

O'REILLY®

Wydanie III

Uczenie maszynowe z użyciem Scikit-Learn, Keras i TensorFlow

powered by



Helion 

Aurélien Géron

Tytuł oryginału: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow:
Concepts, Tools, and Techniques to Build Intelligent Systems, 3rd Edition

Tłumaczenie: Krzysztof Sawka

ISBN: 978-83-8322-423-7

© 2023 Helion S.A.

Authorized Polish translation of the English edition of *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 3E ISBN 9781098125974 © 2023 Aurélien Géron.

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

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

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

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

Autor oraz 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 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. 32 230 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/uczem3>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

Przedmowa	15
-----------------	----

Część I. Podstawy uczenia maszynowego 25

1. Krajobraz uczenia maszynowego	27
Czym jest uczenie maszynowe?	28
Dlaczego warto korzystać z uczenia maszynowego?	28
Przykładowe zastosowania	31
Rodzaje systemów uczenia maszynowego	33
Nadzorowanie uczenia	33
Uczenie wsadowe i uczenie przyrostowe	40
Uczenie z przykładów i uczenie z modelu	44
Główne problemy uczenia maszynowego	49
Niedobór danych uczących	50
Niereprezentatywne dane uczące	50
Dane kiepskiej jakości	52
Nieistotne cechy	52
Przetrenowanie danych uczących	53
Niedotrenowanie danych uczących	55
Podsumowanie	55
Testowanie i ocenianie	56
Strojenie hiperparametrów i dobór modelu	56
Niezgodność danych	57
Ćwiczenia	59
2. Nasz pierwszy projekt uczenia maszynowego	61
Praca z rzeczywistymi danymi	61
Przeanalizuj całokształt projektu	63
Określ zakres problemu	63
Wybierz wskaźnik wydajności	65
Sprawdź założenia	67

Zdobądź dane	67
Uruchom przykładowy kod w serwisie Google Colab	68
Zapisz zmiany w kodzie i w danych	70
Zalety i wady interaktywności	71
Kod w książce a kod w notatnikach Jupyter	72
Pobierz dane	72
Rzut oka na strukturę danych	73
Stwórz zbiór testowy	77
Odkrywaj i wizualizuj dane, aby zdobywać nowe informacje	81
Zwizualizuj dane geograficzne	81
Poszukaj korelacji	84
Eksperymentuj z kombinacjami atrybutów	86
Przygotuj dane pod algorytmy uczenia maszynowego	87
Oczyść dane	88
Obsługa tekstu i atrybutów kategoryalnych	90
Skalowanie i przekształcanie cech	94
Niestandardowe transformatory	97
Potoki transformujące	101
Wybierz i wytrenuj model	105
Trenuj i oceń model za pomocą zbioru uczącego	106
Dokładniejsze ocenianie za pomocą sprawdzianu krzyżowego	107
Wyreguluj swój model	109
Metoda przeszukiwania siatki	109
Metoda losowego przeszukiwania	111
Metody zespołowe	112
Analizowanie najlepszych modeli i ich błędów	112
Oceń system za pomocą zbioru testowego	113
Uruchom, monitoruj i utrzymuj swój system	114
Teraz Twoja kolej!	117
Ćwiczenia	117
3. Klasyfikacja	119
Zbiór danych MNIST	119
Uczenie klasyfikatora binarnego	122
Miary wydajności	122
Pomiar dokładności za pomocą sprawdzianu krzyżowego	123
Macierz pomyłek	124
Precyzja i pełność	125
Kompromis pomiędzy precyzją a pełnością	127
Wykres krzywej ROC	130

Klasyfikacja wieloklasowa	134
Analiza błędów	136
Klasyfikacja wieloetykietowa	140
Klasyfikacja wielowyjściowa	141
Ćwiczenia	143
4. Uczenie modeli	145
Regresja liniowa	146
Równanie normalne	148
Złożoność obliczeniowa	151
Gradient prosty	151
Wsadowy gradient prosty	154
Stochastyczny spadek wzdłuż gradientu	157
Schodzenie po gradiencie z minigrupami	160
Regresja wielomianowa	161
Krzywe uczenia	163
Regularyzowane modele liniowe	167
Regresja grzbietowa	167
Regresja metodą LASSO	170
Regresja metodą elastycznej siatki	172
Wczesne zatrzymywanie	173
Regresja logistyczna	174
Szacowanie prawdopodobieństwa	175
Funkcje ucząca i kosztu	176
Granice decyzyjne	177
Regresja softmax	180
Ćwiczenia	183
5. Maszyny wektorów nośnych	185
Liniowa klasyfikacja SVM	185
Klasyfikacja miękkiego marginesu	186
Nieliniowa klasyfikacja SVM	188
Jądro wielomianowe	189
Cechy podobieństwa	190
Gaussowskie jądro RBF	191
Klasy SVM i złożoność obliczeniowa	192
Regresja SVM	193
Mechanizm działania liniowych klasyfikatorów SVM	195
Problem dualny	198
Kernelizowane maszyny SVM	199
Ćwiczenia	202

6. Drzewa decyzyjne	203
Uczenie i wizualizowanie drzewa decyzyjnego	203
Wylizywanie prognoz	204
Szacowanie prawdopodobieństw przynależności do klas	206
Algorytm uczący CART	207
Złożoność obliczeniowa	208
Wskaźnik Giniego czy entropia?	208
Hiperparametry regularyzacyjne	209
Regresja	211
Wrażliwość na orientację osi	213
Drzewa decyzyjne mają znaczną wariancję	214
Ćwiczenia	215
7. Uczenie zespołowe i losowe lasy	217
Klasyfikatory głosujące	217
Agregacja i wklejanie	221
Agregacja i wklejanie w module Scikit-Learn	222
Ocena OOB	223
Rejony losowe i podprzestrzenie losowe	224
Losowe lasy	225
Zespół Extra-Trees	225
Istotność cech	226
Wzmacnianie	227
AdaBoost	227
Wzmacnianie gradientowe	231
Wzmacnianie gradientu w oparciu o histogram	234
Kontaminacja	235
Ćwiczenia	238
8. Redukcja wymiarowości	240
Kłątwa wymiarowości	241
Główne strategie redukcji wymiarowości	242
Rzutowanie	242
Uczenie różnorodnościowe	244
Analiza PCA	245
Zachowanie wariancji	246
Główne składowe	247
Rzutowanie na d wymiarów	248
Implementacja w module Scikit-Learn	249
Współczynnik wariancji wyjaśnionej	249
Wybór właściwej liczby wymiarów	249

Algorytm PCA w zastosowaniach kompresji	251
Losowa analiza PCA	252
Przyrostowa analiza PCA	252
Rzutowanie losowe	254
Algorytm LLE	256
Inne techniki redukowania wymiarowości	258
Ćwiczenia	259

9. Techniki uczenia nienadzorowanego261

Analiza skupień: algorytm centroidów i DBSCAN	262
Algorytm centroidów	264
Granice algorytmu centroidów	273
Analiza skupień w segmentacji obrazu	274
Analiza skupień w uczeniu półnadzorowanym	276
Algorytm DBSCAN	279
Inne algorytmy analizy skupień	282
Mieszanki gaussowskie	283
Wykrywanie anomalii za pomocą mieszanin gaussowskich	287
Wyznaczanie liczby skupień	289
Bayesowskie modele mieszane	291
Inne algorytmy służące do wykrywania anomalii i nowości	292
Ćwiczenia	293

Część II. Sieci neuronowe i uczenie głębokie 295

10. Wprowadzenie do sztucznych sieci neuronowych i ich implementacji z użyciem interfejsu Keras297

Od biologicznych do sztucznych neuronów	298
Neurony biologiczne	299
Operacje logiczne przy użyciu neuronów	300
Perceptron	301
Perceptron wielowarstwowy i propagacja wsteczna	306
Regresyjne perceptrony wielowarstwowe	309
Klasyfikacyjne perceptrony wielowarstwowe	311
Implementowanie perceptronów wielowarstwowych za pomocą interfejsu Keras	313
Tworzenie klasyfikatora obrazów za pomocą interfejsu sekwencyjnego	314
Tworzenie regresyjnego perceptronu wielowarstwowego za pomocą interfejsu sekwencyjnego	323
Tworzenie złożonych modeli za pomocą interfejsu funkcyjnego	324
Tworzenie modeli dynamicznych za pomocą interfejsu podklasowego	329
Zapisywanie i odczytywanie modelu	331

Stosowanie wywołań zwrotnych	332
Wizualizacja danych za pomocą narzędzia TensorBoard	333
Dostrajanie hiperparametrów sieci neuronowej	337
Liczba warstw ukrytych	341
Liczba neuronów w poszczególnych warstwach ukrytych	342
Współczynnik uczenia, rozmiar grupy i pozostałe hiperparametry	343
Ćwiczenia	345
11. Uczenie głębokich sieci neuronowych	348
Problemy zanikających/eksplodujących gradientów	348
Inicjalizacje wag Glorota i He	349
Lepsze funkcje aktywacji	351
Normalizacja wsadowa	358
Obcinanie gradientu	363
Wielokrotne stosowanie gotowych warstw	363
Uczenie transferowe w interfejsie Keras	365
Nienadzorowane uczenie wstępne	366
Uczenie wstępne za pomocą dodatkowego zadania	367
Szybsze optymalizatory	368
Optymalizacja momentum	369
Przyspieszony spadek wzdłuż gradientu (algorytm Nesterova)	370
AdaGrad	371
RMSProp	373
Optymalizator Adam	373
AdaMax	374
Nadam	375
AdamW	375
Harmonogramowanie współczynnika uczenia	377
Regularyzacja jako sposób zapobiegania przetrenowaniu	381
Regularyzacja ℓ_1 i ℓ_2	381
Porzucanie	382
Regularyzacja typu Monte Carlo (MC)	385
Regularyzacja typu max-norm	387
Podsumowanie i praktyczne wskazówki	388
Ćwiczenia	390
12. Modele niestandardowe i uczenie za pomocą modułu TensorFlow	391
Krótkie omówienie modułu TensorFlow	391
Korzystanie z modułu TensorFlow jak z biblioteki NumPy	394
Tensory i operacje	395
Tensory a biblioteka NumPy	396

Konwersje typów	397
Zmienne	397
Inne struktury danych	398
Dostosowywanie modeli i algorytmów uczenia	399
Niestandardowe funkcje straty	399
Zapisywanie i wczytywanie modeli zawierających elementy niestandardowe	400
Niestandardowe funkcje aktywacji, inicjalizatory, regularyzatory i ograniczenia	402
Niestandardowe wskaźniki	403
Niestandardowe warstwy	405
Niestandardowe modele	408
Funkcje straty i wskaźniki oparte na elementach wewnętrznych modelu	410
Obliczanie gradientów za pomocą różniczkowania automatycznego	411
Niestandardowe pętle uczenia	415
Funkcje i grafy modułu TensorFlow	417
AutoGraph i kreślenie	419
Reguły związane z funkcją TF	421
Ćwiczenia	422
13. Wczytywanie i wstępne przetwarzanie danych za pomocą modułu TensorFlow	424
Interfejs tf.data	425
Łączenie przekształceń	426
Tasowanie danych	428
Przeplatanie wierszy z różnych plików	429
Wstępne przetwarzanie danych	430
Składanie wszystkiego w całość	431
Pobieranie wstępne	432
Stosowanie zestawu danych z interfejsem Keras	434
Format TFRecord	435
Skompresowane pliki TFRecord	436
Wprowadzenie do buforów protokołów	436
Bufory protokołów w module TensorFlow	438
Wczytywanie i analizowanie składni obiektów Example	439
Obsługa list list za pomocą bufora protokołów SequenceExample	440
Warstwy przetwarzania wstępnego Keras	441
Warstwa Normalization	441
Warstwa Discretization	444
Warstwa CategoryEncoding	444
Warstwa StringLookup	446
Warstwa Hashing	447
Kodowanie cech kategoryalnych za pomocą wektorów właściwościowych	447
Wstępne przetwarzanie tekstu	451

Korzystanie z wytrenowanych składników modelu językowego	453
Warstwy wstępne przetwarzania obrazów	454
Projekt TensorFlow Datasets (TFDS)	455
Ćwiczenia	456
14. Głębokie widzenie komputerowe za pomocą spłotowych sieci neuronowych	458
Struktura kory wzrokowej	459
Warstwy spłotowe	460
Filtry	462
Stosy map cech	463
Implementacja warstw spłotowych w interfejsie Keras	465
Zużycie pamięci operacyjnej	468
Warstwa łącząca	469
Implementacja warstw łączących w interfejsie Keras	471
Architektury spłotowych sieci neuronowych	473
LeNet-5	475
AlexNet	476
GoogLeNet	479
VGGNet	482
ResNet	482
Xception	486
SENet	487
Inne interesujące struktury	489
Wybór właściwej struktury CNN	491
Implementacja sieci ResNet-34 za pomocą interfejsu Keras	492
Korzystanie z gotowych modeli w interfejsie Keras	493
Gotowe modele w uczeniu transferowym	494
Klasyfikowanie i lokalizowanie	497
Wykrywanie obiektów	499
W pełni połączone sieci spłotowe	501
Sieć YOLO	503
Śledzenie obiektów	506
Segmentacja semantyczna	507
Ćwiczenia	510
15. Przetwarzanie sekwencji za pomocą sieci rekurencyjnych i spłotowych	512
Neurony i warstwy rekurencyjne	513
Komórki pamięci	515
Sekwencje wejść i wyjść	515
Uczenie sieci rekurencyjnych	517

Prognozowanie szeregów czasowych	517
Rodzina modeli ARMA	522
Przygotowywanie danych dla modeli uczenia maszynowego	525
Prognozowanie za pomocą modelu liniowego	528
Prognozowanie za pomocą prostej sieci rekurencyjnej	529
Prognozowanie za pomocą głębokich sieci rekurencyjnych	530
Prognozowanie wielowymiarowych szeregów czasowych	531
Prognozowanie kilka taktów w przód	532
Prognozowanie za pomocą modelu sekwencyjnego	534
Obsługa długich sekwencji	537
Zwalczanie problemu niestabilnych gradientów	537
Zwalczanie problemu pamięci krótkotrwałej	540
Ćwiczenia	547
16. Przetwarzanie języka naturalnego	
za pomocą sieci rekurencyjnych i mechanizmów uwagi549	
Generowanie tekstów szekspirowskich za pomocą znakowej sieci rekurencyjnej	550
Tworzenie zestawu danych uczących	551
Budowanie i uczenie modelu char-RNN	553
Generowanie sztucznego tekstu szekspirowskiego	554
Stanowe sieci rekurencyjne	555
Analiza opinii	558
Maskowanie	560
Korzystanie z gotowych reprezentacji właściwościowych i modeli językowych	563
Sieć typu koder – dekodek służąca do neuronowego tłumaczenia maszynowego	565
Dwukierunkowe sieci rekurencyjne	571
Przeszukiwanie wiązkowe	572
Mechanizmy uwagi	574
Liczy się tylko uwaga: pierwotna architektura transformatora	578
Zatręsenie modeli transformatorów	587
Transformatory wizualne	592
Biblioteka Transformers firmy Hugging Face	596
Ćwiczenia	599
17. Autokodery, generatywne sieci przeciwstawne i modele rozpraszające601	
Efektywne reprezentacje danych	602
Analiza PCA za pomocą niedopełnionego autokodera liniowego	604
Autokodery stosowe	605
Implementacja autokodera stosowego za pomocą interfejsu Keras	606
Wizualizowanie rekonstrukcji	607
Wizualizowanie zestawu danych Fashion MNIST	608

Nienadzorowane uczenie wstępne za pomocą autokoderów stosowych	609
Wiązanie wag	610
Uczenie autokoderów pojedynczo	611
Autokodery spłotowe	613
Autokodery odszumiające	614
Autokodery rzadkie	615
Autokodery wariacyjne	618
Generowanie obrazów Fashion MNIST	622
Generatywne sieci przeciwstawne	623
Problemy związane z uczeniem sieci GAN	627
Głębokie spłotowe sieci GAN	628
Rozrost progresywny sieci GAN	631
Sieci StyleGAN	634
Modele rozprasające	636
Ćwiczenia	642
18. Uczenie przez wzmacnianie	644
Uczenie się optymalizowania nagród	645
Wyszukiwanie strategii	646
Wprowadzenie do narzędzia OpenAI Gym	648
Sieci neuronowe jako strategie	652
Ocenianie czynności: problem przypisania zasługi	654
Gradienty strategii	655
Procesy decyzyjne Markowa	659
Uczenie metodą różnic czasowych	663
Q-uczenie	664
Strategie poszukiwania	665
Przybliżający algorytm Q-uczenia i Q-uczenie głębokie	666
Implementacja modelu Q-uczenia głębokiego	667
Odmiany Q-uczenia głębokiego	671
Ustalone Q-wartości docelowe	671
Podwójna sieć DQN	672
Odtwarzanie priorytetowych doświadczeń	672
Walcząca sieć DQN	673
Przegląd popularnych algorytmów RN	674
Ćwiczenia	678
19. Wielkoskalowe uczenie i wdrażanie modeli TensorFlow	679
Eksploracja modelu TensorFlow	680
Korzystanie z systemu TensorFlow Serving	680
Tworzenie usługi predykcyjnej na platformie Vertex AI	688
Wykonywanie zadań predykcji wsadowych w usłudze Vertex AI	695

Wdrażanie modelu na urządzeniu mobilnym lub wbudowanym	697
Przetwarzanie modelu na stronie internetowej	699
Przyspieszanie obliczeń za pomocą procesorów graficznych	701
Zakup własnej karty graficznej	702
Zarządzanie pamięcią operacyjną karty graficznej	704
Umieszczanie operacji i zmiennych na urządzeniach	706
Przetwarzanie równoległe na wielu urządzeniach	708
Uczenie modeli za pomocą wielu urządzeń	710
Zrównoleglanie modelu	710
Zrównoleglanie danych	712
Uczenie wielkoskalowe za pomocą interfejsu strategii rozpraszania	718
Uczenie modelu za pomocą klastra TensorFlow	720
Realizowanie dużych grup zadań uczenia za pomocą usługi Vertex AI	723
Strojenie hiperparametrów w usłudze Vertex AI	725
Ćwiczenia	728
Dziękuję!	729
A Lista kontrolna projektu uczenia maszynowego	731
B Różniczkowanie automatyczne	736
C Specjalne struktury danych	744
D Grafy TensorFlow	750
Skorowidz	757

Drzewa decyzyjne

Drzewa decyzyjne (ang. *decision trees*) stanowią wszechstronne algorytmy uczenia maszynowego, służące zarówno do zadań klasyfikacji, jak i regresji, a nawet do operacji wielowyjściowych. Uzyskujemy za ich pomocą potężne modele zdolne do uczenia się wobec złożonych zbiorów danych. Na przykład w rozdziale 2. wyuczaliśmy model `DecisionTreeRegressor` wobec zbioru danych *California Housing* i uzyskaliśmy doskonałe wyniki (w rzeczywistości wręcz przetrenowaliśmy ten model).

Drzewa decyzyjne są również elementami składowymi losowych lasów (zob. rozdział 7.), czyli obecnie jednych z najlepszych algorytmów uczenia maszynowego.

W tym rozdziale zaczniemy od omówienia procesu uczenia drzew decyzyjnych, wyliczania prognoz i wizualizowania wyników. Następnie zajmiemy się algorytmem uczącym CART dostępnym w module Scikit-Learn, a także nauczymy się regularyzować drzewa i wykorzystywać je w zadaniach regresji. Na koniec przyjrzymy się niektórym ograniczeniom drzew decyzyjnych.

Uczenie i wizualizowanie drzewa decyzyjnego

Aby zrozumieć koncepcję drzew decyzyjnych, stwórzmy jedno i zobaczymy, w jaki sposób wylicza prognozy. Za pomocą poniższego kodu wyuczymy model `DecisionTreeClassifier` wobec zbioru danych *Iris* (zob. rozdział 4.):

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier

iris = load_iris(as_frame=True)
X_iris = iris.data[["petal length (cm)", "petal width (cm)"]].values
y_iris = iris.target

tree_clf = DecisionTreeClassifier(max_depth=2, random_state=42)
tree_clf.fit(X_iris, y_iris)
```

Możemy zwizualizować wyuczone drzewo decyzyjne, używając najpierw funkcji `export_graphviz()`, aby stworzyć plik definicji grafu, nazwany *iris_tree.dot*:

```
from sklearn.tree import export_graphviz

export_graphviz(
    tree_clf,
    out_file="iris_tree.dot",
    feature_names=["petal length (cm)", "petal width (cm)"],
```

```

class_names=iris.target_names,
rounded=True,
filled=True
)

```

Możesz następnie użyć metody `graphviz.Source.from_file()`, aby załadować i wyświetlić plik w notatniku Jupyter:

```

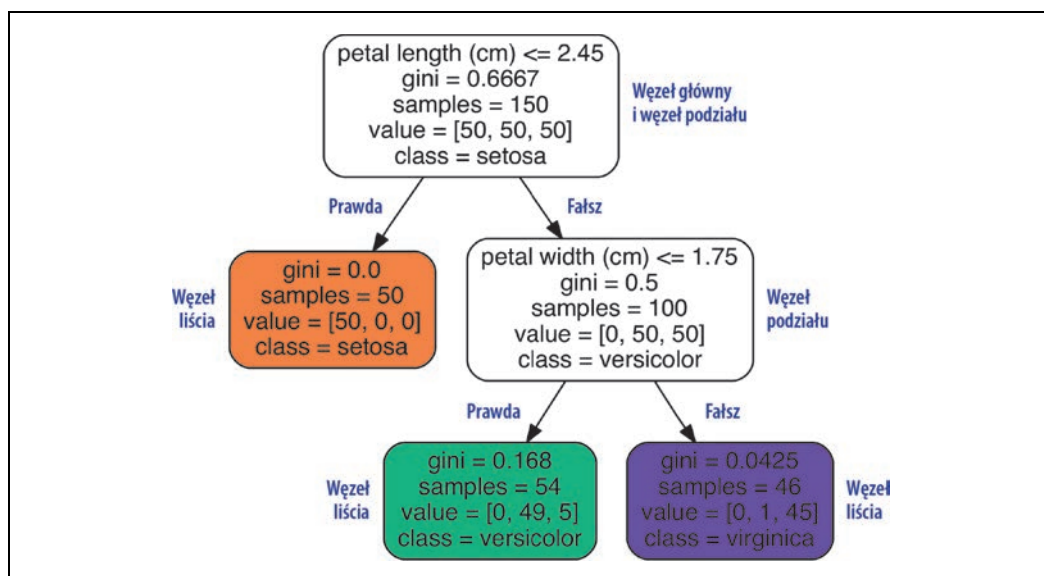
from graphviz import Source

Source.from_file("iris_tree.dot")

```

Graphviz (<https://graphviz.org/>) to objęty licencją otwartego oprogramowania pakiet do wizualizacji grafów. Zawiera on również narzędzie wiersza polecenia `dot` służące do konwertowania plików `.dot` na różne formaty, takie jak `.pdf` lub `.png`.

Twoje pierwsze drzewo decyzyjne zostało pokazane na rysunku 6.1.



Rysunek 6.1. Drzewo decyzyjne wyuczone na zbiorze danych Iris

Wyliczanie prognoz

Zastanówmy się, w jaki sposób drzewo widoczne na rysunku 6.1 przewiduje wyniki. Załóżmy, że znaleźliśmy kwiat kosaćca i chcemy go sklasyfikować na podstawie jego płatków. Zaczynamy od **węzła głównego** (ang. *root node*; wysokość 0 — węzeł na samej górze grafu): zadajemy w tym węźle pytanie, czy długość płatka jest mniejsza niż 2,45 cm. Jeśli jest, przechodzimy do lewego węzła potomnego (wysokość 1, po lewej). W takim przypadku docieramy do **liścia** (ang. *leaf node*; nie wychodzą z niego kolejne węzły potomne), zatem nie zadajemy tu już więcej pytań. Wystarczy teraz spojrzeć na przewidywaną klasę w tym węźle — drzewo decyzyjne przewiduje, że mamy do czynienia z gatunkiem *Iris setosa* (`class=setosa`).

Załóżmy teraz, że znajdujemy kolejny kwiat kosaćca, którego długość płatków przekracza teraz 2,45 cm. Ponownie zaczynamy od węzła głównego, ale tym razem kierujesz się ku prawemu węzłowi potomnemu (wysokość 1, po prawej). Nie jest to węzeł liścia, lecz **węzeł podziału** (ang. *split node*), dlatego zostaje postawione kolejne pytanie: czy szerokość płatków jest mniejsza niż 1,75 cm? Jeśli tak, to prawdopodobnie znaleziony kwiat należy do gatunku *Iris versicolor* (wysokość 2, po lewej). W przeciwnym razie możemy mieć do czynienia z gatunkiem *Iris virginica* (wysokość 2, po prawej). To naprawdę jest takie proste.



Jedną z licznych zalet drzew decyzyjnych jest fakt, że prawie nie wymagają one przygotowania danych. W istocie nie jest potrzebne skalowanie ani środkowanie cech.

Atrybut `sample` węzła zlicza liczbę wyznaczonych do niego próbek uczących. Na przykład 100 próbek ma długość płatków przekraczającą 2,45 cm (wysokość 1, po prawej), spośród których 54 mają szerokość płatków nieprzekraczającą 1,75 cm (wysokość 2, po lewej). Dzięki atrybutowi `value` dowiadujemy się, jak wiele przykładów uczących z każdej klasy przynależy do danego węzła: na przykład węzeł znajdujący się na dole po prawej stronie zawiera 0 próbek *Iris setosa*, 1 *Iris versicolor* i 45 *Iris virginica*. Z kolei atrybut `gini` stanowi miarę **zanieczyszczenia Giniego** (ang. *Gini impurity*) węzła: węzeł jest „czysty” ($gini=0$), jeżeli wszystkie znajdujące się w nim próbki uczące należą do tej samej klasy. Przykładem jest węzeł na wysokości 1 po lewej stronie, ponieważ zawiera on tylko przykłady uczące *Iris setosa*; jego zanieczyszczenie Giniego jest równe 0. Równanie 6.1 pokazuje, w jaki sposób algorytm wylicza zanieczyszczenie Giniego G_i dla i -tego węzła. Węzeł znajdujący się na wysokości 2 po lewej stronie ma zanieczyszczenie Giniego o wartości $1-(0:54)^2-(49:54)^2-(5:54)^2 \approx 0,168$.

Równanie 6.1. Zanieczyszczenie Giniego

$$G_i = 1 - \sum_{k=1}^n p_{i,k}^2$$

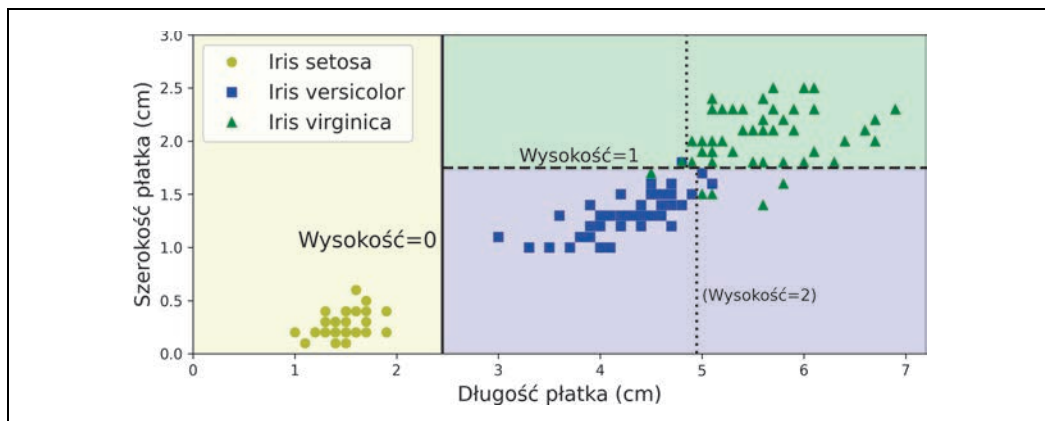
W tym równaniu:

- G_i — zanieczyszczenie Giniego w i -tym węźle.
- $p_{i,k}$ — współczynnik występowania klas k wśród próbek uczących w i -tym węźle.



Moduł Scikit-Learn wykorzystuje algorytm CART generujący wyłącznie **drzewa binarne** (ang. *binary trees*), czyli drzewa, których węzły podziału mają zawsze dokładnie dwoje potomków (tj. odpowiedziami na pytania są „tak” albo „nie”). Jednak inne algorytmy, takie jak ID3, mogą tworzyć drzewa decyzyjne, w których węzły mogą mieć większą liczbę potomków.

Na rysunku 6.2 widzimy granice decyzyjne drzewa decyzyjnego. Pogrubiona linia pionowa przedstawia granicę decyzyjną węzła głównego (wysokość 0): długość płatków = 2,45 cm. Obszar po lewej stronie jest czysty (występuje tu wyłącznie klasa *Iris setosa*), dlatego nie da się go bardziej podzielić. Jednakże obszar po prawej stronie pozostaje zanieczyszczony, zatem prawy węzeł na wysokości 1 rozdziela się na węzły potomne przy szerokości płatków = 1,75 cm (linia kreskowana). Wartość parametru `max_depth` wynosi 2, dlatego na takiej wysokości zatrzymuje się drzewo decyzyjne. Jeśli wyznaczymy wartość 3 parametru `max_depth`, to dwa węzły na wysokości 2 wprowadziłyby kolejną granicę decyzyjną (reprezentowaną przez dwie linie kropkowane).



Rysunek 6.2. Granice decyzyjne drzewa decyzyjnego



Struktura drzewa, zawierająca wszystkie informacje ukazane na rysunku 6.1, jest dostępna za pomocą atrybutu `tree_klasyfikatora`. Aby uzyskać więcej szczegółów, wpisz `help(tree_clf.tree_)`, natomiast w towarzyszącym książce notatniku Jupyter (znajdziesz przykład pod adresem <https://homl.info/colab3> lub <https://ftp.helion.pl/przyklady/uczem3.zip>).

Interpretacja modelu: „czarne skrzynki” i „białe skrzynki”

Algorytm drzew decyzyjnych jest bardzo intuicyjny i możemy go z łatwością interpretować. Tego typu modele są często nazywane **modelami „białej skrzynki”** (ang. *white box models*). Jak się przekonasz, algorytmy losowego lasu i sieci neuronowe należą do zgoła odmiennej kategorii, **modeli „czarnej skrzynki”** (ang. *black box models*). Ich przewidywania są znakomite i bez trudu możemy sprawdzić, jakie zostały przeprowadzone obliczenia do ich uzyskania; nie zmienia to faktu, że nieraz trudno wytłumaczyć prostymi słowami, skąd się te predykcje wzięły. Na przykład jeśli sieć neuronowa stwierdzi, że na danym zdjęciu znajduje się taka to a taka osoba, ciężko stwierdzić, jakie czynniki wpłynęły na ten rezultat: czy model rozpoznał daną osobę po oczach? Po ustach? Nosie? Butach? A może po kanapie, na której ta osoba siedzi? Z drugiej strony drzewo decyzyjne zawiera szereg przejrzystych i dobrze zdefiniowanych reguł klasyfikacji, za pomocą których w razie potrzeby można nawet ręcznie wyliczać prognozy (np. w przypadku klasyfikowania kwiatów). Dziedzina **wytłumaczalnego uczenia maszynowego** (ang. *interpretable ML*) dąży do tworzenia systemów uczenia maszynowego zdolnych do tłumaczenia podejmowanych decyzji w sposób zrozumiały dla człowieka. Jest to ważne w wielu domenach, może na przykład sprawiać, że system nie będzie podejmował niesprawiedliwych decyzji.

Szacowanie prawdopodobieństw przynależności do klas

Drzewo decyzyjne może również szacować prawdopodobieństwo przynależności danej próbki do określonej klasy k . Najpierw jest wyszukiwany liść, w którym dana próbka się znajduje, po czym zostaje zwrócony odsetek przykładów uczących w tym węźle należących do klasy k . Załóżmy na przykład, że znaleźliśmy kwiat, którego płatki mają 5 cm długości i 1,5 cm szerokości. Próbka symbolizująca

ten kwiat znajduje się w lewym liściu na wysokości 2 drzewa, zatem algorytm wyliczy następujące prawdopodobieństwa: 0% przynależności do gatunku *Iris setosa* (0/54), 90,7% dla *Iris versicolor* (49/54) i 9,3% dla *Iris virginica* (5/54). Zostanie dla tego kwiatu przewidziana klasa *Iris versicolor* (klasa 1.), ponieważ uzyskała ona największe prawdopodobieństwo. Sprawdźmy:

```
>>> tree_clf.predict_proba([[5, 1.5]]).round(3)
array([[0. , 0.907, 0.093]])
>>> tree_clf.predict([[5, 1.5]])
array([1])
```

Doskonale! Zwróć uwagę, że szacowane prawdopodobieństwa będą identyczne w dowolnym obszarze prawego dolnego prostokąta widocznym na rysunku 6.2 — na przykład dla płatków o długości 6 cm i szerokości 1,5 cm (nawet jeśli jest dla nas oczywiste, że w tym przypadku płatki te należałyby najprawdopodobniej do gatunku *Iris virginica*).

Algorytm uczący CART

Moduł Scikit-Learn wykorzystuje algorytm **drzew klasyfikacyjnych i regresyjnych** (ang. *classification and regression tree* — CART) do uczenia drzew decyzyjnych (ich „wzrostu”). Algorytm rozdziela najpierw dane uczące na dwa podzbiory przy użyciu pojedynczej cechy k i progu t_k (np. „długość płatka $\leq 2,45$ cm”). W jaki sposób są dobierane wartości k i t_k ? Wyszukiwana jest para parametrów (k, t_k) generująca najczystsze podzbiory, ważone pod względem rozmiaru. Funkcja kosztu, jaką algorytm stara się minimalizować, została przedstawiona w równaniu 6.2.

Równanie 6.2. Funkcja kosztu algorytmu CART używana w zadaniach klasyfikacji

$$J(k, t_k) = \frac{m_{\text{lewy}}}{m} G_{\text{lewy}} + \frac{m_{\text{prawy}}}{m} G_{\text{prawy}}$$

gdzie $\begin{cases} G_{\text{lewy/prawy}} & \text{mierzy zanieczyszczenie lewego/prawego podzbioru} \\ m_{\text{lewy/prawy}} & \text{stanowi liczbę próbek w lewym/prawym podzbiorze} \end{cases}$

Gdy algorytm CART rozdzieli zestaw danych uczących na dwa podzbiory, są one dalej dzielone na tej samej zasadzie aż do osiągnięcia maksymalnej wysokości (zdefiniowanej za pomocą hiperparametru `max_depth`) lub jeśli nie uda się określić takiego podziału, który zmniejszyłoby zanieczyszczenie. Kilka innych hiperparametrów (omówimy je niebawem) określa dodatkowe warunki zatrzymania budowy drzewa: `min_samples_split`, `min_samples_leaf`, `min_weight_fraction_leaf` i `max_leaf_nodes`.



Jak widać, CART jest **algorytmem zachłannym** (ang. *greedy algorithm*): zachłannie poszukuje optymalnego podziału na najniższym poziomie, a następnie na każdym kolejnym powtarza tę czynność. Nie sprawdza on, czy dany podział będzie prowadził kilka poziomów wyżej do najmniejszego możliwego zanieczyszczenia. Algorytmy zachłanne często dają dobre wyniki, nie muszą być one jednak optymalne.

Niestety szukanie optymalnego drzewa jest klasyfikowane jako **problem NP-zupełny**¹. Należy poświęcić $O(\exp(m))$ czasu na jego rozwiązanie, przez co problem staje się trudny nawet dla małych zbiorów uczących. Dlatego musi nam wystarczyć poszukiwanie „w miarę dobrego” rozwiązania podczas trenowania drzew decyzyjnych.

Złożoność obliczeniowa

Wyliczanie prognoz wymaga poruszania się po drzewie decyzyjnym od węzła głównego aż do liścia. Generalnie drzewa decyzyjne są w miarę zrównoważone, zatem poruszanie się po drzewie decyzyjnym wymaga jedynie odwiedzenia mniej więcej $O(\log_2(m))$ węzłów, gdzie zapis \log_2 oznacza **logarytm binarny (dwójkowy)** (ang. *binary logarithm*) z m , równy $\log_2(m) = \log(m)/\log(2)$. W każdym węźle wymagane jest jedynie sprawdzenie wartości jednej cechy, tak więc całkowita złożoność prognoz to $O(\log_2(m))$, niezależnie od liczby cech. Wyliczanie przewidywań jest zatem bardzo szybkie nawet w przypadku bardzo dużych zbiorów uczących.

Algorytm uczący porównuje wszystkie cechy (mniej, jeśli wyznaczymy wartość parametru `max_features`) ze wszystkimi próbkami znajdującymi się w danym węźle. Porównanie wszystkich cech we wszystkich przykładach w każdym węźle skutkuje złożonością uczenia na poziomie $O(n \times m \log_2(m))$.

Wskaźnik Giniego czy entropia?

Domyślnie klasa `DecisionTreeClassifier` wykorzystuje wskaźnik zanieczyszczenia Giniego, ale możemy wybrać również **entropię** jako miarę zanieczyszczenia, wprowadzając wartość entropii dla hiperparametru `criterion`. Pojęcie entropii wywodzi się z termodynamiki, gdzie służy do opisu miary nieuporządkowania cząsteczek: gdy cząsteczki są nieruchome i uporządkowane w przestrzeni, to wartość entropii wynosi 0. Koncepcja entropii wkradła się również w różne inne dziedziny naukowe, w tym również do **teorii informacji** Shannona, gdzie służy do pomiaru średniej zawartości informacji w wiadomości, o czym przekonaliśmy się w rozdziale 4. Entropia wynosi 0, gdy wszystkie wiadomości są takie same. W świecie uczenia maszynowego za pomocą entropii często mierzy się zanieczyszczenie: entropia zbioru jest równa zero, gdy mieszczą się w nim wyłącznie próbki należące do jednej klasy. Równanie 6.3 ukazuje definicję entropii i -tego węzła. Przykładowo entropia lewego

węzła na wysokości 2 (rysunek 6.1) wynosi: $-\frac{49}{54} \log_2\left(\frac{49}{54}\right) - \frac{5}{54} \log_2\left(\frac{5}{54}\right) \approx 0,445$.

¹ P stanowi zbiór problemów, jakie mogą zostać rozwiązane w wielomianowym czasie (tj. wielomianowym od rozmiaru zestawu danych). Zbiór NP zawiera problemy, których rozwiązania mogą zostać zweryfikowane w wielomianowym czasie. Problem NP-trudny to każdy problem, do którego problem NP może zostać zredukowany w czasie wielomianowym. Do problemu NP-zupełnego zaliczają się problemy NP i NP-trudne. Jednym z głównych pytań natury matematycznej, niemających do tej pory odpowiedzi, pozostaje, czy $P = NP$. Jeżeli $P \neq NP$ (co jest bardzo prawdopodobne), to nigdy nie zostanie odkryty wielomianowy algorytm dla dowolnego problemu NP-zupełnego (może zmienić się to w przypadku komputerów kwantowych).

Równanie 6.3. Entropia

$$H_i = - \sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^n p_{i,k} \log_2(p_{i,k})$$

Powinniśmy zatem korzystać ze wskaźnika Giniego czy z entropii? Prawdę mówiąc, w większości przypadków nie ma to większego znaczenia, gdyż uzyskujemy za ich pomocą podobne drzewa. Wskaźnik Giniego jest obliczany nieco szybciej, więc stanowi dobrą wartość domyślną. Jednak w sytuacjach, w których obydwie wskaźniki się różnią, wskaźnikowi Giniego zdarza się izolować najczęściej występującą klasę w osobnej gałęzi, natomiast entropia generuje nieco bardziej zrównoważone drzewa².

Hiperparametry regularyzacyjne

Algorytm drzew decyzyjnych nie przyjmuje niemal żadnych założeń dotyczących danych uczących (w przeciwieństwie na przykład do modeli liniowych, które zakładają, że operują na danych liniowych). Jeżeli nie nałożymy żadnych ograniczeń, struktura drzewa samoistnie dostosuje się do danych uczących i robi to prawie idealnie, niemal z pewnością ulegając przetrenowaniu. Jest to tak zwany **model nieparametryczny** (ang. *nonparametric model*) — nie dlatego, że nie zawiera parametrów (często ma ich znaczną liczbę), lecz dlatego, że liczba tych parametrów nie jest ustalana przed rozpoczęciem trenowania, zatem struktura modelu jest w stanie ściśle dopasować się do danych. Z drugiej strony mamy do czynienia z **modelami parametrycznymi** (ang. *parametric models*; reprezentowanymi m.in. przez model liniowy), w których występuje ustalona liczba parametrów, cechuje je więc ograniczenie stopni swobody, dzięki czemu zmniejszamy ryzyko przetrenowania (ale jednocześnie zwiększamy możliwość niedotrenowania).

Aby uniknąć przetrenowania modelu, musimy ograniczyć swobodę algorytmu drzewa decyzyjnego podczas uczenia. Wiemy już, że ten proces nosi nazwę regularyzacji. Hiperparametry regularyzacyjne zależą od stosowanego algorytmu, zazwyczaj jednak możemy ograniczyć przynajmniej maksymalną wysokość drzewa. W module Scikit-Learn odpowiada za to hiperparametr `max_depth`. Jego wartość domyślna, `None`, powoduje tworzenie drzew o nieograniczonej wysokości. Podanie wartości liczbowej w hiperparametrze `max_depth` spowoduje regularyzację modelu i zmniejszenie ryzyka przetrenowania.

Klasa `DecisionTreeClassifier` zawiera także kilka innych parametrów ograniczających kształt drzewa decyzyjnego:

`max_features`

Maksymalna liczba cech używanych do dzielenia w każdym węźle.

`max_leaf_nodes`

Maksymalna liczba liści.

`min_samples_split`

Minimalna liczba próbek, jakie muszą się znajdować w węźle, aby został podzielony.

² Więcej szczegółów znajdziesz w interesującej analizie (<https://homl.info/19>), której autorem jest Sebastian Raschka.

`min_samples_leaf`

Minimalna liczba próbek, jakie muszą się znajdować w liściu.

`min_weight_fraction_leaf`

Taki sam jak parametr `min_samples_leaf`, tu jednak wartością jest ułamek całkowitej liczby ważonych próbek.

Zwiększanie wartości hiperparametrów `min_*` lub zmniejszanie `max_*` powoduje regularyzację modelu.



Inne algorytmy najpierw trenują model drzewa decyzyjnego bez żadnych ograniczeń, a następnie **przycinają** (ang. *prune*; usuwają) niepotrzebne liście. Węzeł zawierający same liście jest uznawany za niepotrzebny, jeśli zapewniana przez niego redukcja zanieczyszczenia okazuje się **nieistotna statystycznie**. Standardowe testy statystyczne, takie jak test chi kwadrat (test χ^2), służą do oszacowania prawdopodobieństwa, że zmniejszenie zanieczyszczenia stanowi wyłącznie wynik przypadku (jest to tzw. **hipoteza zerowa**). Jeżeli to prawdopodobieństwo, zwane **p-wartością**, przekroczy pewien określony próg (zazwyczaj 5% — definiujemy go za pomocą hiperparametru), to węzeł jest uznawany za niepotrzebny, a jego liście zostają usunięte. Proces przycinania trwa, dopóki nie zostaną usunięte wszystkie niepotrzebne węzły.

Przetestujmy regularyzację na zestawie danych sierpowatych, znanym nam już z rozdziału 5. Wytrenujemy jedno drzewo decyzyjne bez regularyzacji, a drugie z parametrem `min_samples_leaf=5`. Kod znajduje się poniżej; rysunek 6.3 ukazuje granice decyzyjne każdego drzewa:

```
from sklearn.datasets import make_moons
```

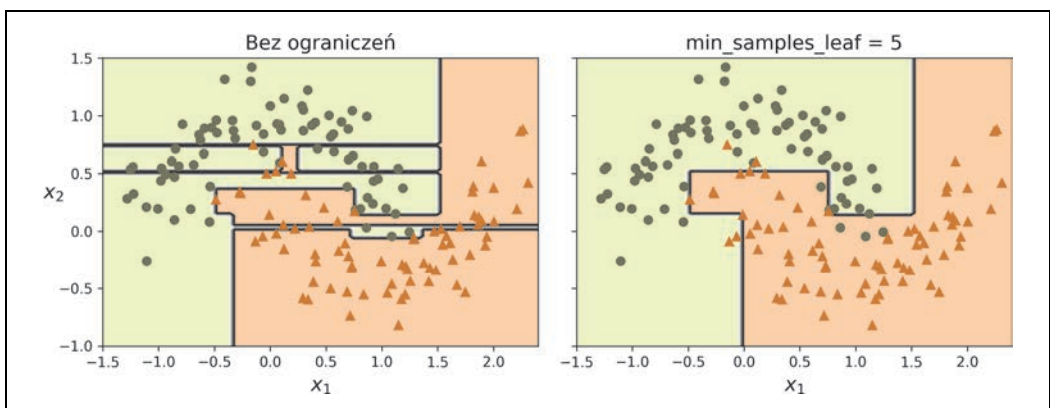
```
X_moons, y_moons = make_moons(n_samples=150, noise=0.2, random_state=42)
```

```
tree_clf1 = DecisionTreeClassifier(random_state=42)
```

```
tree_clf2 = DecisionTreeClassifier(min_samples_leaf=5, random_state=42)
```

```
tree_clf1.fit(X_moons, y_moons)
```

```
tree_clf2.fit(X_moons, y_moons)
```



Rysunek 6.3. Granice decyzyjne nieregularyzowanego (po lewej) i regularyzowanego (po prawej) drzewa

Widoczny po lewej stronie nieregularyzowany model jest wyraźnie przetrenowany, regularyzowany model z prawego wykresu prawdopodobnie będzie sobie lepiej radził z uogólnianiem. Możemy to sprawdzić, oceniając obydwie drzewa na zbiorze testowym wygenerowanym za pomocą innego ziarna losowości:

```
>>> X_moons_test, y_moons_test = make_moons(n_samples=1000, noise=0.2,
...
...
...
>>> tree_clf1.score(X_moons_test, y_moons_test)
0.898
>>> tree_clf2.score(X_moons_test, y_moons_test)
0.92
```

Rzeczywiście, drugie drzewo uzyskuje większą dokładność na zbiorze testowym.

Regresja

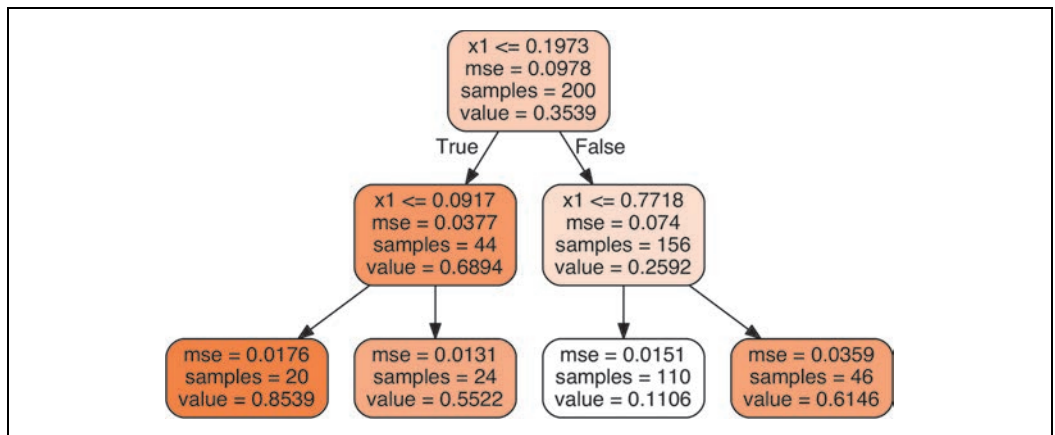
Drzewa decyzyjne mogą również wykonywać zadania regresyjne. Stwórzmy takie drzewo regresyjne za pomocą klasy `DecisionTreeRegressor`; wyuczmy je wobec zaszumionego, kwadratowego zbioru danych, a jego maksymalną wysokość ustalmy na poziomie `max_depth=2`:

```
import numpy as np
from sklearn.tree import DecisionTreeRegressor

np.random.seed(42)
X_quad = np.random.rand(200, 1) - 0.5 # pojedyncza losowa cecha wejściowa
y_quad = X_quad ** 2 + 0.025 * np.random.randn(200, 1)

tree_reg = DecisionTreeRegressor(max_depth=2, random_state=42)
tree_reg.fit(X_quad, y_quad)
```

Rysunek 6.4 przedstawia wygenerowane drzewo decyzyjne.

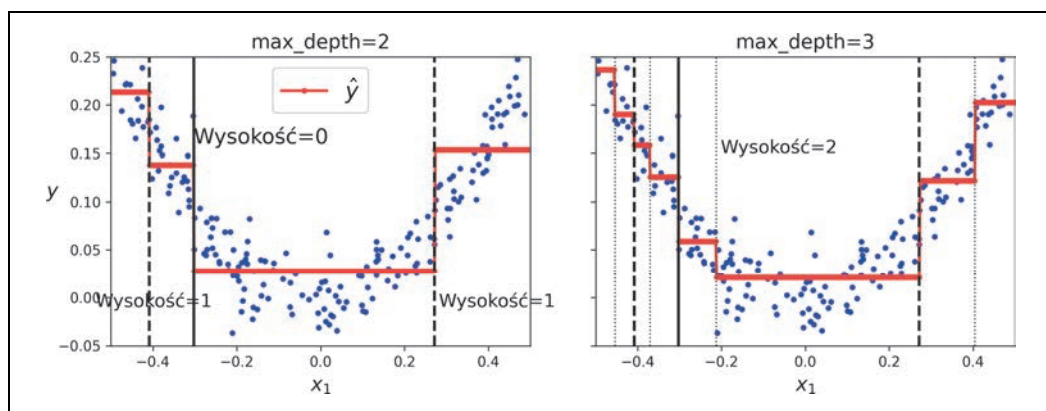


Rysunek 6.4. Regresyjne drzewo decyzyjne

Model ten bardzo przypomina stworzone przez nas wcześniej drzewo klasyfikacyjne. Główna różnica polega na tym, że w każdym węźle jest prognozowana nie klasa, lecz wartość. Załóżmy przykładowo, że chcemy wyliczyć prognozę dla nowej próbki $x_1 = 0,2$. Węzeł główny pyta, czy $x_1 \leq 0,197$.

Otrzymaliśmy inną wartość, więc algorytm przechodzi do prawego węzła potomnego, gdzie jest sprawdzane, czy $x_1 \leq 0,772$. Tak jest w istocie, zostaje zatem wybrany lewy węzeł potomny. Jest to liść przewidujący $value=0.111$. Predykcja ta stanowi średnią wartość docelową 110 próbek uczących powiązanych z tym liściem, a w wyniku tej prognozy otrzymujemy wartość błędu MSE wynoszącą 0,015 dla tych 110 próbek.

Przewidywania tego modelu zostały zaprezentowane po lewej stronie na rysunku 6.5. Jeżeli wyznaczymy hiperparametr $max_depth=3$, prognozy będą wyglądały tak jak na prawym wykresie. Zwróć uwagę, że prognozowana wartość dla każdego obszaru stanowi zawsze średnią wartość docelową próbek znajdujących się na tym obszarze. Algorytm rozdziela każdy obszar w taki sposób, żeby jak najwięcej próbek uczących znajdowało się możliwie blisko tej przewidywanej wartości.



Rysunek 6.5. Prognozy dwóch modeli regresyjnych drzew decyzyjnych

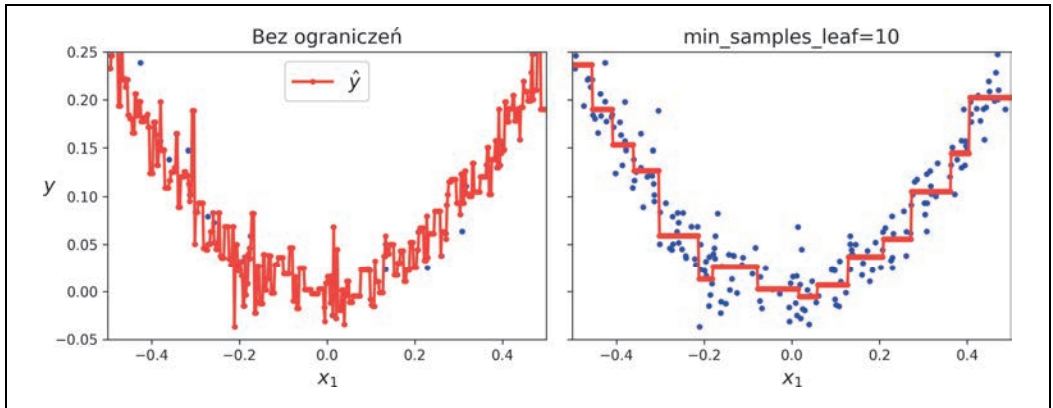
Algorytm CART działa niemal identycznie jak we wcześniejszym opisie, teraz jednak stara się rozdzielać zbiór danych uczących w sposób minimalizujący błąd MSE (a nie zanieczyszczenie). Równanie 6.4 zawiera funkcję kosztu minimalizowaną przez ten algorytm.

Równanie 6.4. Funkcja kosztu CART stosowana w regresji

$$J(k, t_k) = \frac{m_{\text{lewy}}}{m} MSE_{\text{lewy}} + \frac{m_{\text{prawy}}}{m} MSE_{\text{prawy}}$$

$$\text{gdzie } \begin{cases} MSE_{\text{węzeł}} = \sum_{i \in \text{węzeł}} (\hat{y}_{\text{węzeł}} - y^{(i)})^2 \\ \hat{y}_{\text{węzeł}} = \frac{1}{m_{\text{węzeł}}} \sum_{i \in \text{węzeł}} y^{(i)} \end{cases}$$

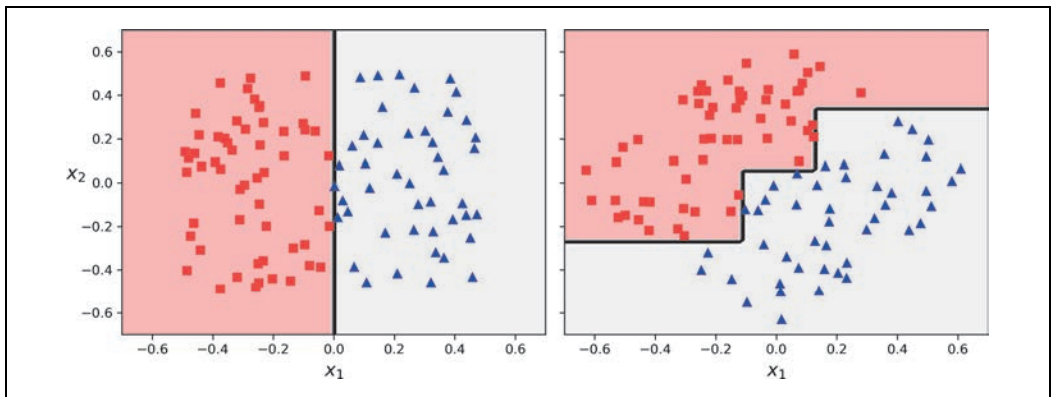
Podobnie jak w przypadku zadań klasyfikacji, drzewa decyzyjne są podatne na przetrenowanie w modelach regresji. Jeśli nie będziemy stosować żadnej formy regularyzacji (np. korzystając wyłącznie z domyślnych wartości hiperparametrów), uzyskamy prognozy widoczne na lewym wykresie rysunku 6.6. Jak widać, model jest znacznie przetrenowany. Wystarczy jednak, że wyznaczymy hiperparametr $min_samples_leaf=10$, aby uzyskać znacznie lepszy model, widoczny na prawym wykresie.



Rysunek 6.6. Przewidywania nieregularyzowanego (po lewej) i regularyzowanego (po prawej) drzewa regresyjnego

Wrażliwość na orientację osi

Mam nadzieję, że dostrzegasz już olbrzymi potencjał drzew decyzyjnych: są względnie łatwe do zrozumienia i interpretacji, przystępne, wszechstronne i wydajne. Mają jednak kilka ograniczeń. Przede wszystkim, jak pewnie zdążyłeś zauważyć, algorytm drzewa decyzyjnego uwielbia ortogonalne granice decyzyjne (wszystkie podziały są przeprowadzane prostopadłe do osi odciętych), przez co model ten jest wrażliwy na orientację danych. Na przykład rysunek 6.7 przedstawia prosty, liniowo rozdzielny zestaw danych: na lewym wykresie próbki są z łatwością rozdzielane, natomiast na prawym wykresie te same dane zostały obrócone o 45° i granica decyzyjna jest tu niepotrzebnie skomplikowana. Pomimo że drzewo decyzyjne zostało perfekcyjnie dopasowane do danych, istnieje duże ryzyko, że model ten nie będzie zbyt dobrze przeprowadzał uogólniania.



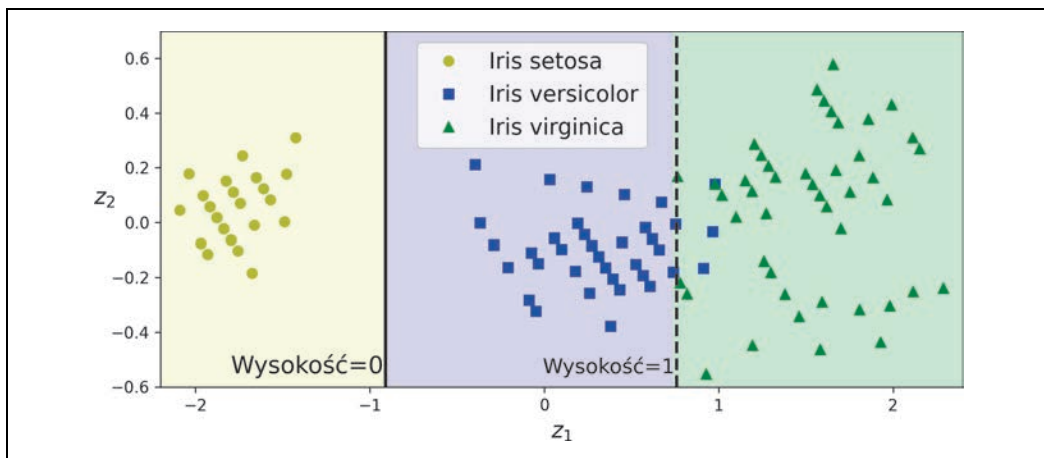
Rysunek 6.7. Wrażliwość na rotację zbioru uczącego

Jednym ze sposobów ograniczenia tego problemu jest skalowanie danych, a następnie wprowadzenie algorytmu analizy głównych składowych (PCA). W rozdziale 8. zajmiemy się PCA, na razie jednak musisz tylko wiedzieć, że rozwiązanie to obraca dane w sposób zmniejszający korelację między cechami, co często (nie zawsze) jest korzystne dla drzew decyzyjnych.

Stwórzmy mały potok skalujący dane i obracający je za pomocą PCA, a następnie wytrenujemy na nich model `DecisionTreeClassifier`. Rysunek 6.8 prezentuje granice decyzyjne tego drzewa: jak widać, dzięki rotacji możliwe jest całkiem skuteczne dopasowanie zestawu danych wyłącznie za pomocą jednej cechy, z_1 , stanowiącej funkcję liniową pierwotnej długości i szerokości płatk. Kod znajduje się poniżej:

```
from sklearn.decomposition import PCA
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

pca_pipeline = make_pipeline(StandardScaler(), PCA())
X_iris_rotated = pca_pipeline.fit_transform(X_iris)
tree_clf_pca = DecisionTreeClassifier(max_depth=2, random_state=42)
tree_clf_pca.fit(X_iris_rotated, y_iris)
```



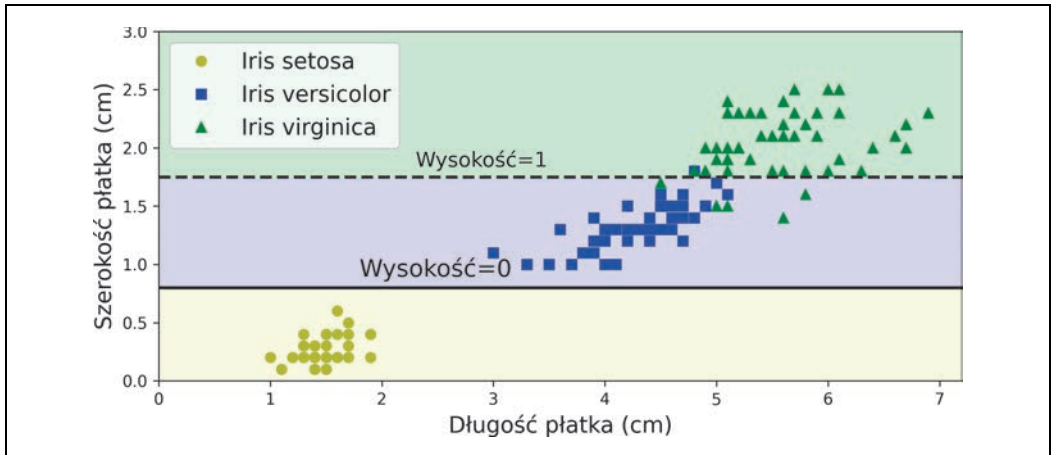
Rysunek 6.8. Granice decyzyjne drzewa na przeskalowanym i obróconym metodą PCA zestawie danych Iris

Drzewa decyzyjne mają znaczną wariację

W bardziej ogólnym ujęciu główny problem drzew decyzyjnych polega na ich całkiem dużej wariacji: małe zmiany w hiperparametrach lub danych mogą generować bardzo różne modele. W istocie skoro algorytm uczący używany przez moduł Scikit-Learn jest stochastyczny³ (losowo dobiera zestaw cech do oceniania każdego węzła), to nawet powtórne wytrenowanie identycznego drzewa decyzyjnego na dokładnie takich samych danych może wygenerować całkiem odmienny model, co widać na rysunku 6.9 (chyba że ustawisz hiperparametr `random_state`). Jak widać, różni się on znacznie od poprzedniego drzewa decyzyjnego (z rysunku 6.2).

Na szczęście dzięki uśrednianiu prognoz uzyskiwanych z wielu drzew jest możliwe znaczne redukcje wariacji. Taki zespół drzew nazywany jest **lasem losowym** i stanowi on jeden z najpotężniejszych współczesnych rodzajów modeli, o czym przekonasz się w następnym rozdziale.

³ Losowo dobiera zestaw cech do oceniania każdego węzła.



Rysunek 6.9. Powtórne uczenie tego samego modelu na tych samych danych może prowadzić do zupełnie odmiennego modelu

Ćwiczenia

1. Jaką przybliżoną wysokość osiągnie drzewo decyzyjne uczone (bez ograniczeń) wobec zestawu danych składającego się z miliona próbek?
2. Czy zanieczyszczenie Giniego węzła podrzędnego jest zazwyczaj większe, czy mniejsze od tego wskaźnika w węzle nadrzędnym? Czy jest on *zazwyczaj*, czy też *zawsze* mniejszy/większy?
3. Czy jeżeli drzewo decyzyjne ulega przetrenowaniu, warto próbować zmniejszyć wartość hiperparametru `max_depth`?
4. Czy dobrym pomysłem jest próba skalowania cech wejściowych, jeżeli drzewo decyzyjne wykazuje oznaki niedotrenowania?
5. Jeżeli wyuczenie drzewa decyzyjnego wobec zbioru danych składającego się z miliona próbek zajmuje godzinę, w przybliżeniu jak wiele czasu należy poświęcić na wytrenowanie drzewa przy użyciu zestawu składającego się z 10 milionów przykładów? Podpowiedź: weź pod uwagę złożoność obliczeniową algorytmów z rodziny CART.
6. Jeżeli wytrenowanie drzewa decyzyjnego na określonym zestawie danych zajmuje jedną godzinę, to jak długo w przybliżeniu zajmie ten proces po podwojeniu liczby cech?
7. Wytrenuj i dostrój model drzewa decyzyjnego wobec danych sierpowatych, korzystając z następujących kroków:
 - a. Stwórz zbiór danych za pomocą funkcji `make_moons(n_samples=10000, noise=0.4)`.
 - b. Rozdziel uzyskany zestaw danych na podzbiory uczący i testowy przy użyciu metody `train_test_split()`.
 - c. Wykorzystaj przeszukiwanie siatki wraz ze sprawdzianem krzyżowym (przyda się klasa `GridSearchCV`), aby znaleźć dobre wartości hiperparametrów dla klasy `DecisionTreeClassifier`. Podpowiedź: wypróbuj różne wartości hiperparametru `max_leaf_nodes`.

- d. Wytrenuj ten model wobec pełnego zbioru uczącego, korzystając z uzyskanych wartości hiperparametrów, a następnie sprawdź wydajność modelu wobec zestawu testowego. Powinieneś uzyskać wyniki rzędu 85 – 87%.
8. Posadź las za pomocą następujących kroków:
 - a. Korzystając z poprzedniego ćwiczenia, wygeneruj 1000 podzbiorów zestawu uczącego, każdy zawierający 100 losowo dobranych próbek. Podpowiedź: możesz w tym celu skorzystać z klasy `ShuffleSplit`.
 - b. Wytrenuj po jednym drzewie decyzyjnym dla każdego podzbioru, korzystając z najlepszych wartości hiperparametrów odkrytych w poprzednim ćwiczeniu. Oceń wydajność tego tysiąca drzew decyzyjnych na zestawie testowym. Drzewa te zostały wyuczone przy użyciu mniejszych zbiorów danych, dlatego prawdopodobnie będą miały gorszą dokładność od pierwotnego drzewa decyzyjnego, oscylującą w granicach 80%.
 - c. Czas na odrobinę magii. Dla każdej próbki zbioru testowego wygeneruj prognozy wyliczane przez wszystkie 1000 drzew i zachowaj jedynie najczęściej powtarzający się wynik (możesz użyć do tego metody `mode()`, stanowiącej część modułu `SciPy`). Uzyskujesz w ten sposób **prognozy metodą głosowania większościowego** dla zbioru testowego.
 - d. Oceń te przewidywania wobec zbioru testowego: powinieneś uzyskać nieco większą dokładność niż w przypadku pierwotnego modelu (wyższą o 0,5% – 1,5%). Gratulacje, właśnie wytrenowałeś swój pierwszy klasyfikator losowego lasu!

Rozwiązania tych ćwiczeń znajdziesz na końcu notatnika Jupyter zawierającego kod tego rozdziału, dostępnego pod adresem <https://ftp.helion.pl/przyklady/uczem3.zip> (po polsku) lub <https://homl.info/colab3> (po angielsku).

A

- A2C, Advantage Actor-Critic, 675
- A3C, Asynchronous Advantage Actor-Critic, 675
- ACF, autocorrelation function, 525
- agregacja, bagging, 221, 222
- aktualizacje
 - asynchroniczne, asynchronous updates, 715
 - synchroniczne, synchronous updates, 714
- aktualizowanie wag, 304
- algebra liniowa przyspieszona, XLA, 418
- algorytm
 - A2C, 675
 - A3C, 675
 - AdaBoost, 227
 - AdaGrad, 371
 - Adam, 373
 - AdaMax, 374
 - AdamW, 375
 - aktor – krytyk, 658, 675
 - AllReduce, 713
 - AlphaGo, 674
 - BIRCH, 282
 - CART, 205, 207
 - centroidów, 262, 264
 - granice, 273
 - granice decyzyjne, 266
 - mechanizm działania, 267
 - metody inicjalizowania centroidów, 268
 - optymalna liczba skupień, 271
 - przyspieszony, 269
 - z minigrupami, 269
 - DBSCAN, 279
 - DQN, 670
 - Fast-MCD, 292
 - Isomap, 258
 - iteracji Q-wartości, 665
 - k-najbliższych sąsiadów, k-nearest neighbors, 48
 - k-średnich, 264
 - LLE, 256, 257
 - Mean-shift, 282
 - Nadam, 375
 - NEAT, 648
 - Nesterova, 370
 - oczekiwania – maksymalizacji, 284
 - optymalizacji momentum, 369
 - PCA, 251
 - POET, 677
 - PPO, 676
 - przeszukiwania drzewa metodą Monte Carlo, 674
 - Q-uczenia, 664–666
 - regresji, 48
 - RMSProp, 373
 - SAC, 676
 - stochastycznego spadku wzdłuż gradientu, 159
 - uczenia metodą różnic czasowych, 663
 - walczącej sieci DQN, 673
 - węgierski, 506
 - zachłanny, greedy algorithm, 207
- algorytmy
 - bez strategii, 665
 - genetyczne, genetic algorithms, 647
 - przyrostowe PCA, 252
 - REINFORCE, 655
 - RN, 674
 - uczące model regresji liniowej, 161
 - uczenia maszynowego, 87
 - wizualizujące, 36
 - ze strategią, 665

analiza

- błędów, 136
- dyskryminacyjna liniowa, LDA, 258
- funkcji autokorelacji, ACF, 525
- głównych składowych, PCA, 245, 604
 - losowa, 252
 - przyrostowa, 252
- grafów, 751
- najlepszych modeli, 112
- opinii, 558
- przepływu sterowania, 754
- składni obiektów, 439
- skupień, 35
 - aglomeracyjna, 282
 - algorytm BIRCH, 282
 - algorytm centroidów, 262
 - algorytm DBSCAN, 279
 - algorytm Mean-shift, 282
 - grupowanie, 261
 - hierarchiczna, 35
 - propagacja podobieństwa, 283
 - w segmentacji obrazu, 274
 - w uczeniu półnadzorowanym, 276
 - widmowa, 283
 - zastosowania, 263

ANN, Artificial Neural Networks, 297

anomalia, anomaly, 261

AP, Average Precision, 505

aplikacja internetowa progresywna, PWA, 700

architektura

- sieci
 - AlexNet, 476
 - GoogLeNet, 479, 481
 - LeNet-5, 475
 - ResNet, 484
 - SENet, 487
 - Xception, 486
 - YOLO, 503
- StyleGAN, 634
 - sieć mapująca, mapping network, 634
 - sieć syntetyzująca, synthesis network, 635
- transformatora, 578, 579

architektury sieci spłotowych, 473, 489

ARMA, autoregressive moving average, 523

asymetria uczenia/eksploatacji, training/serving skew, 425

atrybuty, attributes, 35

- kategorialne, 90
- sztuczne, dummy attributes, 91
- tekstowe, 90

AUC, area under the curve, 131

AutoGraph, 419, 754

autokodery, 601

- generatywne, 618
- liniowe niedopełnione, 604
- probabilistyczne, probabilistic autoencoders, 618
- przepełnione, overcomplete autoencoders, 614
- rzadkie, 615
- sieć generatywna, generative network, 603
- sieć rozpoznawania, recognition error, 603
- splotowe, convolutional autoencoders, 613
- stosowe, stacked autoencoders, 605
 - implementacja, 606
 - odszumiające, stacked denoising autoencoders, 614
 - uczenie wstępne nienadzorowane, 609
- uczenie pojedynczo, 611
- wariacyjne, variational autoencoders, 618, 622

autoregresywny zintegrowany model średniej kroczącej, 523

B

badanie, study, 725

bayesowskie modele mieszane, 291

biblioteka

- CUDA, 703
- Keras Tuner, 337
- NCCL, 719
- NumPy, 394, 396
- TensorFlow.js, 700
- Transformers, 596

blender, 235

blob, 693

blok

- rezydualny, residual block, 408
- SE, 487, 488

błąd

- braku odpowiedzi, nonresponse bias, 52
- nieredukowalny, irreducible error, 167
- rekonstrukcji, reconstruction error, 251
- uogólniania, 56

błędy

- analizowanie, 136

BPE, byte pair encoding, 559

BPTT, backpropagation through time, 517

bramka
 wejściowa, input gate, 541
 wyjściowa, output gate, 541
 zapominająca, 542
bramkowane jednostki aktywacji, 546
bufor odtwarzania, replay buffer, 667
bufory protokołów, buffer protocol, 436
 obsługa list, 440
 w module TensorFlow, 438

C

CART, classification and regression tree, 207
cechy, 35
 kategorialne, 447
 podobieństwa, 190
 przekształcanie, 94
 skalowanie, 94
CGAN, conditional GAN, 631
charakterystyka robocza odbiornika, ROC, 130
CLI, command-line interface, 690
CNN, convolutional neural networks, 458
CSV, comma-separated values, 72
CUDA, 703
człon wygładzający, smoothing term, 359

D

dane
 geograficzne, 81
 nominalne, inlier, 262
 oczyszczanie, 88
 pobieranie, 72
 rzeczywiste, 61
 uczące, 28
DBSCAN, 279
DCGAN, deep convolutional GAN, 628
DDQN, Dueling DQN, 673
dekoder, decoder, 516, 603
dekodowanie zachłanne, greedy decoding, 554
destylacja, distillation, 590
diagram Woronoja, 266
DNN, Deep Neural Network, 306
dobór
 cechy, feature selection, 52
 modelu, 46, 56
Docker
 uruchamianie TF Serving, 683
dogenerowanie danych, data augmentation, 140, 477

dokładność, accuracy, 28
dominanta, 222
dostrajanie hiperparametrów, 337
DQN, deep Q-network, 666
dryf danych, data drift, 41
drzewa
 binarne, binary trees, 205
 decyzyjne, decision trees, 203
 uczenie, 203
 wariancja, 214
 wizualizowanie, 203
 klasyfikacyjne, 207
 regresyjne, 207, 211
 decyzyjne, 211
 wzmocniane gradientowo, GBRT, 231
dylemat poszukiwania/wykorzystywania, 652
Dysk Google, 70
dyskryminator, discriminator, 602, 623
dywergencja Kullbacka-Leiblera, 182, 616

E

efektywność parametryczna, parameter efficiency, 342
eksperyment A/B, 679
eksploracja danych, data mining, 30
ekwiwariancja, equivariance, 470
ELMo, 563
entropia, 208
 krzyżowa, cross entropy, 181, 182
epoka, epoch, 156, 158, 173
estymatory Scikit-Learn, 99
etykiety, labels, 34

F

faza rozgrzewki, warmup phase, 715
FCN, Fully Convolutional Network, 501
filtr, 462
 rozcieńczony, diluted filter, 509
 Kalmana, 506
FNN, Feedforward Neural Network, 306
format
 SavedModel, 680, 686
 TFRecord, 435
funkcja, 751
 aktywacji, 308–310, 345, 350
 ELU, 354
 GELU, 356

funkcja

- przeciekająca ReLU, 352
 - ReLU, 309
 - SELU, 354
 - sigmoidalna, 175, 308, 350
 - tangens hiperboliczny, 308
 - autokorelacji cząstkowej, PACF, 525
 - decyzyjna, 127
 - dopasowania, 47
 - gęstości prawdopodobieństwa, 262, 286
 - konkretna, concrete function, 750
 - logarytmiczna wiarygodności, 290
 - logistyczna, 175
 - podobieństwa, similarity function, 190
 - poszukiwania, exploration function, 666
 - radialna bazowa, RBF, 96
 - sigmoidalna, 175, 308
 - inicjalizacja Xaviera, 350
 - nasycenie, 350
 - skokowa, 301
 - softmax, 569
 - softplus, 310
 - straty, 410
 - Hubera, 311
 - logarytmiczna, 176
 - rekonstrukcji, reconstruction loss, 410
 - rzadkości, sparsity loss, 616
 - ukrytej, latent loss, 620
 - zawiasowa, 197
 - ucząca, 176
 - użyteczności, 47
 - wiarygodności, 289, 290
- ## funkcje
- aktywacji, 308–310, 345, 350
 - parametry inicjalizujące, 351
 - niestandardowe, 402
 - kosztu, 47, 66, 155, 176
 - entropia krzyżowa, 181
 - straty, 410
 - straty niestandardowe, 399
 - TF, 417, 421, 750, 755, 756
 - typu softmax, 180

G

- GAN, generative adversarial networks, 601, 623
- GBRT, gradient boosted regression trees, 231
- GCP, Google Cloud Platform, 688
 - autoryzacja, 691
 - uwierzytelnianie, 691

- generator, 602, 623
- generowanie
 - grafów, 420
 - obrazów, 622, 626, 630, 642
 - tekstów, 550, 554
- głęboki proces gaussowski, deep gaussian process, 385
- głosowanie miękkie, soft voting, 220
- GMM, Gaussian Mixture Model, 283
- Google Cloud
 - konsola platformy, 689
 - Shell, 690
 - Storage, 692
- Google Colab, 68
- gotowe
 - modele, 493, 494
 - reprezentacje właściwościowe, 563
- gradient, 411
 - niestabilny, 537
 - prosty, gradient descent, 151, 154
 - prosty wsadowy, batch gradient descent, 154, 155
- gradienty
 - przedawnione, stale gradients, 715, 716
 - strategii, policy gradients, 644, 648, 655
 - zanikające/eksplodujące, 348
- grafy, 420
 - modułu TensorFlow, 417, 708, 750
 - obliczeniowe, 752
- granica decyzyjna, 177, 179, 183
 - liniowa, 180
- gRPC, 685
- GRU, Gated Recurrent Unit, 543
- grupa zadań, job, 720

H

- harmonogram uczenia, learning schedule, 158, 377
- harmonogramowanie
 - 1cycle, 1cycle scheduling, 378
 - potęgowe, power scheduling, 378
 - stałoprzedziałowe, piecewise constant scheduling, 378
 - wydajnościowe, performance scheduling, 378
 - wykładnicze, exponential scheduling, 378
- haszowanie wrażliwe na lokalizację, 256
- HGB, histogram-based gradient boosting, 234

- hiperparametry, hyperparameters, 54, 344, 478
 - regularyzacyjne, 209
 - strojenie, 725, 727
- hipoteza, 66
 - rozmaitości, manifold hypothesis, 245
 - zerowa, 210

I

- iloczyn skalarny, 576
- implementacja
 - autokodera stosowego, 606
 - normalizacji wsadowej, 360
 - perceptronów wielowarstwowych, 313
 - Q-uczenia głębokiego, 667
 - sieci ResNet-34, 492
 - warstw łączących, 471
 - warstw splotowych, 465
- imputacja, imputation, 88
- inicjacja losowa, random initialization, 151
- inicjalizacja Xaviera, 350
- inicjalizatory, 402
- interfejs
 - funkcyjny
 - złożone modele, 324
 - gRPC, 685
 - Keras, 313, 434
 - wczytywanie danych, 314
 - podklasowy, subclassing API, 330
 - potokowy, 598
 - REST, 684
 - sekwencyjny, 314
 - regresyjny perceptron wielowarstwowy, 323
 - tworzenie modelu, 315
 - strategii rozpraszania, 718
 - tf.data, 424, 425
 - wiersza poleceń, CLI, 690
- interpolacja semantyczna, semantic interpolation, 622, 623
- istotność
 - cech, 226
 - statystyczna, 210
- iteracja Q-wartości, Q-Value iteration, 661

J

- jądro, kernel, 70
 - łączące, pooling kernel, 469
 - RBF, 191

- splotowe, convolution kernel, 462
- wielomianowe, 189, 190
- wielomianowe drugiego stopnia, 200
- jednoklasowa maszyna wektorów nośnych, 293
- jednostka SE-ResNet, 488
- jednostki rezydualne, residual units, 483
- Jupyter, 68, 71

K

- kanal
 - alfa, 275
 - barw, color channel, 463
- karta graficzna, 702
 - pamięć operacyjna, 704
- katastrofalne zapominanie, catastrophic forgetting, 670
- Keras
 - funkcje TF, 756
 - gotowe modele, 493
 - implementacja
 - autokodera stosowego, 606
 - normalizacji wsadowej, 360
 - perceptronów wielowarstwowych, 313
 - sieci ResNet-34, 492
 - warstw łączących, 471
 - warstw splotowych, 465
 - odczytywanie modelu, 331
 - stosowanie
 - wywołań zwrotnych, 332
 - zestawu danych, 434
 - uczenie transferowe, 365
 - warstwy
 - przetwarzania wstępnego, 441
 - splotowe, 508
 - wczytywanie danych, 314
 - zapisywanie modelu, 331
- Keras Tuner
 - strojenie hiperparametrów, 727
- kernelizowane maszyny SVM, 199
- klasa, 34
- klaster TensorFlow, TensorFlow cluster, 720, 721
- klasy SVM, 192
- klasyfikacja, classification, 34, 497
 - liniowa SVM, 185
 - maksymalnego marginesu, 185
 - miękkiego marginesu, soft margin classification, 187
 - nieliniowa SVM, 188

klasyfikacja, classification
 twardego marginesu, hard margin
 classification, 186
 wieloetykietowa, multilabel classification, 140
 wieloklasowa, 134
 wielowyjściowa, multioutput classification,
 141
 wielozadaniowa, 328
 klasyfikator
 binarny, binary classifier, 122
 Extra-Trees, 226
 głosujący, voting classifier, 217
 głosujący większościowo, hard voting
 classifier, 218
 obiektów, 503
 obrazów, 314
 SGD, 122
 silny, strong learner, 218
 słaby, weak learner, 218
 SVM
 liniowy, 189, 195
 wykorzystujący jądro RBF, 192
 wykorzystujący jądro wielomianowe, 190
 wieloklasowy, 134
 klątwa wymiarowości, curse of dimensionality,
 240
 koder, encoder, 516, 603
 kodowanie
 cech kategorialnych, 447
 gorącojedynekowe, one-hot encoding, 91
 par bajtów, BPE, 559
 pozycyjne, Positional Encodings, 580, 581
 kolejki, 399, 748
 kolizja haszowania, hashing collision, 447
 komórki
 GRU, 543
 LSTM, 540, 541
 pamięci, memory cells, 515
 podstawowe, basic cells, 515
 kompromis
 pomiędzy obciążeniem a wariancją, 167
 pomiędzy precyzją a pełnością, 127
 koneksjonizm, 298
 kontaminacja, stacking, 217, 235
 kontrolery bramek, gate controllers, 542
 konwersje typów, 397
 korelacja, 84
 kreślenie, 419
 krok, stride, 461
 krosvalidacja, cross-validation, 57
 kryterium
 informacji teoretycznej, 289
 informacyjne Akaikiego, 289
 informacyjne bayesowskie, 289
 krzywa
 PR, 132, 133
 ROC, 130, 131
 krzywe uczenia, learning curves, 163–166, 336
 kubekowanie, bucketizing, 95
 kwadrat zawiasowej funkcji straty, 198
 kwantyzacja potreningowa, post-training
 quantization, 698
 kwartył, 75

L

las
 izolacyjny, 217, 225, 292
 losowy, 214, 217, 225, 292
 LDA, linear discriminant analysis, 258
 liczba
 iteracji, 345
 skupień, 289
 liczby dualne, 739
 liniowa jednostka progowa, LTU, 301
 lista dwukierunkowa, deque, 668
 liść, 204
 LLE, locally linear embedding, 256
 logarytm
 binarny, binary logarithm, 208
 szans, log odds, 175
 lokalizowanie, 497
 lokalne pola recepcyjne, 459
 lokalnie liniowe zanurzenie, LLE, 256
 losowa analiza PCA, 252
 losowanie warstwowe, stratified sampling, 79,
 81
 losowe
 lasy, random forest, 214, 217, 225, 292
 podprzestrzenie, 224
 próbkiowanie, 81
 przeszukiwanie, 111
 rejony, 224
 LRN, local response normalization, 478
 LSH, locality sensitive hashing, 256
 LSTM, Long Short-Term Memory, 540
 LTU, Linear Threshold Unit, 301

Ł

łańcuchy
Markowa, 659
znaków, 744
łączenie przekształceń, 426

M

macierz
jednostkowa, 169
kodowań pozycyjnych, 582
korelacji, 87
kowariancji, 284
ortogonalna, 569
osadzeń, embedding matrix, 449
parametrów Θ , 180
pomyłek, 124, 138, 139
rzadka, sparse matrix, 92
MAE, Mean Absolute Error, 66
mała wydolność próbkowania, sample inefficient, 658
mapa
cech, feature map, 463
prawdopodobieństw, 504
MAPE, mean absolute percentage error, 521
maskowanie, 560
maskowany model językowy, MDM, 588
maszyna
ograniczona Boltzmanna, 612
wektorów nośnych, SVM, 185
MDM, Masked Language Model, 588
MDS, multidimensional scaling, 258
mechanizm
dopasowywania, Matching Engine, 688
uwagi, attention mechanisms, 550, 573
addytywny, additive attention, 575
konkatenacyjny, concatenative attention, 575
Luonga, Luong attention, 576
maskowany wieloblokowy,
masked multi-head attention, 580
multiplikatywny, multiplicative attention, 576
skalowany, 583
wieloblokowy, multi-head attention, 580, 583, 584
wykorzystujący iloczyn skalarny, 583
wzrokowej, visual attention, 592
mediana, 85
metagraf, 681
metaportale, 62
metauczeń, meta-learner, 235
metoda
elastycznej siatki, 172
gradientu prostego, 154
LASSO, 170
losowego przeszukiwania, 111
przeszukiwania siatki, 109
różnic czasowych, TD, 663
różnic skończonych, 737
wczesnego zatrzymywania, 173
metody zespołowe, 112, 219
miara
podobieństwa, 44
wydajności, 122
mieszanie stylów, style mixing, 635
mieszanie gaussowskie, 283
dla skupień
powiązanych, 287
sferycznych, 287
nieeliptycznych, 292
wykrywanie anomalii, 287
minimum
globalne, 152
lokalne, 152
minipaczki, mini-batches, 160
minipakiet/minigrupa, mini-batches, 42
MLP, Multi-Layer Perceptron, 297, 305
mod, 95
model
ARIMA, 523
ARIMA sezonowy, 524
ARMA, 523
BERT
uczenie i strojenie, 589
DDPM, 602
DeiT, 594
mieszanie gaussowskiej, GMM, 283
odszumiający probabilistyczny rozpraszający,
DDPM, 636
Q-uczenia głębokiego, 667
SARIMA, 524
StyleGAN, 634
tłumaczenia maszynowego, 566
uwagi Bahdanau, Bahdanau attention, 575
WaveNet, 546

modele

- „białej skrzynki”, white box models, 206
- „czarnej skrzynki”, black box models, 206
- analizowanie, 112
- dynamiczne, 330
- generatywne, generative models, 285, 601
- językowe, language models, 453, 550, 563
- liniowe, 46, 47
 - prognozowanie, 528
- nieparametryczne, nonparametric models, 209
- niestandardowe, 391, 408
- o największej wiarygodności, 176
- parametryczne, parametric models, 209
- przyczynowe, causal models, 536
- rozpraszające, diffusion models, 601
- rozpraszające niejawne, latent diffusion models, 641
- regresji
 - liniowej, 47
 - wielomianowej, 163
- rzadkie, 170, 376
- sekwencyjne
 - prognozowanie, 534
- strojenie, 109
- transformatorów, 587
- uczenia maszynowego, 525
- wdrożone, 115
- wyrównania, alignment models, 575
- zapisywanie i wczytywanie, 400
- zawierające elementy niestandardowe, 400
- złożone, 324

moduł

- incepcyjny, inception module, 479, 480
- Scikit-Learn, 89
- SE-Inception, 488
- TensorFlow, 391

N

- nadmierne dopasowanie, overfitting, 53
- naruszanie marginesu, margin violation, 187
- narzędzie
 - AutoGraph, 419, 754
 - OpenAI Gym, 648
 - TensorBoard, 333
- nasycenie przepustowości, 716
- nauczanie według schematu, 677
- neuroewolucja głęboka, Deep Neuroevolution, 342

neuron, 298

- biologiczny, 299, 459
- operacje logiczne, 300
- opiniotwórczy, sentiment neuron, 557
- rekurencyjny, 513
- rozwijany w czasie, 513
- sztuczny, artificial neuron, 300
- neuronowe tłumaczenie maszynowe, NMT, 550, 565
- niedobór danych uczących, 50
- niedorzeczna efektywność danych, 49
- niedotrenowanie, underfitting, 55
- niereprezentatywne dane uczące, 50
- niezgodność danych, 57
- NLDR, nonlinear dimensionality reduction, 256
- NLP, Natural Language Processing, 368, 549
- NLU, Natural Language Understanding, 591
- NMT, Neural Machine Translation, 550
- norma euklidesowa, 67
- normalizacja, 94
 - odpowiedzi lokalnej, LRN, 478
 - warstwowa, layer normalization, 538
 - wsadowa, batch normalization, 358
 - implementacja, 360
- notatnik Jupyter, 68, 71
- NumPy, 394, 396

O

- obciążenie, bias, 167
 - próbekowania, sampling bias, 51
 - związane z podglądaniem danych, 77
- obcinanie gradientu, gradient clipping, 363
- obliczanie gradientów, 411
- obsługa zmiennych, 755
- obszar pod krzywą, AUC, 131
- ocena OOB, 223
- OCR, Optical Character Recognition, 27
- odchylenie standardowe, standard deviation, 75, 619
- odkładanie wag, weight stashing, 717
- odkrywanie cechy, feature extraction, 52
- odtwarzanie
 - doświadczenia, experience replay, 628
 - priorytetowych doświadczeń, PER, 672
- odwrotne różniczkowanie automatyczne, 307
- OEL, Open-Ended Learning, 677
- ograniczenia, 402
- określanie zakresu problemu, 63

OOB, out-of-bag instances, 223
OpenAI Gym, 648
operacje, 395, 706
optymalizacja
 momentum, momentum optimization, 369
 momentum Nesterova, 370
 warstwy wyjściowej, 569
optymalizator, 344, 368
optymalna wartość stanu, 660
oszacowanie
 maksymalne a posteriori, 290
 maksymalnej wiarygodności, 290

P

PACF, partial autocorrelation function, 525
pakiet sklearn.utils.validation, 99
pamięć
 krótkotrwała, 540
 odtworzania, replay memory, 667
 operacyjna, 468, 704
parametry strategii, policy parameters, 647
PCA, Principal Component Analysis, 245
PDF, Probability Density Function, 286
pełność, 125, 127
percentyl, 75
perceptron, 301, 302
 reguła uczenia, 304
 twierdzenie o zbieżności, 304
 wielowarstwowy, MLP, 297, 305, 306, 311, 312
 implementowanie, 313
 klasyfikacyjny, 311
 regresyjny, 309, 323
pętla uczenia niestandardowe, 415
PG, policy gradients, 648
pieńki decyzyjne, decision stump, 230
pierwiastek błędu średniokwadratowego, RMSE, 65
pierwszy projekt, 61
pliki
 CSV, 72, 432
 TFRecord, 436
 zdarzeń, event files, 334
pobieranie wstępne, 432
pochodna cząstkowa, 154
 funkcji kosztu, 155
podobieństwo, 101
podpróbkiwanie, subsample, 469

podsumowanie, summary, 334
połączenia
 pomijające, skip connections, 482, 485
 skrótowe, shortcut connections, 482
pomiar dokładności, 123
porzucanie, dropout, 382
poszukiwanie oparte na ciekawości, 676
potok, pipeline, 64
 Scikit-Learn, 99
 transformujący, 101
 uczenia maszynowego, 30, 64
PPO, Proximal Policy Optimization, 676
PR, precision/recall curve, 131
prawdopodobieństwo, 175, 206
 utrzymywania, keep probability, 384
 warunkowe, 572
prawo wielkich liczb, 219
prawoskośność, skewed right, 76
precyzja, 125, 127
predyktor, predictor, 35
problem
 dualny, dual problem, 198
 NP-zupełny, 208
 pierwotny, primal problem, 198
 przypisania zasługi, credit assignment problem, 654
proces
 decyzyjny Markowa, 659, 660
 gaussowski, 341
 przedni, 638
 rozpraszania, 641
 rozpraszania w przód, 638
 w przód, forward process, 637
 wsteczny, reverse process, 637, 638
procesory graficzne, 701
prognozowanie, forecasting, 45, 519
 kilku taktów w przód, 533
 naiwne, naive forecasting, 519
 szeregów czasowych, 518, 531
 za pomocą
 głębokich sieci rekurencyjnych, 530
 modelu liniowego, 528
 modelu sekwencyjnego, 534
 sieci rekurencyjnej, 529
prognozy, predictions
 algorytmu AdaBoost, 230
 modelu regresji wielomianowej, 163
 progowa jednostka logiczna, TLU, 301

- programowanie
 - dynamiczne, 661
 - kwadratowe, quadratic programming, 197
- projekt
 - TensorFlow Datasets, 455
 - uczenia maszynowego, 731
- propagacja
 - etykiet, label propagation, 277
 - podobieństwa, 283
 - wsteczna, backpropagation, 306, 307
 - wsteczna w czasie, BPTT, 517
- protokół OAuth 2.0, 691
- próba, trial, 725
- próbka ucząca, 28
- próbkowanie
 - jądra, nucleus sampling, 555
 - niepewności, uncertainty sampling, 279
 - ważone, importance sampling, 672
- próg decyzyjny, 127
- przedział ufności, confidence interval, 113
- przeглядarka
 - przetwarzanie modelu, 699
- przekształcenia afiniczne, affine transformations, 634
- przeplatanie wierszy, 429
- przepustowość pamięci, memory bandwidth, 433
- przestrzeń
 - strategii, policy space, 647
 - ukryta, latent space, 619
- przeszukiwanie
 - siatki, 109
 - wiązkowe, beam search, 572, 573
- przetrenowanie, 53, 381
- przetwarzanie
 - brzegowe, edge computing, 697
 - grafu TensorFlow, 702
 - języka naturalnego, NLP, 368, 549
 - obrazów, 454
 - równoległe, 708
 - sekwencji, 544
 - tekstu, 451
 - wstępne, 441
- przewaga czynności, action advantage, 655
- przewidywanie następnego zdania, 589
- przygotowywanie danych, 525
- przykład uczący, 28
- przykłady pozatreningowe, OOB, 223
- pula wątków
 - międzyoperacyjna, 708
 - wewnątrzoperacyjna, 708

- punkt
 - charakterystyczny, landmark, 190
 - końcowy, endpoint, 693
 - obciążenia, bias term, 146
 - przecięcia, intercept term, 146
- PWA, progressive web app, 700
- p-wartość, 210

Q

- Q-uczenie, 664
- Q-uczenie głębokie, 666
 - implementacja modelu, 667
 - podwójna sieć DQN, 672
 - ustalone Q-wartości, 671
 - walcząca sieć DQN, 673

R

- rachunek zdań, propositional logic, 298
- radialna funkcja bazowa, RBF, 96, 190
- realizacja pospieszna, eager execution, 420
- redukowanie wymiarowości, dimensionality reduction, 37, 462
 - główne składowe, 247
 - liniowa analiza dyskryminacyjna, 258
 - nieliniowe, NLDL, 256
 - rzutowanie, 242
 - losowe, 254
 - na d wymiarów, 248
 - skalowanie wielowymiarowe, 258
 - stochastyczne zanurzanie sąsiadów, 258
 - uczenie rozmaitościowe, 244
 - wybór liczby wymiarów, 249
 - zachowanie wariancji, 246
- regresja, 34, 211
 - do średniej, 34
 - grzbietowa, ridge regression, 167, 169
 - jednoczynnikowa, univariate regression, 65
 - k-najbliższych sąsiadów, 48
 - liniowa, 146, 150
 - logistyczna, logistic regression, 35, 174
 - logitowa, logit regression, 174
 - metodą
 - elastycznej siatki, 172
 - LASSO, 170
 - softmax, 180
 - SVM, 193, 195
 - wieloczynnikowa, multivariate regression, 65

wielomianowa, polynomial regression, 161
wielomianowa wysokiego stopnia, 164
wieloraka, multiple regression, 65
regresyjne drzewo decyzyjne, 211
regularyzacja, regularization, 54
 ℓ_1 i ℓ_2 , 381
metodą wczesnego zatrzymywania, 173
mieszająca, mixing regularization, 635
podśłów, subword tokenization, 559
regresji
grzbietowej, 171
metodą LASSO, 171
Tichonowa, Tikhonov regularization, 167
typu max-norm, 387
typu Monte Carlo, 385
regularyzatory, 402
reguła
aktualizowania wag, 229
Hebba, 303
łańcuchowa, chain rule, 308, 742
rekonstrukcje, reconstructions, 603
repliki zapasowe, 715
repozytoria danych, 61
reprezentacje
danych, 602
ukryte, latent representations, 594, 601
właściwościowe słów, 448
ResNet-34
implementacja sieci, 492
REST, 684
retuszowanie, outpainting, 642
RL, Reinforcement Learning, 644
RMSE, Root Mean Square Error, 65, 173
RNN, Recurrent Neural Networks, 512
ROC, receiver operating characteristic, 130
rotacja zbioru uczącego, 213
rozgrzewanie modelu, model warmup, 687
rozkład
gruboogonowy, heavy tail, 95
modelu, model rot, 41
normalny, 75
potęgowy, power law distribution, 95
prawdopodobieństwa, 637
według wartości osobliwych, SVD, 150, 248
rozmaitość, manifold, 245
rozmiar grupy danych, 344
rozpraszanie stabilne, Stable Diffusion, 642
rozróżnianie miniwadami, mini-batch
discrimination, 628
rozumienie języka naturalnego, NLU, 591
równanie
kwadratowe, 162
normalne, 148
optymalności Bellmana, 660
równowaga Nasha, Nash equilibrium, 627
różniczkowanie
automatyczne, 411, 738
automatyczne odwrotne, 741
ręczne, 736
symboliczne, symbolic differentiation, 739
RPN, Region Proposal Network, 506
rzadkość, sparsity, 615
rzutowanie, 242, 247
losowe, random projection, 254
na d wymiarów, 248

S

SAC, Soft Actor-Critic, 676
samodestylacja, self-distillation, 594
samonormalizacja, self-normalize, 355
schemat macierzy pomyłek, 126
schodzenie po gradiencie z minigrupami, 160
Scikit-Learn, 89
segmentacja
instancji, instance segmentation, 509
obrazu, image segmentation, 274
semantyczna, semantic segmentation, 507
serwer
główny, 720
parametrów, parameter server, 714, 720
roboczy, worker, 714, 720
sezonowość, seasonality, 519
SGD, Stochastic Gradient Descent, 122, 157
sieci neuronowe
dostrajanie hiperparametrów, 337
jako strategię, 652, 653
typu koder – dekodek, 516, 565
sieć neuronowa
głęboka, DNN, 306, 348, 388, 484
DQN podwójna, Double DQN, 672
DQN walcząca, Dueling DQN, 673
Q-sieć, Deep Q-networks, DQN, 644, 666
jednokierunkowa, FNN, 306
generatywna przeciwstawna, GAN, 601, 623
sieć neuronowa
uczenie, 627
rozrost progresywny, 631

sieć neuronowa
 plotowa głęboka, DCGAN, 628
 StyleGAN, 634
 warunkowa, CGAN, 631
rekurencyjna, RNN, 512
 bezstanowa, stateless RNN, 549
 prognozowanie, 529
 dwukierunkowa, 571
 głęboka, 530
 sekwencyjna, sequence-to-sequence network, 516
 sekwencyjno-wektorowa, sequence-to-vector network, 516
 stanowa, stateful RNN, 550
 wektorowo-sekwencyjna, vector-to-sequence network, 516
 znakowa, char-RNN, 549, 553, 555
splotowa, CNN, 458, 460
 AlexNet, 476
 GoogLeNet, 479, 481
 LeNet-5, 460, 475
 ResNet, Residual Network, 482, 484
 SENet, 487
 VGGNet, 482
 w pełni połączona, FCN, 501
 wybór struktury, 491
 Xception, 486
sztuczna, SSN, ANN, 297
Wide & Deep, 325
YOLO, 503
 sieć proponowania rejonów, RPN, 506
sigmoidalna jednostka liniowa, 356
silnik Dockera, Docker engine, 682
skalowanie
 cech, feature scaling, 94, 154
 min. – max., min-max scaling, 94
 wielowymiarowe, MDS, 258
składnia obiektów, 439
spadek wzdłuż gradientu, 157, 370
specyficzność, 130
SPMD, single program, multiple data, 713
sprawdzian krzyżowy, 57, 107, 123
 implementacja, 123
stan końcowy, terminal state, 659
standaryzacja, standarization, 94
stochastyczne zanurzenie sąsiadów, 258
stochastyczny spadek wzdłuż gradientu, SGD, 157
stopa dyskontowa, discount factor, 654
stopień
 niezmienniczości, 470
 zintegrowania, order of integration, 523
stopnie swobody, 54
stosy map cech, 463
strata rekonstrukcji, reconstruction loss, 603
strategia, policy, 646
 duplikowania, mirrored strategy, 713
 poszukiwania, 665
 stochastyczna, stochastic policy, 647
 ϵ -zachłanna, ϵ -greedy policy, 665
strojenie, 109
 hiperparametrów, 56, 725, 727
 modelu BERT, 589
struktura
 Mask R-CNN, 509
 CNN, 491
struktury danych, 73, 398, 744
superrozdzielczość, super-resolution, 509
superzbieżność, super-convergence, 378
SVD, singular value decomposition, 248
SVM, support vector machine, 185
 działanie liniowych klasyfikatorów, 195
 klasyfikacja liniowa, 185
 klasyfikacja nieliniowa, 188
 maszyny kernelizowane, 199
 regresja, 193
symulowane wyżarzanie, simulated annealing, 158
syntetyczne nadpróbkiwanie mniejszości, 477
system
 Pathways, 718
 PipeDream, 717
systemy uczenia maszynowego, 33
szacowanie
 gęstości, 262
 prawdopodobieństwa, 175, 178, 206
szereg czasowy, time series, 512, 518, 519
 jednowymiarowy, univariate time series, 519
 prognozowanie, 531
 wielowymiarowy, multivariate time series, 519
szeregowanie, scheduling, 718
 zespołów, gang scheduling, 718
szerokość wiązki, beam width, 572
sztuczka
 z funkcją jądra, kernel trick, 189
 z haszowaniem, hashing trick, 447
szybkość uzyskania zbieżności, 157

Ś

- ścieżki algorytmów gradientu prostego, 161
- śledzenie, tracing, 753
 - obiektów, 506
- średni
 - absolutny błąd, MAE, 66
 - bezwzględny błąd procentowy, MAPE, 521
- średnia
 - harmoniczna, 126
 - krocząca autoregresywna, ARMA, 523
- środowisko
 - CartPole, 650
 - symulowane, simulated environment, 648
 - wykonawcze, runtime, 69

T

- tablice tensorów, 398, 746
- tasowanie danych, 428
- TD, temporal difference, 663
- tensor, 395, 396
 - maski, mask tensor, 561
 - nierówny, 398, 745
 - rzadki, 398, 746
 - symboliczny, symbolic tensor, 420
 - znakowy, 398
- TensorBoard, 333
- TensorFlow, 391
 - architektura modułu, 393
 - bufory protokołów, 438
 - CUDA, 703
 - cuDNN, 703
 - eksploatacja modelu, 680
 - funkcje, 417
 - generowanie grafów, 420
 - grafy, 417
 - klaster, 720, 721
 - przetwarzanie równoległe grafu, 708
 - uczenie modelu, 720
 - wdrażanie modeli, 679
 - wstępne przetwarzanie danych, 424
- TensorFlow Hub, 394
- TensorFlow Lite, 394
- TensorFlow Playground, 313
- TensorFlow Serving, 680
 - eksportowanie obiektów SavedModel, 680
 - instalowanie serwera, 682
 - skalowanie systemu, 688

- uruchamianie serwera, 682
- wdrażanie modelu, 686
- wysyłanie zapytań, 684, 685
- TensorFlow.js, 394
- teoria
 - informacji Shannona, 208
 - sterowania, 649
- TLU, Threshold Logic Unit, 301
- transformator, transformer, 550, 578
 - niestandardowy, 97
 - wizualny, 592
- transpozycja macierzy, 147
- trenowanie, 105

- tryb
 - grafowy, graph mode, 420
 - pospieszny, eager mode, 420
- t-SNE, t-Distributed stochastic neighbor embedding, 258
- twierdzenie
 - Mercera, 200
 - o nieistnieniu darmowych obiadów, 59
 - o zbieżności perceptronu, 304
- tworzenie
 - zbioru testowego, 77
 - zestawu danych uczących, 551

U

- uczenie
 - „jednostrzałowe”, single-shot learning, 510
 - aktywne, active learning, 279
 - autokoderów pojedynczo, 611
 - drzewa decyzyjnego, 203
 - federacyjne, federated learning, 701
 - głębokich sieci neuronowych, 348
 - hebbowskie, 303
 - klasyfikatora binarnego, 122
 - maszynowe, 28
 - maszynowe wytłumaczalne, interpretable ML, 206
 - metodą różnic czasowych, 663
 - miksera, 236
 - modeli, 145
 - modeli rzadkich, 376
 - modelu BERT, 589
 - modelu char-RNN, 553
 - nadzorowane, supervised learning, 34
 - nienadzorowane, unsupervised learning, 35, 261

uczenie

- nieustanne, Open-Ended Learning, 677
- od strzału, ZSL, 590
- perceptronu, 304
- pozakorowe, out-of-core learning, 43
- półnadzorowane, semisupervised learning, 38, 276
- przeciwstawne, adversarial learning, 510, 602
- przez wzmacnianie, reinforcement learning, RL, 40, 644, 646
- przy użyciu reguł asocjacyjnych, association rule learning, 37
- przyrostowe, online learning, 42, 122
- rezydualne, residual learning, 483
- rozmaitościowe, manifold learning, 244, 245
- samonadzorowane, self-supervised learning, 368
- sieci GAN, 627
- sieci rekurencyjnych, 517
- się optymalizowania nagród, 645
- się reprezentacji, representation learning, 93
- transferowe, transfer learning, 40, 342, 363, 494
- w kontekście kwantyzacji, 699
- wielkoskalowe, 679, 718
- wsadowe, batch learning, 40
- wstępne
 - za pomocą dodatkowego zadania, 367
 - nienadzorowane, unsupervised pretraining, 367, 609
- z modelu, model-based learning, 45
- z przykładów, instance-based learning, 44
- zespołowe, 217

ukierunkowania indukcyjne, inductive bias, 593

urządzenia

- logiczne, logical devices, 705, 706
- mobilne
 - wdrażanie modelu, 697
- przetwarzanie równoległe, 708
- uczenie modeli, 710
- umieszczanie operacji i zmiennych, 706

usługa robocza, service worker, 700

usuwanie niemaksymalnych pikseli, 500

uśrednione kodowanie μ , 619

utrzymywanie systemu, 114

uwaga krzyżowa, cross-attention, 580

uzupełnianie, inpainting, 642

- zerami, zero padding, 461, 466

V

VAE, variational autoencoders, 618

Vertex AI

- grupy zadań uczenia, 723
- strojenie hiperparametrów, 725, 727
- usługi predykcyjne, 688
- zadania predykcji wsadowych, 695

W

waga predyktora, 229

wagi Glorota i He, 349

wariancja, variance, 167, 214, 246, 250

- sygnału, 639
- szumu, 639

warstwa

- AdaIN, 635
- CategoryEncoding, 444
- Discretization, 444
- Hashing, 447
- Normalization, 441
- StringLookup, 446

warstwy

- dekonwolucyjne, deconvolution layers, 508
- ekspansji, upsampling layers, 507
- łącznie, pooling layers, 460, 469
 - globalne uśredniające, global average pooling layers, 472
 - implementacja, 471
 - maksymalizujące, max pooling layers, 469
 - uśredniające, average pooling layers, 471
 - w głąb, depth concatenation layers, 479

neuronów

- rekurencyjnych, 514
- rozwijane w czasie, 514

niestandardowe, 405

normalizacji piksel po pikselu, 633

odchylenia standardowego miniwsadów, 632

ograniczające, bottleneck layers, 479

przetwarzania wstępnego, 441

rekurencyjne, 513

splotowe, 460, 464

- implementacja, 465
- jednowymiarowe, 544
- rozdzielne po głębokości, 486
- rozszerzone, α trous convolutional layers, 509
- w interfejsie Keras, 508

- ukryte, 342
 - liczba neuronów, 343
 - wstępnego przetwarzania obrazów, 454
 - wyjściowe
 - optymalizacja, 569
- wartości stanu – czynności, state-action values, 661
- wartość docelowa, 35
- ważenie częstości termów, 452
- wczesne zatrzymywanie, early stopping, 173
- wcześniejsze zakotwiczenia, anchor prior, 504
- wdrażanie modeli TensorFlow, 679
- wektor
 - gradientów funkcji kosztu, 155
 - kolumnowy, 147
 - momentu, 369
 - nośny, support vector, 185
 - podgradientów, 172
 - wierszowy, 147
 - właściwościowy, embedding, 93, 447, 448, 563
 - segmentowy, segment embedding, 589
- węzeł
 - główny, root node, 204
 - podziału, split node, 205
- wiązanie wag, tying weights, 610
- wielomianowa/wieloraka regresja logistyczna, multinomial logistic regression, 180
- wielowarstwowy stos kontaminujący, 237
- wizualizacja, 81
 - danych, 81, 333
 - drzewa decyzyjnego, 203
 - rekonstrukcji, 607
 - zestawu danych, 608
- wklejanie, pasting, 221, 222
- wnioskowanie, 49
- Workbench, 688
- wskaźnik
 - Giniego, 208
 - mAP, 505
 - niestandardowy, 403
 - oparty na elementach wewnętrznych, 410
 - stanowy, stateful metric, 404
 - strumieniowy, streaming metric, 404
 - wartości przeciętnej precyzji, AP, 505
 - wydajności, 65
- współczynnik
 - korelacji liniowej, 84, 86
 - lokalny elementów odstających, LOF, 292
 - porzucenia, dropout rate, 383
 - profilu, silhouette coefficient, 272
 - rozszerzalności, dilution rate, 509
 - uczenia, learning rate, 43, 151, 156, 344, 377
 - uczenia wyrównany, 633
 - wariancji wyjaśnionej, 249
 - ważony błędu, 229
 - wsparcia, support, 141
- wstępne przetwarzanie
 - danych, 430, 432
 - obrazów, 454
 - tekstu, 451
- wydobywanie
 - cech, feature extraction, 37
 - danych, data mining, 30
- wyjaśnialność, explainability, 510, 593
- wykres
 - funkcji logistycznej, 175
 - krzywej ROC, 130
 - mediany, 85
 - precyzji i pełności, 129
 - profilu, silhouette diagram, 272
 - punktowy danych, 82
- wykrywanie
 - anomalii, anomaly detection, 37, 261, 287
 - algorytm Fast-MCD, 292
 - jednoklasowa maszyna wektorów nośnych, 293
 - las izolacyjny, 292
 - współczynnik LOF, 292
 - nowości, novelty detection, 37
 - obiektów, 499, 500
- wyliczanie prognoz, 204
- wymuszanie nauczyciela, teacher forcing, 565
- wynik
 - obiektowości, objectness score, 499
 - profilu, silhouette score, 272
- wynikanie, entailment, 588
- wyostrzenie, sharpening, 595
- wyroczenia, oracle, 339
- wyszukiwanie strategii, policy search, 646, 647
- wyznaczanie liczby skupień, 289
- wzmacnianie, boosting, 217, 227
 - adaptacyjne, 227
 - gradientowe, gradient boosting, 231, 232, 227
- wzmacnianie, boosting
 - drzew, 231
 - w oparciu o histogram, HGB, 234
 - hipotezy, hypothesis boosting, 227

wzór regresji grzbietowej, 169
wzrost gradientu, gradient ascent, 648

Z

zadanie, task, 720
załamanie modu, mode collapse, 594, 627
zanieczyszczenie Giniego, Gini impurity, 205
zasobnik, bucket, 692
zasoby, resources, 755
zastosowania uczenia maszynowego, 31
zaszumienie próbkowania, sampling noise, 51
zbieżność algorytmu, 151, 157
zbiór, 398, 747
 testowy, test set, 56, 113
 tworzenie, 77
 uczący, training set, 28, 56, 106
zespoły
 agregujące, bagging, 217
 kontaminujące, 236
 wklejające, pasting, 217
zespół, ensemble, 108
 Extra-Trees, 225
 GBRT, 233

zestaw danych
 Fashion MNIST, 315, 608, 622
 MNIST, 119
 płaski, flat dataset, 527
 Swiss roll, 244, 257
 ucząco-rozwojowy, train-dev-set, 58
 uczących
 tworzenie, 551
 zagnieżdżony, nested dataset, 526
 złożoność obliczeniowa, 151, 192, 208
 zmienne, 397, 706
znormalizowana funkcja wykładnicza, 180
zrównoleglanie
 danych, data parallelism, 710, 713
 za pomocą scentralizowanych parametrów,
 714
 za pomocą strategii duplikowania, 713
 modelu, model parallelism, 710
 potokowe, pipeline parallelism, 717
ZSL, Zero-Shot Learning, 590

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 

Twórz i trenuj nowoczesne sieci neuronowe!

Znajdziesz tu rozsądne,
intuicyjne objaśnienia,
a także mnóstwo praktycznych porad!

François Chollet, twórca interfejsu Keras

Głębokie sieci neuronowe mają niesamowity potencjał. Osiągnięcia ostatnich lat nadały procesom uczenia głębokiego zupełnie nową jakość. Obecnie nawet programiści niezaznajomieni z tą technologią mogą korzystać z prostych i niezwykle skutecznych narzędzi, pozwalających na sprawne implementowanie programów uczących się z danych.

To trzecie wydanie bestsellerowego przewodnika po uczeniu maszynowym. Książka jest adresowana do osób, które chcą wejść w świat uczenia maszynowego — przy czym wystarczą do tego minimalne umiejętności programistyczne. Zawarto tu minimum teorii, a proces nauki ułatwiają liczne przykłady i ćwiczenia. Dzięki temu przyswoisz niezbędne pojęcia i nauczysz się korzystać z gotowych platform produkcyjnych Pythona: Scikit-Learn, Keras i TensorFlow. W tym wydaniu pokazano różnorodne techniki, od prostej regresji liniowej aż po głębokie sieci neuronowe. Szybko nauczysz się tworzyć działające systemy inteligentne!

To znakomite wprowadzenie do teoretycznych i praktycznych rozważań na temat rozwiązywania problemów za pomocą sieci neuronowych!

Pete Warden, mobile lead projektu TensorFlow

W książce między innymi:

- korzystanie ze Scikit-Learn, z TensorFlow i Keras
- modele: maszyny wektorów nośnych, drzewa decyzyjne, lasy losowe i metody zespołowe
- uczenie nienadzorowane: redukcja wymiarowości, analiza skupień, wykrywanie anomalii
- sieci neuronowe: sieci splotowe, rekurencyjne, modele dyfuzyjne i transformatory
- trenowanie i implementacje sieci neuronowych

Aurélien Géron jest konsultantem i wykładowcą. Studiował mikrobiologię i genetykę ewolucyjną, pracował w Google, JP Morgan i Société Générale, a także w Ministerstwie Obrony Kanady. Napisał kilka książek o programowaniu w języku C++, sieciach Wi-Fi i architekturze sieci internetowych. Uczył też informatyki na francuskiej politechnice.

Helion 

 helion.pl

 **HELION SA**
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 250 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej! ▶



ISBN 978-83-8322-423-7



Cena: 179,00 zł