwa zatrudniają danetyków jako sławnych analityków danych i łączą ich obowiązki z obowiązkami inżynierów danych. W innych firmach zajmują się oni skomplikowanymi wizualizacjami danych.

Danetyk musi być złotą rączką i umieć dopasowywać się do różnych ról, również analityka danych, statystyka, matematyka, programisty, inżyniera uczenia maszynowego lub przetwarzania języka naturalnego. Niewiele jest osób, które są ekspertami lub mają wystarczające umiejętności w każdej z tych ról. Poza tym wyrobienie umiejętności na wystarczającym poziomie wymaga mnóstwa wysiłku i cierpliwości. Z tego właśnie powodu nie można zostać danetykiem w trzy miesiące czy pół roku. Uczenie się danetyki jest podróżą samą w sobie. Danetyk musi posiadać wszechstronny szereg umiejętności, w tym takie jak:

- **Matematyka i statystyka** — większość algorytmów uczenia maszynowego ma podstawy w matematyce i statystyce. Znajomość matematyki pomaga danetykom tworzyć nieszablonowe rozwiązania.
- **Bazy danych** — znajomość języka SQL pozwala danetykom oddziaływać z bazami danych i gromadzić dane używane w predykcji i rekomendacjach.
- **Uczenie maszynowe** — znajomość technik nadzorowanego uczenia maszynowego, takich jak analiza regresyjna, techniki klasyfikacji, a także technik nienadzorowanego uczenia maszynowego, np.: analiza skupień, wykrywanie elementów odstających czy redukcja wymiarowości.
- **Umiejętności programistyczne** — umiejętność programowania pomaga danetykom w automatyzacji sugerowanych rozwiązań. Zalecana jest znajomość języków Python i R.
- **Umiejętność opowiadania historyjek i przygotowywania prezentacji** — przekazywanie wyników w postaci historyjek za pomocą prezentacji PowerPointa.
- **Technologia danych wielkoskalowych (ang. *Big Data*)** — znajomość platform danych wielkoskalowych, takich jak Hadoop czy Spark, pomaga danetykom projektować rozwiązania dla bardzo dużych firm.
- **Narzędzia uczenia głębokiego** — narzędzia uczenia głębokiego, takie jak biblioteki TensorFlow czy Keras, stosowane są w zadaniach przetwarzania języka naturalnego i przetwarzania obrazów.

Oprócz tych umiejętności przydatna jest także znajomość pakietów/narzędzi do ekstrakcji danych z różnych źródeł internetowych, a także struktur aplikacji sieciowych, takich jak Flask czy Django, do projektowania prototypowych rozwiązań. Umiejętności te cechują najlepszych danetyków.

Znamy już cechy analizy danych i danetyki, możemy więc przejść do podstawowej konfiguracji wymaganej do rozpoczęcia przygody z analizą danych. W następnym podrozdziale omawiamy proces instalacji Pythona.

Instalacja środowiska Python 3

Plik instalacyjny środowiska Python 3 można bez trudu pobrać z oficjalnej strony projektu (<https://www.python.org/downloads/>) na systemy Windows, Linux i macOS w wersjach 32- i 64-bitowej. Po pobraniu instalatora wystarczy kliknąć go dwukrotnie. Dostępne jest również środowisko programistyczne IDE o nazwie *IDLE*, które można wykorzystać w celach edukacyjnych. W kolejnych punktach zajmiemy się opisem instalacji Pythona w poszczególnych systemach operacyjnych.

Instalacja i konfigurowanie Pythona w systemie Windows

Opisywane w dalszej części książki informacje bazują na środowisku Python 3. Wszystkie omawiane listingi zostały napisane w tej wersji środowiska, dlatego należy ją zainstalować, aby móc zabrać się za pisanie kodu. Python jest otwartym, łatwo dostępnym i bezpłatnym językiem programowania. Jest także objęty licencją umożliwiającą korzystanie w celach komercyjnych. Istnieje wiele implementacji Pythona, w tym również implementacje i dystrybucje komercyjne. W tej książce koncentrujemy się na standardowej implementacji Pythona, która jest kompatybilna z biblioteką NumPy.

Wersja 3.9.x Pythona jest dostępna do pobrania na stronie <https://www.python.org/downloads/>. Znajdziesz tu pliki instalacyjne dla systemów Windows, Linux, macOS i innych. Pod adresem <https://docs.python.org/3.9/using/index.html> znajdziesz instrukcje (po angielsku) instalacji i korzystania z Pythona w różnych systemach operacyjnych.

Upewnij się, że masz zainstalowany Python w wersji co najmniej 3.5.x. Termin zakończenia wsparcia wersji 2.7 został przesunięty z 2015 r. na 2020 r. i nie należy już jej instalować.

W naszej wirtualnej maszynie z systemem Windows 10 zainstalowaliśmy środowisko Python 3.8.3 (<https://www.python.org/ftp/python/3.8.3/python-3.8.3.exe>).

Instalacja i konfigurowanie Pythona w Linuksie

Instalacja Pythona w Linuksie jest zdecydowanie łatwiejsza niż w innych systemach operacyjnych. Aby zainstalować najważniejsze biblioteki, wpisz następujące polecenie w wierszu poleceń:

```
$ pip3 install numpy scipy pandas matplotlib jupyter notebook
```

Jeżeli nie masz wystarczających uprawnień na danym komputerze, być może będziesz musiał najpierw skorzystać z polecenia `sudo`.

Instalacja i konfigurowanie Pythona w systemie macOS za pomocą instalatora graficznego

Środowisko Python można zainstalować za pomocą pliku instalacyjnego dostępnego na oficjalnej stronie projektu Python. W przypadku systemu macOS adres pliku instalacyjnego jest następujący: <https://www.python.org/downloads/mac-osx/>. Znajdziesz tu także środowisko IDE o nazwie *IDLE*, służące do celów edukacyjnych.

Instalacja i konfigurowanie Pythona w systemie macOS za pomocą menedżera Homebrew

W przypadku systemów z rodziny macOS możesz zainstalować Pythona z poziomu menedżera pakietów Homebrew. W ten sposób programiści, badacze i naukowcy mogą w łatwy sposób instalować wymagane aplikacje. Polecenie `brew install` służy do instalacji zarówno aplikacji, np. Python 3, jak również dodatkowych pakietów Pythona, takich jak NLTK czy spaCy.

Aby zainstalować bieżącą wersję Pythona, wpisz w Terminalu następujące polecenie:

```
$ brew install python3
```

Po zainstalowaniu Pythona możesz sprawdzić jego wersję w następujący sposób:

```
$ python3 --version
Python 3.7.4
```

Możesz także otworzyć powłokę Pythona z poziomu wiersza poleceń za pomocą poniższego polecenia:

```
$ python3
```

Skoro umiemy już zainstalować Pythona w systemie operacyjnym, zajmijmy się teraz właściwymi narzędziami używanymi w analizie danych.

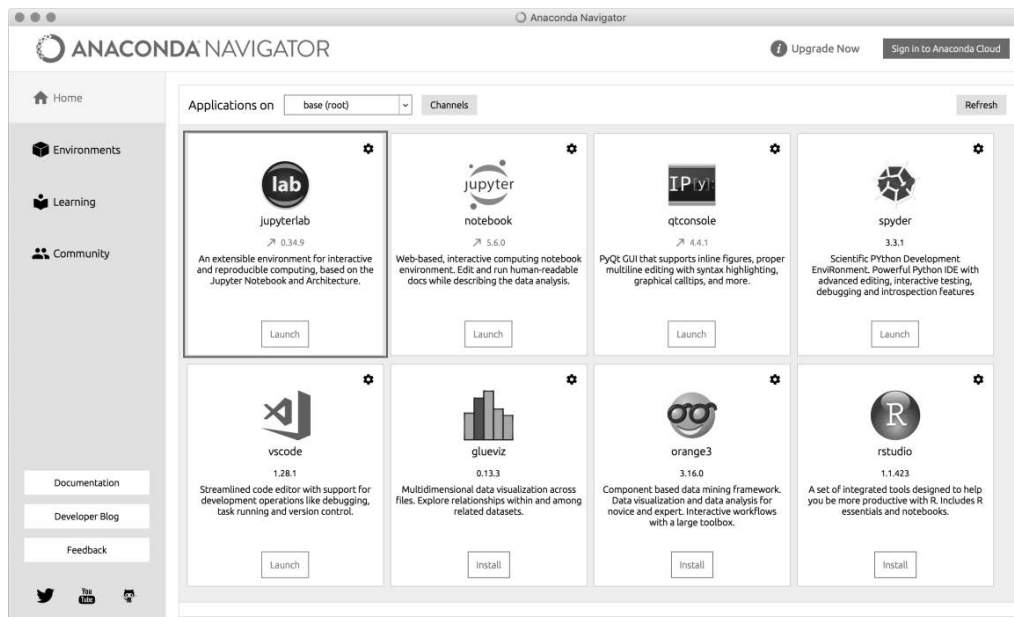
Oprogramowanie używane w tej książce

Przyjrzyjmy się oprogramowaniu używanemu w tej książce. Będziemy posługiwać się tu środowiskiem IDE do analizowania danych. Zanim jednak je zainstalujemy, wyjaśnimy dokładniej, czym ono jest.

Program napisany w Pythonie można bez problemu uruchomić w systemie mającym zainstalowany ten język. Możemy napisać program w Notatniku i uruchomić go z poziomu wiersza poleceń. Możemy także pisać i uruchamiać programy pythonowe w różnych środowiskach IDE, takich jak Jupyter Notebook, Spyder czy PyCharm. Anaconda jest bezpłatnym, otwartym pakietem zawierającym różnorodne środowiska IDE umożliwiające manipulowanie danymi, a także kilka bibliotek przeznaczonych do analizy danych, np.: NumPy, SciPy, pandas, scikit-learn. Pobranie i instalacja Anacondy nie nastroczają większych problemów:

1. Pobierz instalator ze strony <https://www.anaconda.com/products/individual>.
2. Wybierz używany system operacyjny.
3. Z sekcji Python 3.8 wybierz wersję 32- lub 64-bitową i pobierz ją.
4. Kliknij dwukrotnie pobrany plik, aby rozpocząć proces instalacji.
5. Po zakończeniu instalacji możesz poszukać swojego programu w menu *Start* lub uruchomić Anacondę.

Anaconda zawiera również aplikację graficzną Anaconda Navigator, z której poziomu możemy uruchamiać takie programy jak Jupyter Notebook, Spyder, RStudio, Visual Studio Code czy JupyterLab:



Przyjrzyjmy się teraz powłokowemu środowisku obliczeniowemu IPython, które jest używane w analizie danych.

Używanie aplikacji IPython jako powłoki

IPython jest interaktywną powłoką stanowiącą odpowiednik interaktywnego środowiska obliczeniowego, takiego jak Matlab czy Mathematica. Powłoka ta została utworzona z myślą o przeprowadzaniu szybkich eksperymentów. Stanowi bardzo przydatne narzędzie w rękach profesjonalistów wykonujących drobne eksperymenty.

Powłoka IPython ma następujące cechy:

- łatwy dostęp do poleceń systemowych,
- łatwa edycja wpisywanych poleceń,
- uzupełnianie poleceń, ułatwiające wyszukiwanie poleceń i przyspieszające pracę,
- historia poleceń, umożliwiająca przeglądanie wcześniej wpisanych poleceń,
- łatwe uruchamianie zewnętrznych skryptów Pythona,
- łatwe usuwanie błędów za pomocą wbudowanego debugera.

Przekonajmy się na własnej skórze, jak działa IPython. Aby go uruchomić, wpisz następujące polecenie w wierszu poleceń:

```
$ ipython3
```

Po wpisaniu tego polecenia Twoim oczom ukaże się następujące okno:

```
Python 3.7.3 (default, Mar 27 2019, 22:11:17)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.4.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]:
```

Sprawdźmy niektóre możliwości oferowane przez powłokę IPython:

- **Historia poleceń** — za pomocą polecenia history możemy sprawdzić listę uprzednio wpisywanych poleceń. Poniższy rysunek prezentuje wynik wpisania tego polecenia:

```
In [1]: x=5
In [2]: x=x+5
In [3]: y=x+5
In [4]: x,y
Out[4]: (10, 15)
In [5]: history
x=5
x=x+5
y=x+5
x,y
history
```

- **Polecenia systemowe** — możemy także uruchamiać polecenia systemowe z poziomu powłoki IPython poprzez wstawienie na początku wykrzyknika (!). W takim przypadku polecenie wprowadzone po tym znaku będzie traktowane jako polecenie systemowe. Na przykład za pomocą polecenia `!date` wyświetlimy bieżącą datę systemową, natomiast polecenie `!pwd` wyświetla bieżący katalog roboczy:

```
$ ipython3
Python 3.7.3 (default, Mar 27 2019, 22:11:17)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.4.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: !pwd
/home/avinash
```

- **Pisanie funkcji** — możemy pisać tu funkcje tak samo jak w każdym innym środowisku IDE, np.: Jupyter Notebook, Python IDLE, PyCharm czy Spyder. Spójrzmy na przykładową funkcję:

```
In [1]: def helloworld():
...:     print("Witajcie wszyscy!")
...:
In [2]: helloworld()
Witajcie wszyscy!
```

- **Wyjście z powłoki** — możesz opuścić powłokę IPython za pomocą polecenia `exit()`:

```
$ ipython3
Python 3.7.3 (default, Mar 27 2019, 22:11:17)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.4.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: exit()
```

Możesz także dokonać tego, wpisując polecenie `quit()` lub wciskając kombinację klawiszy `CTRL+D`:

```
$ ipython3
Python 3.7.3 (default, Mar 27 2019, 22:11:17)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.4.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: quit()
```

W tym punkcie poznaliśmy kilka podstawowych poleceń dostępnych w powłoce IPython. Sprawdźmy teraz, do czego służy dostępne w niej polecenie `help`.

Odczytywanie stron podręcznika

W powłoce IPython możemy otworzyć listę dostępnych poleceń za pomocą polecenia `help`. Nie musimy wpisywać pełnej nazwy funkcji. Wystarczy wprowadzić kilka pierwszych znaków i wcisnąć klawisz `Tab`, a powłoka uzupełni poszukiwane słowo. Użyjmy np. funkcji `arrange()`. Informacje o funkcjach możemy uzyskać na dwa sposoby:

- **Za pomocą polecenia `help`** — wpisz polecenie `help` i kilka pierwszych liter funkcji. Wciśnij następnie klawisz `Tab`, wybierz właściwą funkcję za pomocą kursora i wciśnij `Enter`:

```
In [8]: help(numpy.ar
```

<code>arange()</code>	<code>arcsinh</code>	<code>argmax()</code>	<code>argwhere()</code>	<code>array_equal()</code>
<code>arccos</code>	<code>arctan</code>	<code>argmin()</code>	<code>around()</code>	<code>array_equiv()</code>
<code>arccosh</code>	<code>arctan2</code>	<code>argpartition()</code>	<code>array()</code>	<code>array_repr()</code>
<code>arcsin</code>	<code>arctanh</code>	<code>argsort()</code>	<code>array2string()</code>	<code>array_split()</code>

- **Za pomocą pytajnika** — możesz wprowadzić znak zapytania po nazwie funkcji. Poniżej został zaprezentowany przykład:

```
In [1]: import numpy
In [2]: numpy.arange?
```

W tym punkcie poznaliśmy znaczenie polecenia `help` i pytajnika w uzyskiwaniu informacji na temat funkcji. Możemy również uzyskać pomoc dzięki dokumentacji bibliotek. Nauczmy się pozyskiwać dokumentację związaną z bibliotekami analizy danych w Pythonie.

Źródła informacji na temat bibliotek analizy danych w Pythonie

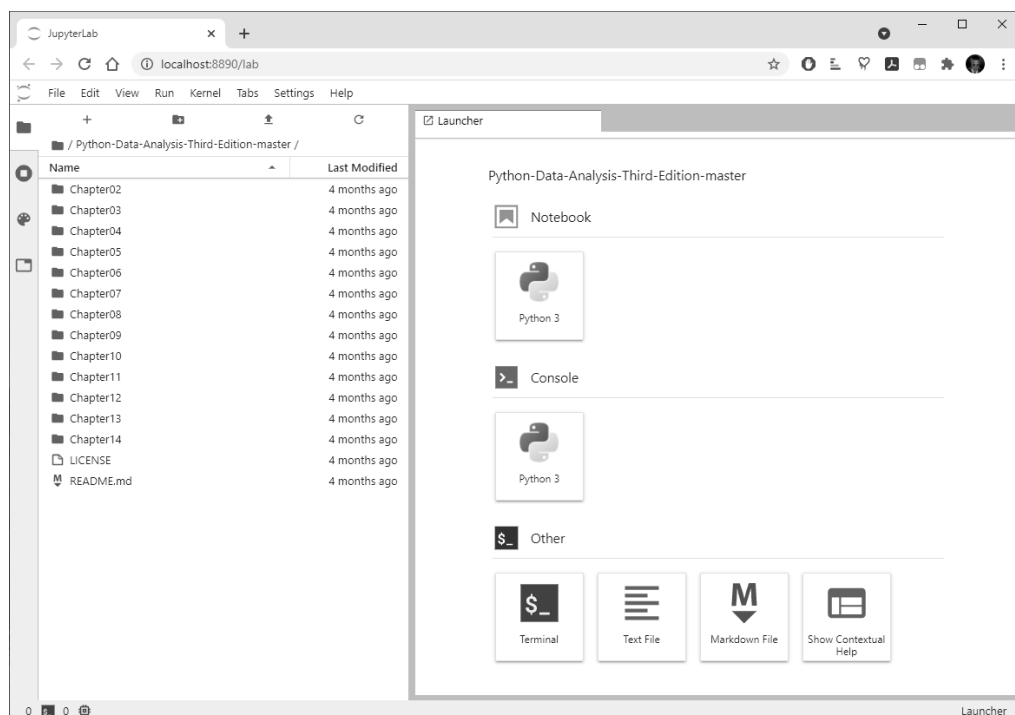
Poniższa tabela zawiera adresy dokumentacji omówionych w tym rozdziale bibliotek analizy danych Pythona:

Pakiety/oprogramowanie	Adres
NumPy	https://numpy.org/doc/
SciPy	https://docs.scipy.org/doc/
pandas	https://pandas.pydata.org/docs/
Matplotlib	https://matplotlib.org/3.2.1/contents.html
seaborn	https://seaborn.pydata.org/
scikit-learn	https://scikit-learn.org/stable/
Anaconda	https://www.anaconda.com/products/individual

Wiele odpowiedzi na zagadnienia programistyczne w Pythonie związane z bibliotekami NumPy, SciPy, pandas, Matplotlib, seaborn i scikit-learn znajdziesz w serwisie StackOverflow. Z kolei problemy z ich działaniem możesz zgłaszać w serwisie GitHub.

Korzystanie z aplikacji JupyterLab

JupyterLab to nowej generacji sieciowy interfejs użytkownika. Stanowi on połączenie narzędzi analizy danych i uczenia maszynowego, takich jak edytor tekstu, notatniki, konsole kodu i terminale. Jest to elastyczne i potężne narzędzie, które powinno się znajdować w arsenale każdego analityka danych:



Możesz zainstalować JupyterLab za pomocą poleceń conda, pip lub pipenv.

Instalacja za pomocą polecenia conda wygląda następująco:

```
$ conda install -c conda-forge jupyterlab
```

Za pomocą polecenia pip:

```
$ pip install jupyterlab
```

Z kolei za pomocą polecenia `pipenv`:

```
$ pipenv install jupyterlab
```

W tym podrozdziale nauczyliśmy się instalować aplikację JupyterLab. Przejdźmy teraz do aplikacji Jupyter Notebook.

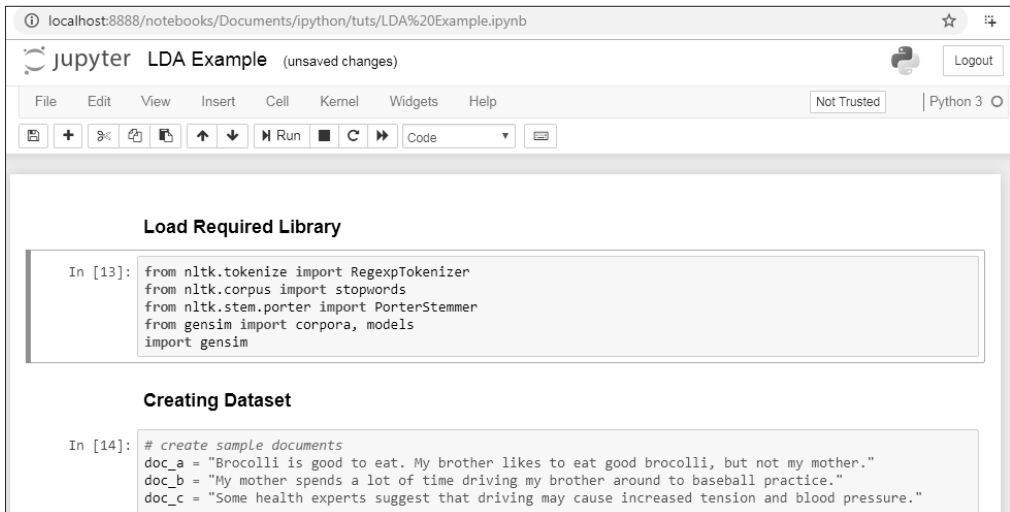
Stosowanie aplikacji Jupyter Notebook

Jupyter Notebook jest aplikacją sieciową wykorzystywaną do tworzenia notatników analizy danych, zawierających listingi, tekst, rysunki, odnośniki, równania matematyczne i wykresy. Od pewnego czasu dostępna jest również nowa generacja tej aplikacji o nazwie JupyterLab. Liczne przykłady notatników Jupyter znajdziesz pod następującymi adresami:

- <https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks>,
- <https://nbviewer.jupyter.org/>.

Notatniki te są często wykorzystywane w celach edukacyjnych lub do demonstrowania oprogramowania napisanego w Pythonie. Możemy importować lub eksportować notatniki za pomocą standardowego kodu Pythona lub przy użyciu wyspecjalizowanego formatu notatników. Można je uruchamiać lokalnie lub udostępniać sieciowo przez uruchomienie serwera notatników. Niektóre serwisy, takie jak Wakari, PiCloud czy Google Colaboratory, umożliwiają obsługę notatników w chmurze.

Słowo Jupyter stanowi akronim utworzony ze słów Julia, Python i R. Początkowo aplikacja ta została utworzona z myślą o tych trzech językach, obecnie jednak jest rozbudowana o obsługę wielu innych języków, takich jak C, C++, Scala, Perl, Go, PySpark czy Haskell:



The screenshot shows a Jupyter Notebook interface with the following content:

```

Load Required Library

In [13]: from nltk.tokenize import RegexpTokenizer
         from nltk.corpus import stopwords
         from nltk.stem.porter import PorterStemmer
         from gensim import corpora, models
         import gensim

Creating Dataset

In [14]: # create sample documents
         doc_a = "Broccoli is good to eat. My brother likes to eat good broccoli, but not my mother."
         doc_b = "My mother spends a lot of time driving my brother around to baseball practice."
         doc_c = "Some health experts suggest that driving may cause increased tension and blood pressure."

```

Jupyter Notebook oferuje następujące możliwości:

- edytowanie kodu w przeglądarce z uwzględnieniem właściwych wcięć wierszy,
- wykonywanie kodu z poziomu przeglądarki,
- wyświetlanie wyników w przeglądarce,
- wyświetlanie wykresów, obrazów i filmów w komórkach wynikowych,
- eksportowanie listingów do formatów PDF, HTML, LaTeX, a także plików Pythona.

Możemy także używać Pythona w wersjach 2 i 3 w aplikacji Jupyter Notebook; wystarczy wpisać poniższe polecenia w wierszu zachęty Anacondy:

```
# Python 2.7
conda create -n py27 python=2.7 ipykernel
```

```
# Python 3.5
conda create -n py35 python=3.5 ipykernel
```

Znamy już różne przydatne narzędzia oraz biblioteki, a także zainstalowaliśmy Pythona, przejdźmy więc do bardziej zaawansowanych funkcji najczęściej używanej aplikacji, czyli Jupyter Notebook.

Zaawansowane funkcje aplikacji Jupyter Notebook

Aplikacja Jupyter Notebook zawiera wiele przydatnych funkcji, takich jak skróty klawiszowe, możliwość instalacji innych jąder, wykonywanie poleceń powłoki, a także stosowanie różnych rozszerzeń przyspieszających analizę danych. Przyjrzyjmy się kolejno tym funkcjom.

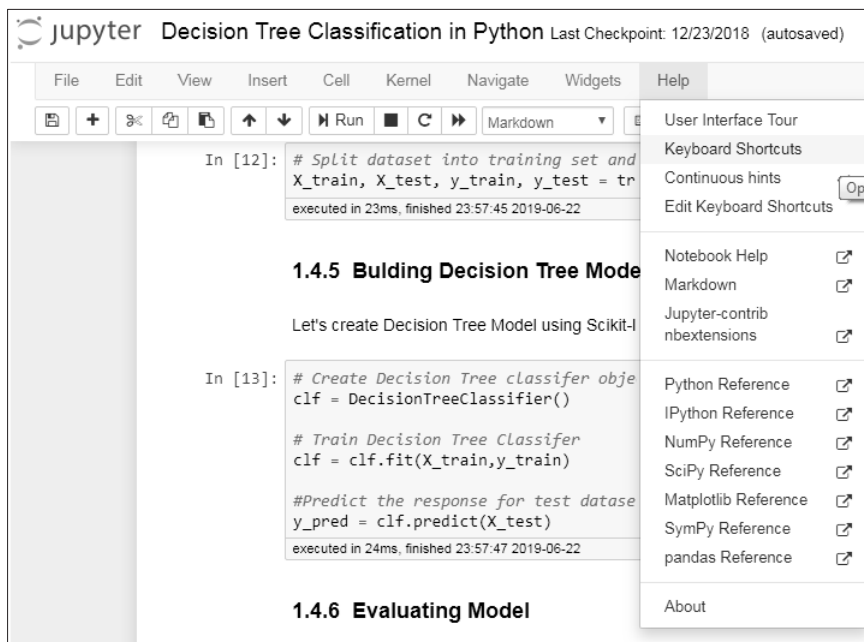
Skróty klawiszowe

Wszystkie skróty klawiszowe dostępne w aplikacji Jupyter Notebook zostaną wyświetlone po wciśnięciu opcji *Keyboard Shortcuts* w menu *Help* lub po wciśnięciu kombinacji klawiszy *Cmd+Shift+P*. Pojawi się okno zawierające listę skrótów klawiszowych wraz z ich krótkim opisem. Jest to bardzo przydatna funkcja, gdy wyleci nam z głowy jakiś ważny skrót (patrz zrzut na następnej stronie).

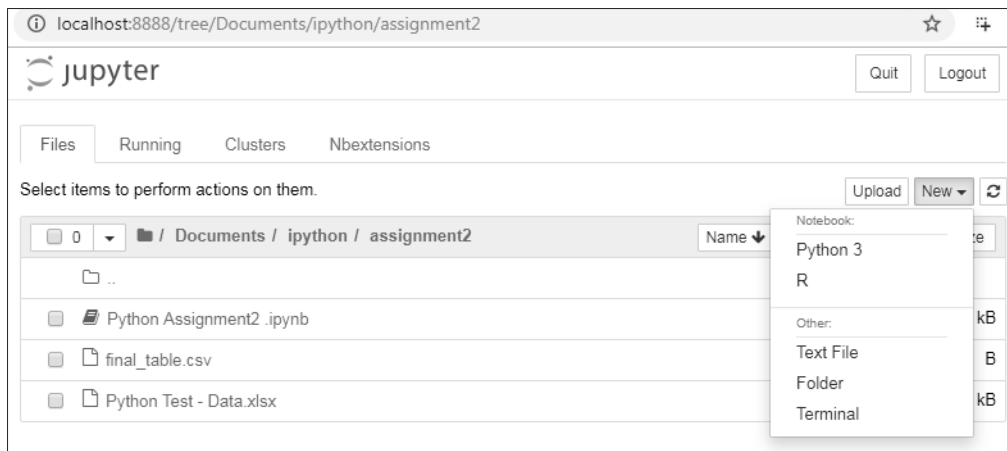
Instalowanie innych jąder

Aplikacja Jupyter Notebook może wykorzystywać jądra różnych języków. W Anacondzie utworzenie środowiska obsługującego określony język nie stanowi żadnego wyzwania. Możesz np. wyznaczyć jądro języka R w następujący sposób:

```
$ conda install -c r r-essentials
```



W ten sposób w dostępnych jądrach powinno się pojawić jądro R, tak jak zaprezentowano poniżej:



Realizowanie poleceń powłoki

Użytkownicy aplikacji Jupyter Notebook mogą w niej wykonywać polecenia powłoki systemów Unix i Windows. Powłoka stanowi interfejs umożliwiający komunikację z komputerem. Użytkownik musi przed danym poleceniem wstawić wykrzykownik (!):

```
In [1]: ldir

Volume in drive C has no label.
Volume Serial Number is D0C9-975F

Directory of C:\Users\Aadmin\Documents\ipython\tuts

22-06-2019  23:34  <DIR>          .
22-06-2019  23:34  <DIR>          ..
11-05-2019  16:19                1,147,737  Inventory Model Simulation with Spreadsheet.ipynb
22-06-2019  22:56  <DIR>          .ipynb_checkpoints
24-02-2019  20:10                23,475  AB Testing.ipynb
02-05-2019  21:15                150,385  AdaBoost Classifier in Python-Copy1.ipynb
19-03-2019  09:21                191,553  AdaBoost Classifier in Python.ipynb
10-07-2018  15:54                25,337,978  articles.txt
20-12-2018  15:26                163,127  Cohort Analysis.ipynb
18-03-2019  09:15  <DIR>          computer_vision
22-06-2019  23:34                209,831  Customer Life Time Value (final version).ipynb
11-05-2019  10:03                81,155  Customer life time value V1.ipynb
11-05-2019  10:01                431,692  Customer Lifetime Value - Predictive Modeling.ipynb
```

Rozszerzenia

Rozszerzenia notatników (ang. *notebook extensions* — `nbextensions`) dodają pewne funkcje do standardowych notatników Jupyter. Funkcje te usprawniają interfejs oraz poprawiają doświadczenie użytkownika. Można z łatwością wybrać interesujące nas rozszerzenia w zakładce *NBextensions*.

Aby zainstalować rozszerzenie aplikacji Jupyter Notebook za pomocą menedżera `conda`, wpisz następujące polecenie:

```
conda install -c conda-forge jupyter_nbextensions_configurator
```

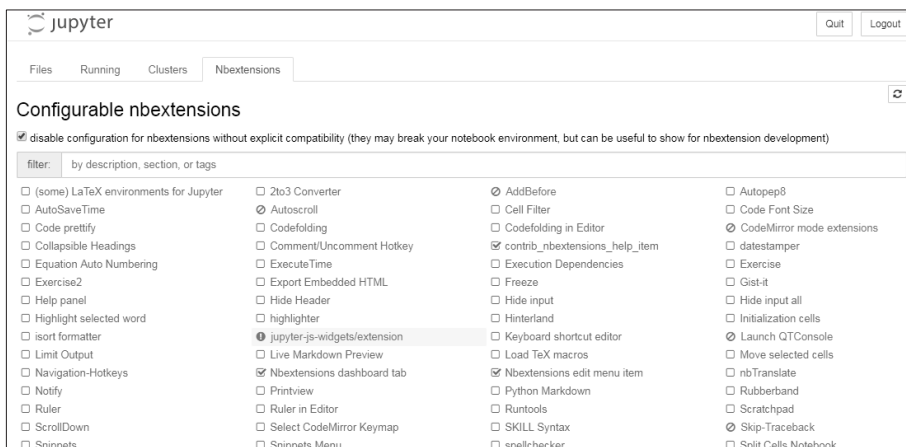
Możemy dokonać tego samego również za pomocą menedżera `pip`:

```
pip install jupyter_contrib_nbextensions && jupyter contrib nbextension install
```

Jeżeli w przypadku systemu `macOS` pojawiają się błędy uprawnień, użyj poniższego polecenia:

```
pip install jupyter_contrib_nbextensions && jupyter contrib nbextension install --user
```

Wszystkie modyfikowalne rozszerzenia będą umieszczone w osobnej zakładce, tak jak widać poniżej:



Przjrzyjmy się teraz kilku przydatnym rozszerzeniom Jupyter Notebook:

- **Hinterland** — dodaje menu autouzupełniania każdego znaku wpisywanego do komórki i działa podobnie jak PyCharm:

```
In [8]: import pandas as pd
import num
      |
      | numba
      | numbers
      | numexpr
      | numpy
      | numpydoc
```

- **Spis treści (Table of Contents)** — rozszerzenie to pokazuje wszystkie nagłówki w panelu bocznym lub w menu nawigacji. Można zmieniać rozmiar tego panelu, przesuwać go, minimalizować i dokować w wybranym miejscu:

The screenshot shows a Jupyter Notebook interface with the following content:

Table of Contents

- 1 Understanding NumPy Array
- 2 NumPy Array Numerical Data Types
- 3 Manipulating Shape of NumPy Array
- 4 Stacking of Numpy arrays
- 5 Partitioning Numpy Array
- 6 Changing Datatype of NumPy Array:
- 7 Creating NumPy views and copies
- 8 Slicing NumPy Array
- 9 Boolean and Fancy Indexing
- 10 Broadcasting arrays
- 11 Create DataFrame
- 12 Pandas Series
- 13 Querying Data
- 14 Statistics
- 15 Grouping Pandas DataFrames
- 16 Joins
- 17 Missing Values
- 18 Pivot Table
- 19 Dealing with dates

1 Understanding NumPy Array

```
In [1]: # Creating an array
import numpy as np
a = np.array([2,4,6,8,10])
print(a)

[ 2  4  6  8 10]
```

```
In [2]: # Creating an array using arange()
import numpy as np
a = np.arange(1,11)
print(a)

[ 1  2  3  4  5  6  7  8  9 10]
```

```
In [3]: import numpy as np

p = np.zeros((3,3)) # Create an array of all zeros
print(p)

q = np.ones((2,2)) # Create an array of all ones
print(q)
```

- **Czas realizacji (Execute Time)** — rozszerzenie to pokazuje datę realizacji kodu zawartego w danej komórce oraz czas, jaki został poświęcony na tę operację:

```
In [13]: # Create Decision Tree classifier object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifier
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

executed in 24ms, finished 23:57:47 2019-06-22
```

- **Sprawdzanie pisowni (Spellchecker)** — rozszerzenie to sprawdza i weryfikuje ortografię (języka angielskiego) w każdej komórce oraz zaznacza ewentualne literówki.
- **Dobór zmiennych (Variable Selector)** — rozszerzenie to organizuje przestrzeń roboczą użytkownika. Pokazuje nazwy wszystkich zmiennych utworzonych przez użytkownika, a także ich typ, rozmiar, wymiary i wartość:

The screenshot shows a Jupyter Notebook interface with a 'Variable Inspector' widget on the right. The notebook content includes code for splitting a dataset, building a Decision Tree Model, and evaluating it. The Variable Inspector displays the following table:

Name	Type	Size	Shape	Value
DecisionTreeC	ABCMeta	1484		
X	DataFrame	42088	(788, 7)	pregnant insulin bmi age gl...
X_test	DataFrame	14784	(231, 7)	pregnant insulin bmi age gl...
X_train	DataFrame	34355	(537, 7)	pregnant insulin bmi age gl...
clf	DecisionTreeC	55		DecisionTreeClassifier(estimator...
cat_names	list	138		['pregnant', 'glucose', 'bmi', 'skinf...
feature_cols	list	120		['pregnant', 'insulin', 'bmi', 'age', ...]
pima	DataFrame	55376	(788, 6)	pregnant glucose bp skin in...
y	Series	6144	(788,)	0 1 1 0 2 1 3 0 4
y_pred	ndarray	1548	(231,)	[0 0 0 0 0 1 0 1 0 1 1 1 0 0 0 ...]
y_test	Series	1548	(231,)	587 0 123 0 616 0 462 0 2
y_train	Series	4296	(537,)	62 0 281 0 199 1 744 0 2

- **Pokaz slajdów (Slideshow)** — wyniki obliczone w notatnikach można prezentować za pomocą rozszerzenia Slideshow. Jest to doskonałe narzędzie do opowiadania historyjek. Użytkownicy mogą z łatwością przekształcić notatniki w prezentację bez używania aplikacji PowerPoint. Jak widać na poniższym rysunku, można uruchomić to rozszerzenie za pomocą opcji *Slideshow* dostępnej w menu *Cell Toolbar*, otrzymywanym po rozwinięciu zakładki *View*:

The screenshot shows a Jupyter Notebook interface with the 'View' menu open. The 'Slideshow' option is highlighted. The notebook content includes code for importing modules: pandas, matplotlib, seaborn, datetime, and numpy. The background shows the text 'CLTV Implem' and 'on(Using Formula)'.

Istnieje także możliwość wyświetlania lub ukrywania dowolnej komórki w ramach rozszerzenia Slideshow. Po dodaniu opcji *Slideshow* do paska narzędzi komórki w menu *View* wystarczy użyć listy rozwijalnej *Slide Type* w każdej komórce i wybrać interesujące nas ustawienie, co widać na poniższym zrzucie ekranu:

Jupyter Customer Life Time Value (final version) Last Checkpoint: 11/03/2018 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

In [74]:

```
# import model
from sklearn.linear_model import LinearRegression

# instantiate
linreg = LinearRegression()

# fit the model to the training data (Learn the coefficients)
linreg.fit(X_train, y_train)
```

Slide Type: Slide, Sub-Slide, Fragment, Skip, Notes

- **Umieszczanie dokumentów PDF** — użytkownicy aplikacji Jupyter Notebook mogą z łatwością dodawać dokumenty PDF. Należy skorzystać z następującej składni:

```
from IPython.display import IFrame
IFrame('https://arxiv.org/pdf/1811.02141.pdf', width=700,
height=400)
```

Otrzymamy następujący rezultat:

In [3]:

```
from IPython.display import IFrame
IFrame('https://arxiv.org/pdf/1811.02141.pdf', width=700, height=400)
```

Out[3]:

1

Extended Isolation Forest

Sahand Hariri, Matias Carrasco Kind, Robert J. Brunner

Abstract—We present an extension to the model-free anomaly detection algorithm, Isolation Forest. This extension, named Extended Isolation Forest (EIF), resolves issues with assignment of anomaly score to given data points. We motivate the problem using heat maps for anomaly scores. These maps suffer from artifacts generated by the criteria for branching operation of the binary tree. We explain this problem in detail and demonstrate the mechanism by which it occurs visually. We then propose two different approaches for improving the situation. First we propose transforming the data randomly before creation of each tree, which results in averaging out the bias. Second, which is the preferred way, is to allow the slicing of the data to use hyperplanes with random slopes. This approach results in remedying the artifact seen in the anomaly score heat maps. We show that the robustness of the algorithm is much improved using this method by looking at the variance of scores of data points distributed along constant level sets. We report AUROC and AUPRC for our synthetic datasets, along with real-world benchmark datasets. We find no appreciable difference in the rate of convergence nor in computation time between the standard Isolation Forest and EIF.

Index Terms—Anomaly Detection, Isolation Forest

ov 2019

- **Umieszczanie filmików z serwisu YouTube** — użytkownicy aplikacji Jupyter Notebook mogą z łatwością dodawać dokumenty, a także filmiki z serwisu YouTube. Należy skorzystać z następującej składni:

```
from IPython.display import YouTubeVideo
YouTubeVideo('ukzF19rgwfu', width=700, height=400)
```

Otrzymamy następujący rezultat:

```
In [2]: from IPython.display import YouTubeVideo
        YouTubeVideo('ukzFI9rgwFU', width=800, height=300)

Out[2]:
```



Wiesz już, czym jest analiza danych, na czym polega ten proces, a także czym cechują się związane z nim role. Nauczyłeś się także instalować środowisko Python oraz aplikacje JupyterLab i Jupyter Notebook. W kolejnych rozdziałach dowiesz się więcej na temat różnych bibliotek Pythona i technik analizy danych.

Podsumowanie

W tym rozdziale omówiliśmy różne procesy analizy danych, takie jak KDD, SEMMA czy CRISP-DM. Następnie przyjrzelśmy się rolom i umiejętnościom analityków danych oraz danetyków. W dalszej części rozdziału zainstalowaliśmy biblioteki NumPy, SciPy, pandas, Matplotlib, a także aplikacje IPython, Jupyter Notebook, Anaconda i JupyterLab, ponieważ będziemy z nich korzystać w pozostałych rozdziałach. Nie musisz instalować tych modułów, lecz wystarczy zainstalować Anacondę lub JupyterLab, gdyż mają wbudowane pakiety NumPy, pandas, SciPy i scikit-learn.

Następnie napisaliśmy krótki programik dodający wektory i przekonaliśmy się, że NumPy ma o wiele większą wydajność w porównaniu do innych bibliotek. Przejrzeliśmy dostępną dokumentację i zasoby internetowe. Ponadto poznaliśmy aplikacje JupyterLab, Jupyter Notebook oraz ich zaawansowane funkcje.

W następnym rozdziale zajmiemy się bibliotekami NumPy i pandas, a przy okazji omówimy podstawowe pojęcia dotyczące tablic i obiektów DataFrame.

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 

Python: wydobywaj z danych wiedzę o wielkiej wartości!

Analiza danych sprawia, że dzięki ich dużym i mniejszym kolekcjom uzyskujemy wartościową wiedzę, która pozwala na podejmowanie najlepszych decyzji. Dzieje się to poprzez odkrywanie wzorców lub trendów. Obecnie Python udostępnia przeznaczone specjalnie do tego celu narzędzia i biblioteki. Możemy więc łatwo korzystać z wyrafinowanych technik wydobywania wiedzy z danych. Aby jednak osiągnąć zamierzone efekty, trzeba dobrze poznać zarówno metodologię analizy danych, jak i zasady pracy ze służącymi do tego narzędziami.

Dzięki tej książce zdobędziesz wszystkie potrzebne informacje i umiejętności, aby skutecznie używać Pythona do analizy danych. Omówiono tu niezbędne podstawy statystyki i zasady analizy danych. Wyczerpująco przedstawiono zaawansowane zagadnienia dotyczące przygotowania, przetwarzania i modelowania danych, a także ich wizualizacji. W zrozumiały sposób wyjaśniono takie procesy jak inteligentne przetwarzanie i analizowanie danych za pomocą algorytmów uczenia maszynowego: regresji, klasyfikacji, analizy głównych składowych czy analizy skupień. Nie zabrakło praktycznych przykładów przetwarzania języka naturalnego i analizy obrazów. Ciekawym zagadnieniem jest również wykonywanie obliczeń równoległych za pomocą biblioteki Dask.

W książce między innymi:

- podstawy analizy danych i korzystanie z bibliotek NumPy i pandas
- praca z danymi w różnych formatach
- interaktywna wizualizacja z bibliotekami Matplotlib, seaborn i Bokeh
- inżynieria cech, analiza szeregów czasowych i przetwarzanie sygnałów
- zaawansowana analiza danych tekstowych i obrazów

Avinash Navlani — był wykładowcą akademickim, obecnie wdraża narzędzia do analizy danych wielkoskalowych i tworzenia modeli. Angażuje się w projekty badawcze.

Armando Fandango — jest ekspertem w dziedzinach uczenia głębokiego, uczenia maszynowego i uczenia rozproszonego. Publikuje artykuły w czasopiśmie branżowych.

Ivan Idris — jest programistą, twórcą hurtowni danych i analitykiem biznesu. Słynie ze schludnego kodu i z interesującego sposobu pisania.

Helion	<i>Sprawdź nasze szkolenia!</i>	KOD KORZYŚCI Stęgnij po więcej! ▶	
 helion.pl	SZKOLENIA  AKADEMIA IT & BUSINESS	ISBN 978-83-283-8360-9	
 0 801 339900	WWW.SZKOLENIA.HELION.PL		
 0 601 339900	INFORMATYKA W NAJLEPSZYM WYDANIU	9 788328 383609	Cena: 89,00 zł

Packt