O'REILLY®

# Jak projektować systemy uczenia maszynowego

## Iteracyjne tworzenie aplikacji gotowych do pracy

Helion

Chip Huyen

Tytuł oryginału: Designing Machine Learning Systems:
An Iterative Process for Production-Ready Applications

Tłumaczenie: Jacek Janusz

ISBN: 978-83-283-9912-9

© 2023 ***Helion S.A.***

Authorized Polish translation of the English edition of *Designing Machine Learning Systems*
ISBN 9781098107963 © 2022 Huyen Thi Khanh Nguyen.

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 jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

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

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

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

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

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

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

# Spis treści





Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

# ROZDZIAŁ 4.

# Dane treningowe

Z rozdziału 3. dowiedziałeś się, jak można pracować z danymi na poziomie systemowym. W tym rozdziale zajmiemy się ich obsługą z punktu widzenia danetyki. Pomimo tego, że dane treningowe odgrywają ważną rolę w projektowaniu i ulepszaniu modeli uczenia maszynowego, kursy szkoleniowe są mocno ukierunkowane na samo tworzenie, które przez wielu praktyków jest uważane za „przyjemną" część procesu. Tworzenie najnowocześniejszego modelu jest interesujące. Spędzanie dni na zmaganiu się z ogromną ilością źle sformatowanych danych, które nie mieszczą się nawet w pamięci maszyny, jest frustrujące.

Dane są nieuporządkowane, złożone, nieprzewidywalne i potencjalnie zdradliwe. Jeśli nie są odpowiednio obsługiwane, mogą łatwo się przyczynić do upadku projektu uczenia maszynowego. Jest to więc powód, dlaczego danetycy i inżynierowie ML powinni się nauczyć dobrze obsługiwać dane — pozwoli im to później zaoszczędzić dużo czasu i uniknąć bólu głowy.

W tym rozdziale przeanalizujemy techniki pozyskiwania lub tworzenia poprawnych danych treningowych. Dane treningowe obejmują wszystkie dane używane w fazie projektowania modeli uczenia maszynowego, w tym różne zbiory wykorzystywane do trenowania, walidacji i testowania (czyli zbiory treningowe, walidacyjne i testowe). Rozdział ten rozpoczyna się od zaprezentowania różnych technik próbkowania w celu wyboru danych treningowych. Następnie przeanalizowane zostaną wyzwania związane z tworzeniem danych treningowych, w tym problem mnogości etykiet, ich braku, niezrównoważenia klas oraz metody wykorzystywane do generowania sztucznych danych w celu radzenia sobie z brakiem wystarczającej ilości informacji.

Używamy terminu „dane treningowe" zamiast „zestaw danych treningowych", ponieważ słowo „zestaw" oznacza coś skończonego i stałego. Dane produkcyjne nie są skończone ani stałe. Więcej o tym dowiemy się w rozdziale 8., w podrozdziale „Zmiany rozkładów danych". Podobnie jak inne etapy tworzenia systemów uczenia maszynowego, również generowanie danych treningowych jest procesem iteracyjnym. W miarę jak model ewoluuje w cyklu życia projektu, dane treningowe prawdopodobnie również się zmieniają.

Zanim przejdziemy dalej, chciałabym powtórzyć przestrogę, która choć pojawiła się już w tej książce wiele razy, wciąż należy o niej pamiętać. Dane są pełne potencjalnych błędów. Wynika to z wielu przyczyn. Zniekształcenia pojawiają się podczas gromadzenia danych, próbkowania lub etykietowania. Na dane historyczne mogą mieć wpływ ludzkie uprzedzenia, a modele uczenia maszynowego, trenowane za ich pomocą, mogą je utrwalać. Korzystaj z danych, ale nie ufaj im za bardzo!



Kup książkę Poleć książkę

# Próbkowanie

Próbkowanie jest integralną częścią procesu uczenia maszynowego, która niestety jest często pomijana na typowych kursach. Ma ono miejsce na wielu etapach cyklu życia projektu ML. Chodzi na przykład o pobieranie próbek ze wszystkich możliwych danych rzeczywistych w celu utworzenia danych treningowych, pobieranie próbek ze zbioru danych w celu utworzenia paczek treningowych, walidacyjnych i testowych, czy też pobieranie próbek ze wszystkich możliwych zdarzeń, które mają miejsce w systemie uczenia maszynowego w celu ich monitorowania. W tym rozdziale skoncentrujemy się na metodach próbkowania umożliwiających tworzenie danych treningowych. Takie metody mogą być jednak stosowane również na innych etapach cyklu życia projektu uczenia maszynowego.

Próbkowanie jest konieczne w wielu przypadkach. Jednym z nim jest sytuacja, w której nie masz dostępu do wszystkich możliwych danych ze świata rzeczywistego. Wówczas do trenowania modelu możesz użyć podzbioru utworzonego za pomocą wybranej metody próbkowania. Może się także zdarzyć, że nie będziesz potrafił przetworzyć wszystkich danych, do których masz dostęp, ponieważ wymaga to poświęcenia zbyt dużej ilości czasu lub zasobów. Należy więc przeprowadzić próbkowanie, by wygenerować mniejszy podzbiór, który da się już przetworzyć. W wielu innych przypadkach próbkowanie jest pomocne, ponieważ pozwala wykonać zadanie szybciej i taniej. Załóżmy, że chcemy wdrożyć nowy model. Zanim zaczniemy go trenować na wszystkich danych, możemy przeprowadzić szybki eksperyment na niewielkim podzbiorze, aby sprawdzić, czy warto go rozwijać[1].

Zrozumienie różnych metod próbkowania i sposobu ich wykorzystania może, po pierwsze, pomóc uniknąć potencjalnych błędów, a po drugie, ułatwić wybranie metod, które poprawiają wydajność próbkowanych danych.

Istnieją dwa rodzaje próbkowania — **próbkowanie nieprobabilistyczne** i **próbkowanie losowe**. Rozpoczniemy od metod próbkowania nieprobabilistycznego, a następnie przedstawimy kilka popularnych metod próbkowania losowego.

## Próbkowanie nieprobabilistyczne

Z próbkowaniem nieprobabilistycznym mamy do czynienia wówczas, gdy wybór danych nie jest oparty na żadnych kryteriach prawdopodobieństwa. Oto niektóre z kryteriów doboru próbki nieprobabilistycznej:

*Próbkowanie oparte na wygodzie*

Próbki danych są wybierane na podstawie ich dostępności. Popularność tej metody wynika po prostu z jej wygody.

[1] Niektórzy Czytelnicy mogliby stwierdzić, że to podejście nie jest odpowiednie w przypadku skomplikowanych modeli, ponieważ działają one z dużymi, a nie małymi ilościami danych. W tym przypadku nadal należy jednak eksperymentować z paczkami danych o różnych rozmiarach, aby się dowiedzieć, jaki wpływ ma rozmiar zbioru na działanie modelu.



Kup książkę Poleć książkę

*Próbkowanie oparte na zasadzie działania „kuli śnieżnej”*

Kolejne próbki są wybierane na podstawie istniejących. Na przykład aby zebrać informacje o poprawnych kontach na Twitterze bez dostępu do jego baz danych, powinieneś zacząć od niewielkiego, początkowego zbioru, następnie zająć się kontami, które są obserwowane, itd.

*Próbkowanie na podstawie osądu*

Eksperci decydują o tym, jakie próbki należy uwzględnić.

*Próbkowanie na podstawie limitów*

Wybierasz próbki bez żadnej randomizacji na podstawie limitów dla pewnych wycinków danych. Na przykład podczas tworzenia ankiety będziesz chciał otrzymać 100 odpowiedzi dla każdej z grup wiekowych: poniżej 30 lat, między 30 a 60 lat i powyżej 60 lat, niezależnie od rzeczywistego rozkładu wieku.

Próbki wybrane według kryteriów nieprobabilistycznych nie są reprezentatywne dla rzeczywistych danych, więc dlatego są obarczone błędami selekcji[2]. Z tego powodu mógłbyś więc stwierdzić, że taki wybór danych służących do trenowania modeli uczenia maszynowego jest złym pomysłem. Masz rację. Niestety, w wielu przypadkach podczas wyboru danych dla modeli ML stosuje się metody, które są wygodne.

Jednym z przykładów jest modelowanie języka. Modele językowe są często trenowane nie przy użyciu danych reprezentatywnych dla wszystkich możliwych tekstów, ale takich, które można łatwo zgromadzić, korzystając z dostępnych źródeł (Wikipedia, Common Crawl lub Reddit).

Innym przykładem są dane do analizy emocjonalnej treści ogólnych. Wiele z tych danych jest pobieranych ze źródeł zawierających gotowe etykiety (oceny), takich jak recenzje na stronach IMDB i Amazon. Gotowe zbiory danych są następnie wykorzystywane do innych zadań związanych z analizą emocjonalną. Recenzje na stronach IMDB i Amazon są generowane przez użytkowników, którzy lubią się dzielić swoimi opiniami w internecie. Niekoniecznie więc są reprezentatywne dla osób, które nie mają dostępu do sieci lub po prostu niechętnie publicznie udostępniają swoje uwagi.

Trzecim przykładem są dane służące do trenowania pojazdów autonomicznych. Początkowo takie dane pochodziły w dużej mierze z dwóch obszarów: Phoenix w Arizonie (ze względu na łagodne przepisy) oraz Bay Area w Kalifornii (ponieważ ma tu swoje siedziby wiele firm konstruujących samochody autonomiczne). Oba obszary charakteryzują się słoneczną pogodą. W 2016 roku firma Waymo otworzyła biura w mieście Kirkland w stanie Waszyngton, ponieważ występuje tam wiele dni deszczowych[3]. Nadal jednak istnieje o wiele więcej danych dotyczących używania pojazdów autonomicznych podczas słonecznej pogody niż w trakcie opadów deszczu czy śniegu.

Próbkowanie nieprobabilistyczne może być szybkim i łatwym sposobem na zebranie wstępnych danych, które pozwolą rozpocząć projekt. Aby jednak uzyskać wiarygodny model, powinieneś użyć próbkowania losowego, które przeanalizujemy w następnej kolejności.

---

[2] James J. Heckman, *Sample Selection Bias as a Specification Error*, „Econometrica”, 47, nr 1, styczeń 1979, s. 153 – 161, *https://oreil.ly/I5AhM*.

[3] Rachel Lerman, *Google Is Testing Its Self-Driving Car in Kirkland*, „Seattle Times”, 3 lutego 2016, *https://oreil.ly/3IA1V*.



Kup książkę Poleć książkę

## Proste próbkowanie losowe

Najprostsza forma próbkowania losowego polega na tym, że każdy element danych ma takie samo prawdopodobieństwo bycia wybranym. Na przykład wybierasz losowo 10% populacji[4], dając wszystkim elementom równą, 10-procentową szansę bycia wybranym.

Zaleta tej metody polega na tym, że jest łatwa do wdrożenia. Wadą jest to, że rzadkie kategorie danych mogą się nie pojawić w Twoim wyborze. Rozważ przypadek, w którym klasa pojawia się tylko w 0,01% danych. Jeśli losowo wybierzesz 1% tych danych, próbki tej rzadkiej klasy prawdopodobnie nie zostaną uwzględnione. Modele wytrenowane za pomocą uzyskanych danych mogą po prostu stwierdzić, że taka kategoria nie istnieje.

## Próbkowanie warstwowe

Aby uniknąć wad prostego próbkowania losowego, mógłbyś najpierw podzielić populację na grupy, które Cię interesują, a następnie z każdej z nich pobierać oddzielnie próbki. Na przykład aby pobrać 1% danych ze zbioru, który ma dwie klasy (A i B), można pobrać 1% próbek z klasy A i tyle samo z klasy B. Dzięki temu, niezależnie od tego, jak rzadko występuje klasa A lub B, masz pewność, że jej próbki zostaną uwzględnione. Każda grupa jest nazywana warstwą, a sama metoda próbkowaniem warstwowym.

Jedną z wad tej metody próbkowania jest to, że nie zawsze jest ona możliwa — na przykład nie można jej zastosować, gdy nie da się podzielić danych na grupy. Szczególna trudność pojawia się wówczas, gdy jedna próbka może należeć do wielu grup, na przykład podczas rozwiązywania problemów z wieloma etykietami[5]. W takim przypadku próbka może należeć zarówno do klasy A, jak i B.

## Próbkowanie ważone

Podczas próbkowania ważonego każdej próbce jest nadawana określona waga, która określa prawdopodobieństwo jej wyboru. Na przykład, jeśli masz trzy próbki, A, B i C, a chciałbyś, by zostały wybrane z prawdopodobieństwem 50%, 30% i 20%, możesz nadać im wagi 0,5, 0,3 i 0,2.

Metoda ta pozwala na wykorzystanie wiedzy o dziedzinie. Jeśli przykładowo wiesz, że pewna podgrupa danych, zawierająca bardziej aktualne informacje, jest bardziej wartościowa dla Twojego modelu, a więc powinna mieć większą szansę na wybór, możesz nadać jej wyższą wagę.

Takie próbkowanie jest również przydatne w przypadku, gdy dane, które masz, mają inny rozkład niż dane realne. Załóżmy, że w Twoich danych czerwone próbki stanowią 25%, a niebieskie 75%. Wiesz jednak, że w świecie rzeczywistym kolory czerwony i niebieski mają równe prawdopodobieństwo wystąpienia. W takim przypadku możesz więc nadać czerwonym próbkom wagi trzy razy wyższe niż niebieskim.

[4] Słowo „populacja" odnosi się tutaj do „populacji statystycznej" (*https://oreil.ly/w7GDX*), czyli (potencjalnie nieskończonego) zbioru wszystkich możliwych próbek, które można pobrać

[5] Problemy z wieloma etykietami dotyczą danych, które mogą mieć wiele etykiet.

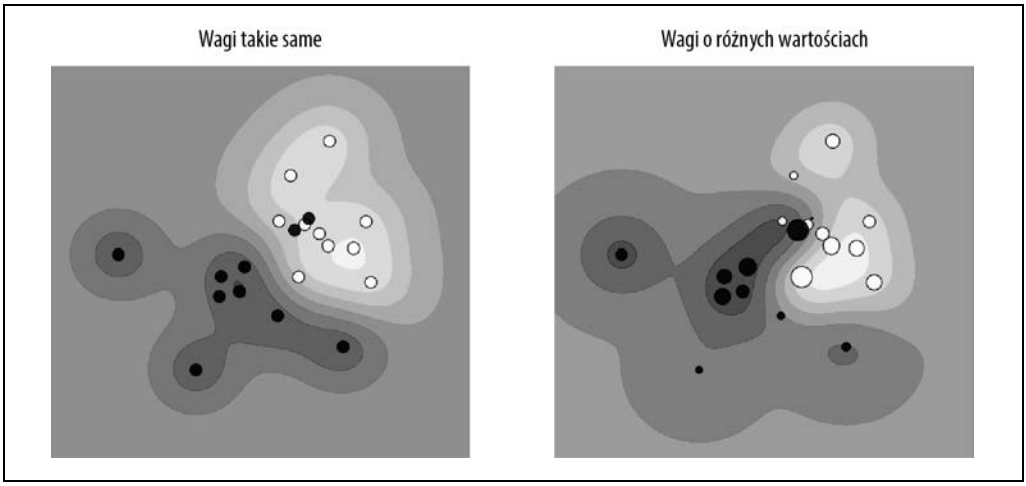


Kup książkę Poleć książkę

W Pythonie możesz wykonać ważone próbkowanie za pomocą metody `random.choices`:

```
# Spraw, by elementy 1, 2, 3, 4 miały po 20% szans na wybór,
# a elementy 100 i 1000 jedynie po 10%.
import random
random.choices(population=[1, 2, 3, 4, 100, 1000],
               weights=[0.2, 0.2, 0.2, 0.2, 0.1, 0.1],
               k=2)
# Jest to równoważne następującej instrukcji:
random.choices(population=[1, 1, 2, 2, 3, 3, 4, 4, 100, 1000], k=2)
```

W uczeniu maszynowym powszechnie używaną koncepcją, która jest ściśle związana z próbkowaniem ważonym, są wagi próbek. Próbkowanie ważone jest stosowane w celu wyboru próbek niezbędnych do trenowania modelu, podczas gdy wagi próbek są wykorzystywane do określania „znaczenia" lub „ważności" danych treningowych. Próbki o wyższych wagach bardziej wpływają na funkcję straty. Zmiana wag próbek może znacząco wpłynąć na obszar decyzyjny modelu, jak pokazano na rysunku 4.1.




*Rysunek 4.1. Wagi próbek mogą wpływać na obszar podejmowania decyzji. Po lewej stronie przedstawiono sytuację, w której wszystkie próbki mają równe wagi, natomiast po prawej taką, w której próbkom nadano różne wagi. Źródło: scikit-learn*[6]

## Próbkowanie do rezerwuaru

Próbkowanie do rezerwuaru to fascynujący algorytm, który jest szczególnie przydatny, gdy masz do czynienia z danymi strumieniowymi, czyli takimi, jakie często pojawiają się w produkcji.

Wyobraź sobie, że masz przychodzący strumień tweetów, a chciałbyś pobrać ich próbkę o wielkości *k*, aby przeprowadzić analizę lub wytrenować model. Nie wiesz, ile jest tweetów, ale zdajesz sobie sprawę, że nie możesz ich wszystkich zmieścić w pamięci. Oznacza to, że nie znasz z góry prawdopodobieństwa, dla którego dany tweet powinien zostać wybrany. Chcesz jednak zapewnić, że:

[6] *SVM: Weighted Samples*, scikit-learn, *https://oreil.ly/BDqbk*.



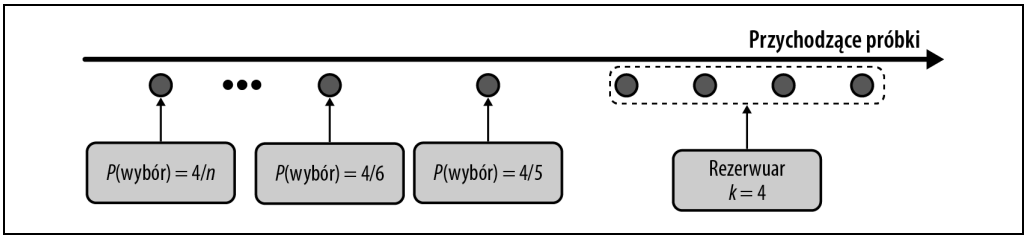
Kup książkę Poleć książkę

- Każdy tweet będzie miał równe prawdopodobieństwo bycia wybranym.
- Możesz zatrzymać algorytm w dowolnym momencie, a tweety będą próbkowane z poprawnym prawdopodobieństwem.

Jednym z rozwiązań tego problemu jest próbkowanie do rezerwuaru. Algorytm ten wykorzystuje rezerwuar, którym może być tablica, i składa się z trzech etapów:

1. Umieść w rezerwuarze pierwsze $k$ elementów.
2. Dla każdego przychodzącego $n$-tego elementu wygeneruj liczbę losową $i$ taką, że $1 \leq i \leq n$.
3. Jeśli $1 \leq i \leq k$, zastąp $i$-ty element w rezerwuarze elementem $n$-tym. W przeciwnym razie nie rób nic.

Oznacza to, że każdy przychodzący $n$-ty element będzie mógł się znaleźć w rezerwuarze z prawdopodobieństwem równym $\frac{k}{n}$. Można również udowodnić, że każdy element występujący już w rezerwuarze został tam umieszczony z prawdopodobieństwem równym $\frac{k}{n}$. Oznacza to, że wszystkie próbki mają równą szansę na wybór. Nawet jeśli zatrzymamy algorytm w dowolnym momencie, będziemy pewni, że próbki znajdujące się już w rezerwuarze zostały pobrane z poprawnym prawdopodobieństwem. Na rysunku 4.2 przedstawiono schemat działania próbkowania do rezerwuaru.




*Rysunek 4.2. Wizualizacja działania próbkowania do rezerwuaru*

## Próbkowanie istotnościowe

Próbkowanie istotnościowe jest jedną z najważniejszych metod próbkowania nie tylko w przypadku uczenia maszynowego. Pozwala próbkować dane o określonym rozkładzie, gdy istnieje tylko dostęp do innego rozkładu.

Wyobraź sobie, że musisz pobrać próbkę $x$ z rozkładu $P(x)$, którego uzyskanie jest szczególnie kosztowne i czasochłonne. Być może też taki rozkład nie pozwoli na jego późniejsze próbkowanie. Jednakże dysponujesz innym rozkładem $Q(x)$, który jest dużo łatwiejszy do próbkowania. Próbkujemy więc $x$ z $Q(x)$, a następnie określamy wagę próbki równą $\frac{P(x)}{Q(x)}$. Rozkład $Q(x)$ jest nazywany **rozkładem wnioskowanym** lub **rozkładem istotnościowym**. Może on być dowolnym rozkładem, dopóki $Q(x) > 0$, a $P(x) \neq 0$. Poniższe równanie dowodzi, że zgodnie z oczekiwaniem próbka $x$ z $P(x)$ jest równa próbce $x$ z $Q(x)$, dla której zdefiniowano wagę równą $\frac{P(x)}{Q(x)}$:

$$E_{P(x)}[x] = \sum_x P(x)x = \sum_x Q(x)x \frac{P(x)}{Q(x)} = E_{Q(x)}\left[x\frac{P(x)}{Q(x)}\right]$$



Kup książkę Poleć książkę

Jednym z przykładów użycia próbkowania istotnościowego w uczeniu maszynowym jest uczenie ze wzmocnieniem oparte na polityce. Weźmy pod uwagę przypadek, w którym chciałbyś uaktualnić swoją politykę. Należy oszacować funkcje wartości w nowej polityce, ale wyznaczenie sumarycznych nagród za podjęcie akcji mogłoby być kosztowne, ponieważ wymagałoby rozważenia wszystkich możliwych wyników aż do końca horyzontu czasowego. Jeśli jednak nowa polityka jest dość podobna do wcześniejszej, można obliczyć sumaryczne nagrody na podstawie starszej, a następnie nadać im wagi zgodnie z nową. Nagrody ze starszej polityki tworzą rozkład wnioskowany.

# Etykietowanie

Pomimo tego, że specjaliści dążą do szerokiego zastosowania systemów nienadzorowanego uczenia maszynowego, większość działających obecnie rozwiązań ML wykorzystuje modele nadzorowane, co oznacza, że do nauki wymagają danych oznaczonych etykietami. Wydajność modelu uczenia maszynowego nadal w dużej mierze zależy od jakości i ilości zaetykietowanych danych, za pomocą których jest trenowany.

Andrej Karpathy, dyrektor ds. sztucznej inteligencji w firmie Tesla, podczas rozmowy ze studentami przytoczył następującą anegdotę. Gdy zdecydował się utworzyć wewnętrzny zespół, który miał się zajmować zagadnieniami związanymi z etykietowaniem, rekruter zadał pytanie, jak długo taki zespół będzie potrzebny. Andrej odpowiedział: „Jak długo musimy zatrudniać zespół inżynierów?”. Etykietowanie danych było kiedyś traktowane jako zadanie pomocnicze, a obecnie stało się podstawową funkcją wielu zespołów uczenia maszynowego wdrażających rozwiązania w produkcji.

W tym podrozdziale przeanalizujemy kwestię, jaką jest uzyskanie etykiet dla danych. Najpierw omówimy metodę, która zwykle pojawia się jako pierwsza w głowie danetyków — chodzi o etykietowanie ręczne. Następnie przeanalizujemy zadania z etykietami naturalnymi, czyli takie, w których etykiety mogą zostać wywnioskowane z systemu bez konieczności dokonywania adnotacji przez człowieka. Wreszcie zastanowimy się, co można zrobić, gdy brakuje etykiet naturalnych i ręcznych.

## Etykiety nadawane ręcznie

Każdy, kto kiedykolwiek musiał obsługiwać dane produkcyjne, prawdopodobnie doświadczył tego na poziomie emocjonalnym — ręczne generowanie etykiet dla danych jest trudne z bardzo wielu powodów. Po pierwsze, może być kosztowne, zwłaszcza jeśli wymagana jest wiedza merytoryczna. Aby sklasyfikować, czy komentarz jest spamem, możesz za pomocą platformy crowdsourcingowej znaleźć 20 osób i przeszkolić je w ciągu 15 minut do etykietowania danych. Jeśli jednak chcesz oznaczyć zdjęcia rentgenowskie klatki piersiowej, musisz znaleźć radiologów z uprawnieniami, których czas jest ograniczony i kosztowny.

Po drugie, etykietowanie ręczne stanowi zagrożenie dla prywatności danych. Oznacza ono, że ktoś musi zajrzeć do danych, co nie zawsze jest możliwe, jeśli istnieją ścisłe wymagania dotyczące prywatności. Na przykład w celu etykietowania nie można po prostu wysłać dokumentacji medycznej pacjentów lub poufnych informacji finansowych do firmy zewnętrznej. W wielu przypadkach dane nie mogą nawet opuścić organizacji i konieczne może być zatrudnienie osób, które zajmą się etykietowaniem danych na miejscu.



Kup książkę Poleć książkę

Po trzecie, etykietowanie ręczne jest powolne. Dokładne przepisanie wypowiedzi na poziomie fonetycznym może trwać 400 razy dłużej niż czas jej trwania[7]. Jeśli więc chciałbyś zaetykietować 1 godzinę wypowiedzi, mogłoby to zająć 400 godzin lub prawie trzy miesiące. Moi koledzy musieli czekać prawie rok, aby uzyskać poprawne etykiety w badaniu mającym na celu wykorzystanie uczenia maszynowego do pomocy w klasyfikowaniu raka płuc na podstawie zdjęć rentgenowskich.

Powolne etykietowanie pogarsza cykl projektowy i sprawia, że Twój model gorzej adaptuje się do zmieniających się środowisk i wymagań. Jeśli zmianie ulegną zadanie lub dane, przed aktualizacją modelu będziesz musiał czekać na ponowne przeprowadzenie procesu etykietowania. Załóżmy, że masz model analizy emocjonalnej sprawdzający tweety, w których wspomina się o marce Twojej firmy. Model wykorzystuje tylko dwie klasy: OCENA_NEGATYWNA i OCENA_POZYTYWNA. Jednak po wdrożeniu zespół PR zdaje sobie sprawę, że najwięcej szkód generują tweety tworzone przez rozgniewanych użytkowników. Musisz więc uaktualnić model analizy emocjonalnej i uzupełnić go o trzecią klasę — UŻYTKOWNIK_ROZGNIEWANY. Aby to zrobić, będziesz musiał ponownie przyjrzeć się danym, by sprawdzić, które istniejące przykłady treningowe powinny zostać oznaczone jako UŻYTKOWNIK_ROZGNIEWANY. Jeśli nie masz wystarczająco dużo takich tweetów, będziesz musiał zebrać więcej danych. Im dłużej będzie trwał ten proces, tym bardziej pogorszy się wydajność istniejącego modelu.

### Mnogość etykiet

Aby uzyskać wystarczającą ilość zaetykietowanych danych, firmy często muszą wykorzystywać informacje pochodzące z wielu źródeł i polegać na osobach, które mają różne poziomy wiedzy. Takie źródła danych i osoby tworzące etykiety charakteryzują się różnymi poziomami dokładności. Prowadzi to do powstawania problemu niejednoznaczności lub mnogości etykiet. Co zrobić, gdy istnieje wiele sprzecznych etykiet opisujących dany element?

Rozważmy proste zadanie rozpoznawania encji (podmiotów). Prosisz trzy osoby o zaetykietowanie wszystkich encji, które potrafią znaleźć w następującym zdaniu:

> Darth Sidious, zwany w skrócie Imperatorem, był Mrocznym Lordem Sithów, który panował nad galaktyką jako Galaktyczny Imperator Pierwszego Imperium Galaktycznego.

Uzyskane wyniki przedstawiono w tabeli 4.1. Każdy z komentatorów znalazł inne encje. Na jakich wynikach powinno się oprzeć trenowanie modelu? Model wytrenowany na danych zaetykietowanych przez komentatora 1. będzie działał zupełnie inaczej niż model wytrenowany na danych zaetykietowanych przez komentatora 2.

Tego typu nieporozumienia pojawiają się niezwykle często. Im wyższy wymagany poziom wiedzy z danej dziedziny, tym większe prawdopodobieństwo wystąpienia nieporozumień podczas etykietowania[8]. Jak rozwiązać problem i wybrać odpowiedź odpowiadającą prawdzie, jeśli jeden z ekspertów uważa, że należy przypisać etykietę A, podczas gdy inny jest pewien, iż musi być B? Jeśli nie można dojść do porozumienia w sprawie etykiety, to czym jest w ogóle wydajność na poziomie ludzkim?

---

[7] Xiaojin Zhu, *Semi-Supervised Learning with Graphs*, praca doktorska, Carnegie Mellon University, 2005, *https://oreil.ly/VYy4C*.

[8] Jeśli coś jest oczywiste, w celu etykietowania nie potrzebujesz wykorzystywać wiedzy o dziedzinie.



Kup książkę Poleć książkę

*Tabela 4.1. Encje wykryte przez różnych komentatorów mogą się od siebie bardzo różnić*

| Komentator | Liczba wykrytych encji | Wynik |
|---|---|---|
| 1 | 3 | [*Darth Sidious*], zwany w skrócie Imperatorem, był [*Mrocznym Lordem Sithów*], który panował nad galaktyką jako [*Galaktyczny Imperator Pierwszego Imperium Galaktycznego*]. |
| 2 | 6 | [*Darth Sidious*], zwany w skrócie [*Imperatorem*], był [*Mrocznym Lordem*] [*Sithów*], który panował nad galaktyką jako [*Galaktyczny Imperator*] [*Pierwszego Imperium Galaktycznego*]. |
| 3 | 4 | [*Darth Sidious*], zwany w skrócie [*Imperatorem*], był [*Mrocznym Lordem Sithów*], który panował nad galaktyką jako [*Galaktyczny Imperator Pierwszego Imperium Galaktycznego*]. |

Aby zminimalizować różnice zdań wśród komentatorów, należy najpierw dokładnie zdefiniować problem. Na przykład w poprzednim zadaniu rozpoznawania encji niektóre nieporozumienia mogłyby zostać wyeliminowane, gdybyśmy wyjaśnili, że w przypadku wielu możliwych fraz należy wybrać tę, która ma największą liczbę znaków. Oznacza to, że poprawną odpowiedzią będzie encja *Galaktyczny Imperator Pierwszego Imperium Galaktycznego* zamiast dwóch encji *Galaktyczny Imperator* i *Pierwsze Imperium Galaktyczne*. Poza tym musisz o tym poinformować komentatorów już podczas szkolenia, aby się upewnić, że zrozumieją zasady.

### Pochodzenie danych

Bezkrytyczne wykorzystywanie danych z wielu źródeł generowanych za pomocą różnych komentatorów, a także brak kontroli jakościowej mogą być powodem tajemniczych awarii modelu. Rozważ przypadek, w którym wytrenowałeś umiarkowanie dobry model za pomocą 100 000 próbek danych. Twoi inżynierowie uczenia maszynowego są pewni, że więcej danych poprawi jego wydajność, więc wydajesz dużo pieniędzy i zatrudniasz osoby, które będą etykietować nowy milion próbek.

Jednak po kolejnym wytrenowaniu modelu jego wydajność spada. Powodem jest to, że nowy milion próbek został przekazany komentatorom, którzy zaetykietowali dane ze znacznie mniejszą dokładnością niż w przypadku danych wcześniejszych. Taki problem może być szczególnie trudny do rozwiązania, jeśli dane zostały już wymieszane ze sobą i nie możesz odróżnić nowych od starych.

Dobrym wzorcem postępowania jest oznaczanie pochodzenia każdej próbki danych oraz jej etykiet. Jest to technika znana pod nazwą **pochodzenia danych**. Pomaga ona zarówno oznaczać potencjalne błędy w danych, jak i debugować modele. Na przykład jeśli Twój model zawodzi głównie na ostatnio pozyskanych próbkach danych, powinieneś sprawdzić, jak je pozyskano. Niejednokrotnie odkrywaliśmy, że problem nie był związany z modelem, ale wynikał z powodu niezwykle dużej liczby błędnych etykiet w danych, które niedawno uzyskaliśmy.

## Etykiety naturalne

Etykietowanie ręczne nie jest jedyną dostępną metodą. Być może będziesz miał szczęście brać udział w projektach wykorzystujących etykiety naturalne jako źródła prawdy. Prognozy modelu używanego w takim projekcie mogą być przynajmniej częściowo automatycznie oceniane przez



Kup książkę Poleć książkę

system. Weźmy pod uwagę model, który dla wybranej trasy w usłudze Mapy Google szacuje czas przybycia. Jeśli użytkownik pojedzie tą trasą, pod koniec podróży usługa Mapy Google będzie wiedzieć, jak długo trwała podróż, a więc może ocenić dokładność prognozy czasu przybycia. Innym przykładem jest przewidywanie cen akcji. Jeśli Twój model przewiduje, jak zmieni się cena akcji w ciągu najbliższych dwóch minut, po tym czasie możesz porównać prognozę z ceną rzeczywistą.

Kanonicznym przykładem zadań z etykietami naturalnymi są systemy rekomendujące. Ich celem jest polecanie użytkownikom przedmiotów, które są dla nich istotne. Kliknięcie (lub nie) polecanego przedmiotu może być postrzegane jako informacja zwrotna dla rekomendacji. Rekomendacja, która się zakończyła kliknięciem, może zostać uznana za dobrą (a więc zostanie zaetykietowana jako OCENA_POZYTYWNA), natomiast ta, która w ciągu pewnego czasu (na przykład 10 minut) nie została kliknięta, może zostać uznana za złą (i zaetykietowana jako OCENA_NEGATYWNA).

Wiele problemów można zdefiniować w taki sposób, aby należały do klasy zadań z rekomendacjami. Na przykład można określić zadanie prognozowania współczynnika klikalności reklam jako polecanie najbardziej odpowiednich reklam użytkownikom na podstawie ich historii aktywności i profili. Etykiety naturalne, które są wnioskowane z zachowań użytkowników, takich jak kliknięcia i oceny, są również znane pod nazwą etykiet behawioralnych.

Nawet jeśli w projekcie nie możesz wykorzystywać etykiet naturalnych, być może da się skonfigurować system w taki sposób, który pozwoli Ci zebrać informacje zwrotne dotyczące modelu. Na przykład jeśli tworzysz system tłumaczenia maszynowego, taki jak Tłumacz Google, możesz społeczności udostępnić opcję, która pozwoli na przesyłanie alternatywnych tłumaczeń. Mogą one następnie zostać użyte do trenowania modeli w kolejnej iteracji (chociaż zapewne będziesz chciał najpierw przejrzeć te sugerowane tłumaczenia). Ranking kanałów informacyjnych nie jest zadaniem wykorzystującym etykiety naturalne, ale dzięki uzupełnieniu każdej wiadomości o przycisk *Lubię to* i inne opcje Facebook jest w stanie zebrać informacje zwrotne o swoim algorytmie rankingowym.

Zadania z etykietami naturalnymi są dość powszechnie spotykane w przemyśle. Po przeprowadzeniu badania z udziałem 86 firm stwierdziłam, że 63% z nich wykorzystuje projekty z etykietami naturalnymi (rysunek 4.3). Nie oznacza to jednak, że 63% zadań wykorzystujących uczenie maszynowe stosuje etykiety naturalne. Bardziej prawdopodobne jest to, że firmy uważają, iż łatwiej i taniej jest najpierw zacząć od zadań, które mają etykiety naturalne.

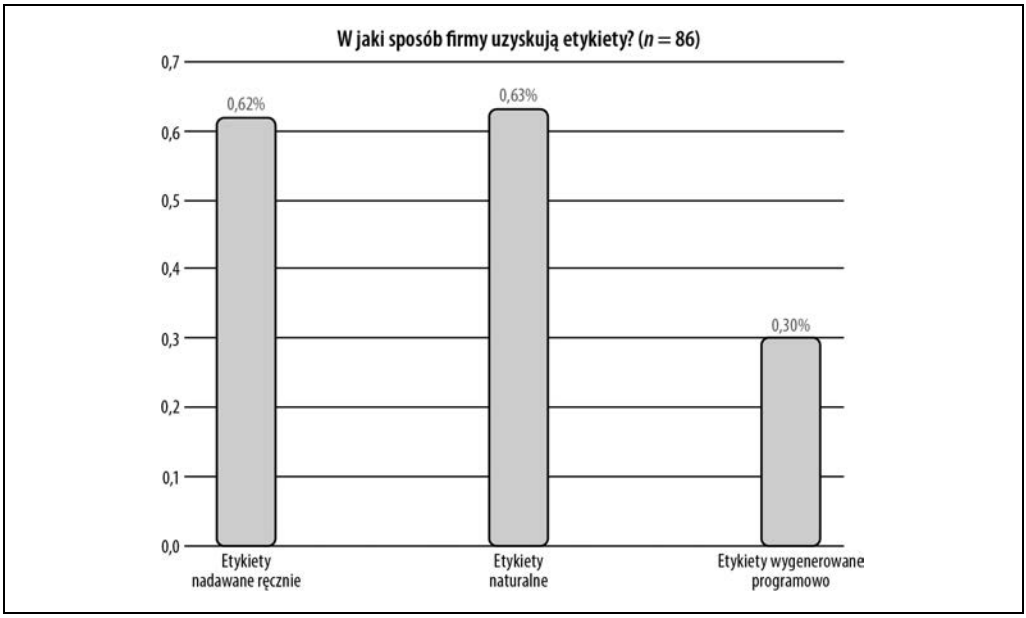
Jeśli we wcześniejszym przykładzie rekomendacja nie zostanie kliknięta w określonym czasie, może zostać uznana za negatywną. W tym przypadku uzyskujemy **etykietą ukrytą**, ponieważ jest ona domyślnie stosowana z powodu braku oceny pozytywnej. Jej przeciwieństwem są **etykiety jawne**, w przypadku których użytkownicy wyraźnie prezentują swoje opinie o rekomendacjach poprzez przypisanie im niskich ocen lub ich obniżenie.

### Długość pętli sprzężenia zwrotnego

W przypadku zadań z etykietami naturalnymi można określić czas, jaki upływa od podania prognozy do momentu przekazania informacji zwrotnej na jej temat. Czas ten jest zwany **długością pętli sprzężenia zwrotnego**. W zadaniach z krótkimi pętlami sprzężenia zwrotnego etykiety są



Kup książkę Poleć książkę




*Rysunek 4.3. 63% firm realizuje projekty z etykietami naturalnymi. Suma wszystkich wartości nie jest równa 1, ponieważ dana firma może korzystać z różnych źródeł etykiet*[9]

zazwyczaj dostępne w ciągu kilku minut. Wiele systemów rekomendacyjnych ma krótkie pętle sprzężenia zwrotnego. Jeśli polecane pozycje są produktami z portalu Amazon lub osobami, które warto śledzić na Twitterze, czas pomiędzy rozpoczęciem rekomendowania pozycji a jej kliknięciem (jeśli w ogóle zostanie kliknięta) jest krótki.

Jednak nie wszystkie systemy rekomendacji charakteryzują się minutowymi pętlami sprzężenia zwrotnego. Jeśli przetwarzane są bardziej złożone treści, takie jak wpisy na blogu, artykuły lub filmy na YouTubie, pętla zwrotna może trwać godziny. Jeśli wdrożysz system polecający ubrania dla użytkowników (taki jak ten, który wykorzystuje usługa Stitch Fix), nie otrzymasz informacji zwrotnej, dopóki dana osoba nie otrzyma produktu i nie przymierzy go, co może nastąpić nawet po kilku tygodniach.

Wybór odpowiedniej długości pętli sprzężenia zwrotnego wymaga dokładnego rozważenia, ponieważ wiąże się z kompromisem między szybkością a dokładnością. Niewielka długość oznacza, że można szybciej przechwytywać etykiety, co pozwala wykorzystać je do wykrywania problemów związanych z modelem i szybkiego ich rozwiązywania. Znaczy to jednak również, że można przedwcześnie zaetykietować rekomendację jako negatywną, zanim jeszcze zostanie kliknięta.

Niezależnie od tego, jaka długość zostanie ustalona, etykiety negatywne nadal mogą się przedwcześnie pojawiać. Badanie przeprowadzone przez zespół Ads na początku 2021 roku na Twitterze wykazało, że nawet jeśli większość kliknięć reklam odbywa się w ciągu pierwszych pięciu minut,

[9] Etykiety generowane programowo przeanalizujemy w punkcie „Nadzór słaby".



Kup książkę Poleć książkę

## Różne rodzaje informacji zwrotnych otrzymywanych od użytkowników

Jeśli chcesz wyodrębnić etykiety z informacji zwrotnych uzyskanych od użytkowników, powinieneś pamiętać, że istnieją różne rodzaje takich informacji. Mogą one być generowane w wielu miejscach aplikacji i różnić się od siebie objętością, siłą sygnału i długością pętli sprzężenia zwrotnego.

Dla przykładu rozważmy aplikację handlu elektronicznego podobną do tej, którą udostępnia Amazon. Użytkownik może przekazać takie informacje zwrotne jak kliknięcie rekomendowanego produktu, dodanie go do koszyka, zakup, ocena, pozostawienie recenzji i zwrot towaru, który został wcześniej kupiony.

Klikanie produktów odbywa się dużo szybciej i częściej (a zatem generuje większy wolumen) niż ich kupowanie. Jednak zakup jest znacznie silniejszym sygnałem, świadczącym o tym, że użytkownik lubi dany produkt.

Wiele firm podczas tworzenia systemów rekomendacji produktów koncentruje się na optymalizacji aplikacji pod kątem kliknięć, które umożliwiają uzyskanie większego wolumenu informacji zwrotnych służących do oceny ich modeli. Niektóre jednak skupiają wysiłki na obsłudze zakupów, które zwracają sygnał silniejszy, a jednocześnie bardziej skorelowany ze wskaźnikami biznesowymi (na przykład przychód ze sprzedaży produktu). Oba podejścia są ważne. Nie ma jednoznacznej odpowiedzi, jaki rodzaj informacji zwrotnej powinieneś zoptymalizować dla danego przypadku użycia. W każdej sytuacji wymagane jest przeprowadzenie dyskusji pomiędzy wszystkimi zainteresowanymi stronami.

niektóre zdarzają się kilka godzin po tym, jak reklama została wyświetlana[10]. Oznacza to, że w tym przypadku istnieje tendencja do podawania niedoszacowanego rzeczywistego współczynnika klikalności. Jeśli zarejestrujesz tylko 1000 etykiet OCENA_POZYTYWNA, rzeczywista liczba kliknięć może być większa.

W przypadku zadań z długimi pętlami sprzężenia zwrotnego etykiety naturalne mogą się nie pojawiać przez tygodnie, a nawet miesiące. Przykładem takiego zadania jest wykrywanie oszustw. Przez pewien okres po dokonaniu transakcji użytkownicy mogą się spierać, czy jest ona oszustwem, czy też nie. Załóżmy, że klient zapoznał się z wyciągiem z karty kredytowej i zobaczył transakcję, której nie rozpoznał. W takim przypadku może ją zakwestionować w banku, a przez to przekazać informację zwrotną, by oznaczyć transakcję jako fałszywą. Typowy czas rozpatrywania wniosku wynosi od jednego do trzech miesięcy. Jeśli po upływie tego okresu użytkownik nie zgłosi zastrzeżeń, można założyć, że transakcja jest legalna.

Etykiety z długimi pętlami sprzężenia zwrotnego są wykorzystywane do informowania o wydajności modelu w kwartalnych lub rocznych raportach biznesowych. Nie są one jednak zbyt pomocne, jeśli chcesz jak najszybciej wykryć problemy związane z modelami. Przypuśćmy, że w Twoim modelu wykrywania oszustw wystąpi problem, którego rozwiązanie zajmie kilka miesięcy. Zanim model zacznie poprawnie działać, wszystkie fałszywe transakcje, na które on zezwolił, będą mogły spowodować bankructwo małej firmy.

[10] Sofia Ira Ktena, Alykhan Tejani, Lucas Theis, Pranay Kumar Myana, Deepak Dilipkumar, Ferenc Huszar, Steven Yoo i Wenzhe Shi, *Addressing Delayed Feedback for Continuous Training with Neural Networks in CTR Prediction*, arXiv, 15 lipca 2019, *https://oreil.ly/5y2WA*.



Kup książkę Poleć książkę

## Co zrobić w przypadku braku etykiet?

Ze względu na wyzwania związane z pozyskaniem wystarczającej liczby etykiet o wysokiej jakości wymyślono wiele metod pozwalających rozwiązać wynikające z tego problemy. W tym punkcie przeanalizujemy cztery z nich: nadzór słaby, półnadzór, uczenie transferowe i uczenie aktywne. Podsumowanie tych metod przedstawiono w tabeli 4.2.

*Tabela 4.2. Podsumowanie czterech metod pozwalających rozwiązać problem z brakiem etykiet*

| Metoda | Opis | Czy potrzebne źródło prawdy? |
|---|---|---|
| Nadzór słaby | Wykorzystuje (często zakłócone) heurystyki do generowania etykiet | Nie, ale zaleca się wykorzystanie niewielkiej liczby etykiet w celu opracowania heurystyki |
| Półnadzór | Wykorzystuje założenia strukturalne do generowania etykiet | Tak, niewielka liczba wstępnych etykiet służy do generowania kolejnych |
| Uczenie transferowe | Wykorzystuje wstępnie wytrenowane modele, specjalizujące się w rozwiązywaniu innych zadań | Nie w przypadku uczenia od zera. Tak w przypadku dostrajania, choć liczba wymaganych etykiet jest często znacznie mniejsza od tej, która byłaby potrzebna przy trenowaniu modelu od podstaw |
| Uczenie aktywne | Etykietuje próbki danych, które są najbardziej przydatne dla danego modelu | Tak |

### Nadzór słaby

Skoro etykietowanie ręczne jest problematyczne, czy możemy w ogóle zrezygnować z tak wygenerowanych etykiet? Jedną z metod, która zdobyła znaczną popularność, jest nadzór słaby. Wśród najpopularniejszych narzędzi o otwartych źródłach, wspierających nadzór słaby, znalazł się system Snorkel, opracowany w Stanford AI Lab[11]. Idea nadzoru słabego polega na tym, że przy etykietowaniu danych polegamy na heurystykach, które mogą być opracowane z wykorzystaniem specjalistycznej wiedzy. Na przykład lekarz może używać następujących heurystyk, aby zdecydować, czy przypadek pacjenta powinien zostać potraktowany jako nagły:

> Jeśli notatka sporządzona przez pielęgniarkę zawiera informację o poważnym schorzeniu, takim jak zapalenie płuc, przypadek pacjenta powinien zostać potraktowany priorytetowo.

Biblioteki takie jak Snorkel wykorzystują koncepcję **funkcji etykietującej** (ang. *labeling function*, w skrócie *LF*), czyli takiej, która koduje heurystykę. Heurystyka wynikająca z powyższego zdania może zostać wyrażona przez następującą funkcję:

```
def labeling_function(note):
   if "zapalenie płuc" in note:
     return "PRZYPADEK_PILNY"
```

[11] Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu i Christopher Ré, *Snorkel: Rapid Training Data Creation with Weak Supervision*, „Proceedings of the VLDB Endowment", 11, nr 3, 2017, s. 269 – 282, *https://oreil.ly/vFPjk*.



Kup książkę Poleć książkę

Funkcje etykietujące mogą kodować wiele różnych rodzajów heurystyk. Oto niektóre z nich:

*Heurystyka słów kluczowych*

Wykorzystywana w powyższym przykładzie.

*Wyrażenia regularne*

Na przykład jeśli sporządzona notatka pasuje lub nie pasuje do określonego wyrażenia regularnego.

*Przeglądanie bazy danych*

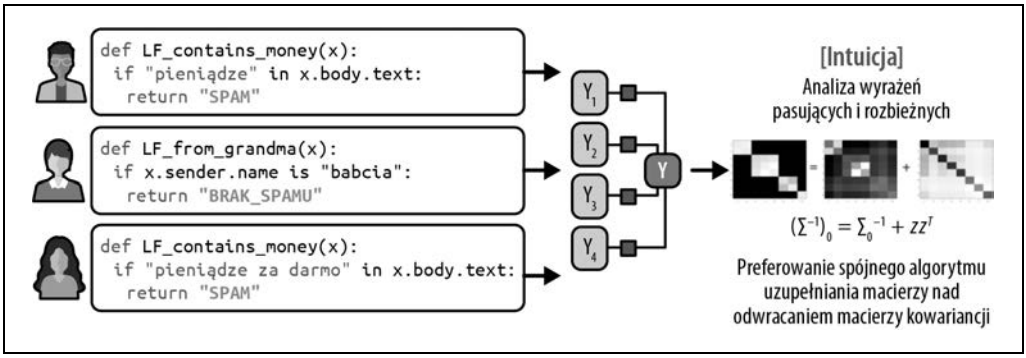
Na przykład jeśli notatka zawiera informację o jednostce wymienionej na liście niebezpiecznych chorób.

*Wyniki działania innych modeli*

Na przykład jeśli istniejący system zakwalifikuje sytuację jako PRZYPADEK_PILNY.

Po utworzeniu funkcji etykietujących można je zastosować do próbek, które powinno się zaetykietować.

Ponieważ funkcje etykietujące kodują heurystyki, które są zakłócone, takie będą również wygenerowane etykiety. Takie same dane mogą zostać przetworzone przez różne funkcje etykietujące, a w wyniku otrzymamy sprzeczne etykiety. Jedna funkcja może na przykład stwierdzić, że notatka pielęgniarki powinna zostać zaetykietowana jako PRZYPADEK_PILNY, a druga wręcz przeciwnie. Pewna heurystyka może być znacznie dokładniejsza niż inna — nie można jednak tego potwierdzić, ponieważ nie ma możliwości porównania z etykietami zawierającymi źródła prawdy. Aby uzyskać jak najbardziej poprawny zestaw etykiet, należy dla wszystkich funkcji etykietujących przeprowadzić operacje łączenia, usuwania zakłóceń i dopasowywania wag. Na rysunku 4.4 przedstawiono wysokopoziomowy schemat działania funkcji etykietujących.




*Rysunek 4.4. Wysokopoziomowy schemat łączenia funkcji etykietujących.*
*Źródło: utworzono na podstawie rysunku autorstwa Ratnera i in.*[12]

Teoretycznie w przypadku nadzoru słabego żadnych etykiet nie trzeba generować ręcznie. Aby się jednak dowiedzieć, jak dokładne są funkcje etykietujące, zalecane jest utworzenie niewielkiej liczby etykiet ręcznych. Mogą one pomóc odkryć wzorce w danych, co przyczyni się do ulepszenia funkcji etykietujących.

[12] Ratner i in., *Snorkel: Rapid Training Data Creation with Weak Supervision.*



Kup książkę Poleć książkę

Nadzór słaby może być szczególnie przydatny, gdy dane mają ścisłe wymagania dotyczące prywatności. Wystarczy przeanalizować niewielki, oczyszczony zestaw danych, aby utworzyć funkcje etykietujące, które można zastosować do pozostałej części zbioru.

Dzięki funkcjom etykietującym można wersjonować, ponownie wykorzystywać i udostępniać wiedzę o danym zagadnieniu. Wiedza specjalistyczna będąca w posiadaniu jednego zespołu może zostać zakodowana i wykorzystana przez inny. Jeśli ulegną zmianie dane lub wymagania, można będzie po prostu ponownie zastosować funkcje etykietujące do próbek. Podejście polegające na wykorzystaniu funkcji etykietujących do generowania etykiet dla danych jest również znane pod nazwą **etykietowania programowego**. W tabeli 4.3 przedstawiono niektóre zalety etykietowania programowego w porównaniu z etykietowaniem ręcznym.

*Tabela 4.3. Zalety etykietowania programowego w porównaniu z etykietowaniem ręcznym*

| Etykietowanie ręczne | Etykietowanie programowe |
|---|---|
| *Kosztowne* — zwłaszcza gdy wymagana jest specjalistyczna wiedza o danym zagadnieniu | *Oszczędzające koszty* — wiedza specjalistyczna może być wersjonowana, współdzielona i ponownie wykorzystywana w całej organizacji |
| *Brak prywatności* — konieczność przekazania danych osobom generującym etykiety | *Prywatność* — tworzenie funkcji etykietujących przy użyciu oczyszczonej próbki danych, a następnie wykorzystanie z innymi danymi bez ich analizowania |
| *Wolne* — wymagany czas wzrasta liniowo wraz z liczbą potrzebnych etykiet | *Szybkie* — łatwe skalowanie od tysiąca do miliona próbek |
| *Nieadaptacyjne* — każda zmiana wymaga ponownego etykietowania danych | *Adaptacyjne* — gdy pojawiają się zmiany, wystarczy ponownie zastosować funkcję etykietującą! |

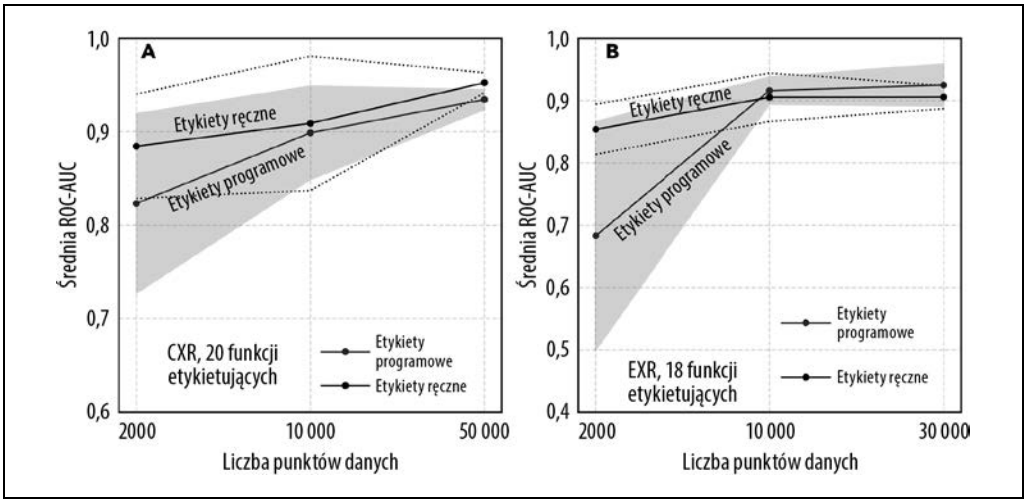
Oto studium przypadku pokazujące, że nadzór słaby sprawdza się dobrze w praktyce. Zgodnie z badaniem przeprowadzonym we współpracy ze szkołą Stanford Medicine[13], modele wytrenowane na podstawie etykiet wygenerowanych za pomocą metody nadzoru słabego, dla których funkcje etykietujące zostały napisane w ciągu ośmiu godzin przez jednego radiologa, miały porównywalną wydajność z takimi, które zostały wytrenowane na danych uzyskanych w wyniku procesu etykietowania ręcznego, trwającego prawie rok (rysunek 4.5). Oto dwa interesujące fakty dotyczące wyników eksperymentu. Po pierwsze, modele działały coraz lepiej przy większej ilości niezaetykietowanych danych, nawet bez konieczności zwiększania liczby funkcji etykietujących. Po drugie, funkcje etykietujące były ponownie wykorzystywane w różnych zadaniach. Badacze byli w stanie użyć sześciu takich samych funkcji etykietujących w zadaniach CXR (zdjęcia rentgenowskie klatki piersiowej) i EXR (zdjęcia rentgenowskie kończyn)[14].

[13] Jared A. Dunnmon, Alexander J. Ratner, Khaled Saab, Matthew P. Lungren, Daniel L. Rubin i Christopher Ré, *Cross-Modal Data Programming Enables Rapid Medical Machine Learning*, „Patterns", 1, nr 2, 2020, 100019, *https://oreil.ly/nKt8E*.

[14] W dwóch zadaniach CXR i EXR wykorzystanych zostało odpowiednio 20 i 18 funkcji etykietujących. Wiem z praktyki, że zespoły używały setek funkcji etykietujących w przypadku każdego z zadań.



Kup książkę Poleć książkę




*Rysunek 4.5. Porównanie wydajności zadań CXR i EXR dla modelu trenowanego przy użyciu etykietowania ręcznego oraz etykietowania programowego. Źródło: Dunnmon i in.*[15]

Moi studenci często dziwią się — jeśli heurystyka działa tak dobrze w przypadku etykietowania danych, dlaczego w ogóle potrzebujemy modeli uczenia maszynowego? Jednym z powodów jest to, że funkcje etykietujące mogą nie obsługiwać wszystkich próbek danych, więc można trenować modele ML na danych zaetykietowanych programowo, a następnie używać wytrenowanych modeli do generowania prognoz dla próbek, które nie zostały przeanalizowane przez funkcje etykietujące.

Nadzór słaby to prosty, ale potężny paradygmat. Nie jest on jednak doskonały. W niektórych przypadkach etykiety uzyskane za pomocą nadzoru słabego mogą być nadmiernie zakłócone, a przez to nieużyteczne. Jednak nawet w takich przypadkach nadzór słaby można zastosować na samym początku pracy, gdy chcemy zbadać wydajność modeli uczenia maszynowego bez konieczności inwestowania zbyt wiele w etykietowanie ręczne.

## Półnadzór

Nadzór słaby wykorzystuje heurystykę w celu uzyskania zakłóconych etykiet, natomiast półnadzór stosuje założenia strukturalne do generowania nowych etykiet na podstawie niewielkiego zestawu etykiet początkowych. W przeciwieństwie do nadzoru słabego półnadzór wymaga początkowego zestawu etykiet.

Uczenie półnadzorowane jest techniką stosowaną już w latach 90. XX wieku[16]. Od tego czasu pojawiło się wiele nowych rozwiązań. Kompleksowy przegląd uczenia półnadzorowanego wykracza poza zakres tej książki. Przeanalizujemy więc mały podzbiór metod, aby Czytelnik mógł zrozumieć, w jaki sposób są one wykorzystywane. W celu uzyskania dodatkowych informacji warto się zapoznać

[15] Dummon i in., *Cross-Modal Data Programming*.

[16] Avrim Blum i Tom Mitchell, *Combining Labeled and Unlabeled Data with Co-Training*, „Proceedings of the Eleventh Annual Conference on Computational Learning Theory", lipiec 1998, s. 92 – 100, *https://oreil.ly/T79AE*.

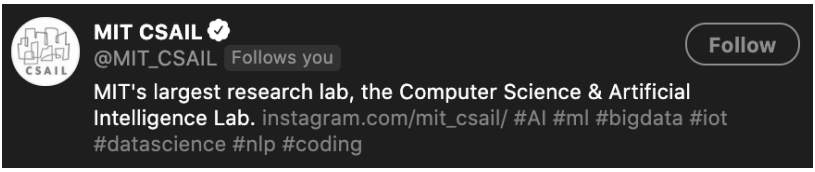


Kup książkę Poleć książkę

z dokumentami *Semi-Supervised Learning Literature Survey* (*https://oreil.ly/ULeWD*) (Xiaojin Zhu, 2008) i *A Survey on Semi-Supervised Learning* (*https://oreil.ly/JYgCH*) (Engelen i Hoos, 2018).

Klasyczną metodą wykorzystującą półnadzór jest **autotrenowanie**. Zaczynamy od trenowania modelu na istniejącym zbiorze zaetykietowanych danych. Następnie używamy tego modelu do tworzenia prognoz dla próbek niezaetykietowanych. Zakładając, że prognozy z wysokimi wynikami prawdopodobieństwa są poprawne, dodajemy do zbioru treningowego etykiety prognozowane z wysokim prawdopodobieństwem. Przy użyciu takiego rozszerzonego zbioru danych trenujemy nowy model. Proces trwa do momentu, aż będziemy zadowoleni z uzyskanej wydajności.

Inna metoda półnadzoru zakłada, że próbki danych, które mają podobne cechy, charakteryzują się takimi samymi etykietami. Podobieństwo może być oczywiste, jak na przykład w zadaniu klasyfikacji hasztagów na Twitterze. Zacznijmy od zaetykietowania hasztagu „#AI" jako *Informatyka*. Załóżmy, że hasztagi, które pojawiają się w tym samym tweecie lub profilu, prawdopodobnie dotyczą tego samego zagadnienia. W takim wypadku moglibyśmy również zaetykietować hasztagi „#ML" i „#BigData" jako *Informatyka* (rysunek 4.6).




*Rysunek 4.6. Ponieważ hasztagi #ML i #BigData pojawiają się na tym samym profilu co hasztag #AI, możemy założyć, że dotyczą tego samego typu zagadnień*

W większości przypadków podobieństwo może zostać odkryte tylko przez zastosowanie bardziej złożonych metod. Na przykład aby odkryć próbki, które należą do tej samej klasy, może być konieczne użycie metody grupowania lub algorytmu *k*-najbliższych sąsiadów.

Jedna z metod, która zyskała popularność w ostatnich latach, jest oparta na perturbacjach. Wykorzystuje ona założenie, że niewielkie perturbacje próbki nie powinny zmieniać jej etykiety. W instancjach treningowych stosuje się więc małe perturbacje, aby uzyskać nowe instancje. Perturbacje mogą być użyte bezpośrednio w próbkach (na przykład do obrazów jest dodawany biały szum) lub w ich reprezentacji (przykładowo do osadzenia słów dodawane są niewielkie, losowe wartości). Próbki z perturbacjami mają takie same etykiety jak oryginalne. Więcej informacji na ten temat pojawi się w punkcie „Perturbacja".

W niektórych przypadkach metody wykorzystujące półnadzór osiągnęły wydajność uczenia nadzorowanego, nawet jeśli z danego zbioru danych została odrzucona znaczna część etykiet[17].

Półnadzór jest najbardziej przydatny, gdy liczba etykiet treningowych jest ograniczona. Podczas wykonywania metody półnadzoru z ograniczoną ilością danych należy wziąć pod uwagę, ile z nich powinno zostać użytych do oceny wielu kandydujących modeli i wybrania spośród nich najlepszego.

[17] Avital Oliver, Augustus Odena, Colin Raffel, Ekin D. Cubuk i Ian J. Goodfellow, *Realistic Evaluation of Deep Semi-Supervised Learning Algorithms*, „NeurIPS 2018 Proceedings", *https://oreil.ly/dRmPV*.



Kup książkę Poleć książkę

Jeśli dysponujesz małą ilością danych, modelem najlepiej działającym ze zbiorem ewaluacyjnym jest ten, który jest do tego zbioru nadmiernie dopasowany. Jeśli jednak do ewaluacji używasz dużej ilości danych, wzrost wydajności uzyskany przez wybór najlepszego modelu może być mniejszy niż taki, który osiągnie się przez dodanie zbioru ewaluacyjnego do ograniczonego zbioru treningowego. Wiele firm przezwycięża ten kompromis poprzez użycie rozsądnie dużego zbioru ewaluacyjnego w celu wyboru najlepszego modelu, a następnie kontynuuje trenowanie zwycięskiego modelu za pomocą tego samego zbioru.

### Uczenie transferowe

Uczenie transferowe dotyczy rodziny metod, w których model opracowany dla pewnego zadania jest ponownie wykorzystywany jako punkt wyjścia dla innego. Model bazowy jest najpierw trenowany w celu rozwiązania zadania bazowego. Takie zadanie charakteryzuje się tanimi i licznymi danymi treningowymi. Świetnym przykładem jest modelowanie języka, ponieważ nie wymaga etykietowanych danych. Modele językowe mogą być trenowane przy użyciu dowolnych tekstów — książek, artykułów z Wikipedii, historii rozmów. Zadanie polega na tym, że biorąc pod uwagę istniejącą sekwencję tokenów[18], należy przewidzieć następny. W przypadku sekwencji „Kupiłem akcje firmy NVIDIA, ponieważ ważny jest", model językowy może jako kolejny token zaproponować „sprzęt" lub „procesor graficzny".

Wytrenowany model może następnie zostać użyty do rozwiązywania zadania niższego rzędu, takiego jak analiza emocji, wykrywanie intencji lub odpowiadanie na pytania. W niektórych przypadkach, jak na przykład w projektach wymagających uczenia od podstaw, można w zadaniu niższego rzędu użyć bezpośrednio modelu bazowego. Często jednak może być konieczne *dostrojenie* takiego modelu. Proces dostrajania oznacza wprowadzanie niewielkich zmian w modelu bazowym, takich jak kontynuowanie trenowania modelu bazowego lub jego fragmentu za pomocą danych pochodzących z zadania niższego rzędu[19].

Aby model bazowy wygenerował pożądane dane wyjściowe, czasami trzeba zmodyfikować dane wejściowe za pomocą szablonu[20]. Na przykład aby w przypadku zadania odpowiadania na pytania użyć modelu językowego jako bazowego, można wykorzystać takie pytania:

*P: Kiedy powstały Stany Zjednoczone?*

*O: 4 lipca 1776 roku.*

*P: Kto był głównym autorem Deklaracji Niepodległości?*

*O: Thomas Jefferson.*

*P: W którym roku urodził się Alexander Hamilton?*

*O:*

[18] Tokenem może być słowo, znak lub fragment słowa.

[19] Jeremy Howard i Sebastian Ruder, *Universal Language Model Fine-tuning for Text Classification*, arXiv, 18 stycznia 2018, *https://oreil.ly/DBEbw*.

[20] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi i Graham Neubig, *Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing*, arXiv, 28 lipca 2021, *https://oreil.ly/0lBgn*.



Kup książkę Poleć książkę

Po wprowadzeniu powyższych pytań do modelu językowego, takiego jak GPT-3 (*https://oreil.ly/qT0r3*), odpowiedzią na ostatnie może być rok urodzenia Alexandra Hamiltona.

Uczenie transferowe jest szczególnie atrakcyjne w przypadku zadań, które nie zawierają wielu danych oznaczonych etykietami. Nawet w przypadku takich, które mają dużo danych zaetykietowanych, użycie wstępnie wytrenowanego modelu jako punktu początkowego może często znacznie zwiększyć wydajność w porównaniu z trenowaniem od podstaw.

Uczenie transferowe cieszy się w ostatnich latach dużym zainteresowaniem z właściwych powodów. Umożliwiło ono powstanie wielu aplikacji, które wcześniej były niemożliwe do utworzenia ze względu na brak próbek treningowych. Spora część modeli uczenia maszynowego działających obecnie w produkcji to wynik zastosowania uczenia transferowego. Wśród nich są systemy wykrywania obiektów, które wykorzystują modele wstępnie wytrenowane za pomocą zbioru ImageNet, a także aplikacje klasyfikujące teksty, które używają wstępnie wytrenowanych modeli językowych, takich jak BERT lub GPT-3[21]. Uczenie transferowe ułatwia wdrażanie systemów uczenia maszynowego, ponieważ obniża koszty początkowe potrzebne do etykietowania danych, wymaganych w aplikacjach ML.

Od około pięciu lat panuje przekonanie, że (zazwyczaj) im większy wstępnie wytrenowany model bazowy, tym lepsza jest jego wydajność podczas rozwiązywania kolejnych zadań. Trenowanie dużych modeli jest kosztowne. Na podstawie konfiguracji modelu GPT-3 można oszacować, że koszt wytrenowania tego modelu to dziesiątki milionów dolarów. Wiele osób wysnuło hipotezę, że w przyszłości tylko garstka firm będzie mogła sobie pozwolić na trenowanie dużych, wstępnie wytrenowanych modeli. Reszta branży będzie z nich bezpośrednio korzystać lub dopasuje je do swoich specyficznych potrzeb.

### Uczenie aktywne

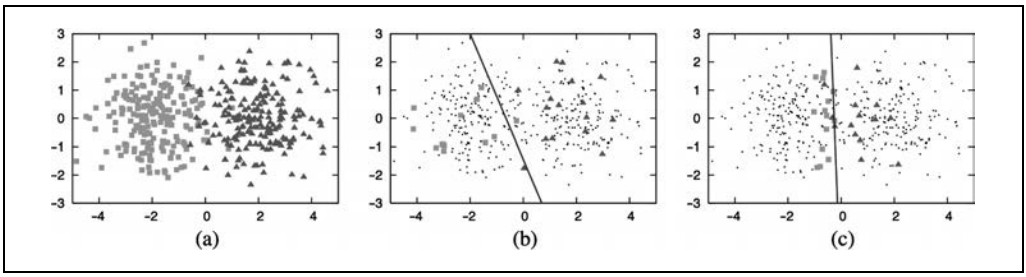
Uczenie aktywne jest metodą poprawiającą wydajność etykiet danych. Jeśli model uczenia maszynowego ma możliwość wyboru, na podstawie jakich próbek danych powinien się uczyć, może osiągnąć większą dokładność przy mniejszej liczbie etykiet treningowych. Uczenie aktywne jest czasem nazywane uczeniem na podstawie zapytań (jednak ten termin staje się coraz mniej popularny), ponieważ model (obiekt uczący się) odsyła zapytania w postaci niezaetykietowanych próbek, które mają zostać zaetykietowane przez komentatorów (zazwyczaj ludzi).

Zamiast etykietować losowe próbki danych, wybiera się te, które na podstawie pewnych wskaźników lub heurystyk mogą być najbardziej przydatne dla modeli. Najprostszym wskaźnikiem jest pomiar niepewności — etykietujemy przykłady, co do których model jest najmniej pewny. Dzięki takiej operacji model może lepiej poznać granice decyzyjne. Na przykład w przypadku rozwiązywania problemów z klasyfikacją model generuje prawdopodobieństwa dla różnych klas. Przy zastosowaniu pomiaru niepewności mógłby on wybrać próbki danych z najniższymi prawdopodobieństwami. Na rysunku 4.7 pokazano działanie metody na danych testowych.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee i Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, arXiv, 11 października 2018, *https://oreil.ly/RdIGU*; Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan i in., *Language Models Are Few-Shot Learners*, OpenAI, 2020, *https://oreil.ly/YVmrr*.



Kup książkę Poleć książkę




*Rysunek 4.7. Sposób działania uczenia aktywnego opartego na pomiarze niepewności. (a) Testowy zbiór danych zawierający 400 instancji, uzyskanych poprzez równomierne próbkowanie z dwóch klas gaussowskich. (b) Model wytrenowany na 30 losowo zaetykietowanych próbkach charakteryzuje się dokładnością 70%. (c) Model wytrenowany na 30 próbkach wybranych przez uczenie aktywne ma dokładność 90%. Źródło: Burr Settles*[22]

Inna powszechnie stosowana heurystyka opiera się na wykorzystaniu braku jedności wśród wielu modeli kandydujących. Jest ona przykładem metody zespołowej, a jej formalna nazwa brzmi **query-by-committee**[23]. Wymagany jest zespół złożony z kilku modeli kandydujących, które są zwykle tym samym modelem wytrenowanym z różnymi zestawami hiperparametrów lub przy użyciu różnych zbiorów danych. Każdy model ma możliwość oddania jednego głosu dotyczącego tego, jakie próbki powinny zostać zaetykietowane w następnej kolejności. Model podejmuje decyzję, biorąc pod uwagę poziom niepewności prognozy. Etykietowane są te próbki, którym zespół jest najbardziej przeciwny.

Istnieją też inne heurystyki, takie jak wybieranie próbek, które pozwalają uzyskać najwyższy poziom uaktualnienia gradientu lub zmniejszenia straty. Aby dowiedzieć się więcej o metodach uczenia aktywnego, zapoznaj się z dokumentem *Active Learning Literature Survey* (*https://oreil.ly/4RuBo*) (Settles, 2010).

Próbki przeznaczone do etykietowania mogą pochodzić z różnych źródeł. Mogą być sztucznie tworzone przez model w obszarze przestrzeni wejściowej, co do którego jest najbardziej niepewny[24]. Mogą pochodzić z rozkładu stacjonarnego, w którym znajduje się wiele niezaetykietowanych danych, a model musi wybrać spośród nich te, które powinny zostać zaetykietowane. Mogą pochodzić z rozkładu ze świata rzeczywistego (na przykład z produkcji), w którym istnieje strumień danych przychodzących, a model wybiera z nich próbki do zaetykietowania.

Najbardziej ekscytuje mnie działanie uczenia aktywnego w przypadku, gdy system pracuje z danymi w czasie rzeczywistym. Dane się ciągle zmieniają — jest to zjawisko, które w skrócie przenalizowaliśmy w rozdziale 1., a dokładniej przedstawimy w rozdziale 8. Uczenie aktywne pozwala modelowi uczyć się efektywniej w czasie rzeczywistym i szybciej dopasowywać do zmieniających się środowisk.

[22] Burr Settles, *Active Learning*, Morgan & Claypool, Williston 2012.

[23] Zespoły zostaną przeanalizowane w rozdziale 6.

[24] Dana Angluin, *Queries and Concept Learning*, „Machine Learning", 2, 1988, s. 319 – 342, *https://oreil.ly/0uKs4*.



Kup książkę Poleć książkę

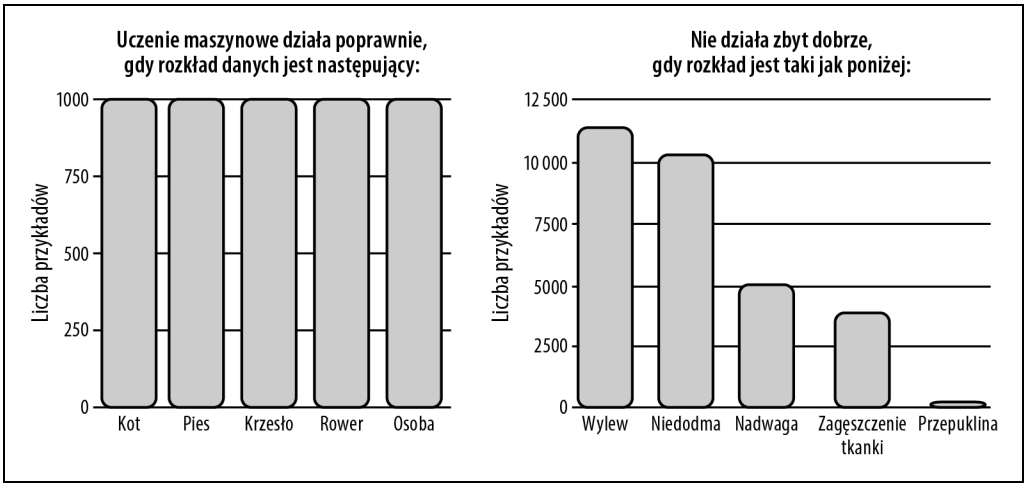
# Niezrównoważenie klas

Niezrównoważenie klas dotyczy zazwyczaj zadań klasyfikacyjnych, w których występuje znaczna różnica w liczbie próbek dla każdej z klas danych treningowych. Na przykład w zbiorze danych treningowych wykorzystywanych w modelu wykrywającym raka płuc na podstawie zdjęć rentgenowskich 99,99% z nich może przedstawiać zdrowe narządy, a tylko 0,01% takie, w których znajdują się komórki rakowe.

Niezrównoważenie klas może również wystąpić w zadaniach regresji, w których etykiety są ciągłe. Rozważmy zadanie szacowania kosztów opieki zdrowotnej[25]. Takie koszty są bardzo niesymetryczne — mediana jest niska, ale wartość z 95. percentyla jest już astronomiczna. Przy przewidywaniu kosztów opieki szpitalnej ważniejsze może być więc dokładne prognozowanie rachunków z 95. percentyla niż mediany. Różnica 100% w rachunku na 1000 zł jest do zaakceptowania (rzeczywista kwota 2000 zł, przewidywana 1000 zł), ale 100-procentowa różnica w rachunku na 40 000 zł już taka nie jest (rzeczywista kwota 80 000 zł, przewidywana 40 000 zł). Być może trzeba będzie więc wytrenować model, który powinien lepiej prognozować kwoty z 95. percentyla, nawet jeśli przyczyni się to do pogorszenia ogólnych wskaźników.

## Wyzwania związane z niezrównoważeniem klas

Uczenie maszynowe, a zwłaszcza uczenie głębokie, działa dobrze w sytuacjach, gdy rozkład danych jest bardziej zrównoważony. Ma ono natomiast problemy, gdy klasy są znacząco niezrównoważone, jak pokazano na rysunku 4.8.




*Rysunek 4.8. Uczenie maszynowe sprawdza się w sytuacjach, w których klasy są zrównoważone. Źródło: utworzono na podstawie rysunku autorstwa Andrew Ng*[26]

[25] Podziękowania dla Eugene Yana za ten wspaniały przykład!

[26] Andrew Ng, *Bridging AI's Proof-of-Concept to Production Gap*, HAI Seminar, 22 września 2020, wideo, 1:02:07, *https://oreil.ly/FSFWS*.



Kup książkę Poleć książkę

Niezrównoważenie klas może utrudniać uczenie z następujących trzech powodów. Po pierwsze, często oznacza, że model nie potrafi się dostatecznie dobrze nauczyć wykrywać klasy mniejszościowe. W przypadku gdy istnieje niewielka liczba instancji w klasie mniejszościowej, mamy w rzeczywistości do czynienia z uczeniem few-shot (na podstawie niewielu danych). Model tylko kilka razy może mieć dostęp do klasy mniejszościowej, zanim będzie musiał podjąć decyzję na jej temat. W przypadku gdy zbiór treningowy nie zawiera rzadkich klas, model może założyć, że one po prostu nie istnieją.

Drugim powodem jest to, że model wykorzystując prostą heurystykę, zamiast zdobywać wiedzę o podstawowym wzorcu danych, może opracować nieoptymalne rozwiązanie. Rozważmy poprzedni przykład wykrywania raka płuc. Jeśli model nauczy się zawsze zwracać klasę większościową, jego dokładność wyniesie 99,99%[27]. Taka heurystyka może być nieodpowiednia dla algorytmów spadku wzdłuż gradientu, ponieważ już niewielki poziom przypadkowości może spowodować pogorszenie dokładności.

Wreszcie można stwierdzić, że niezrównoważenie klas prowadzi do asymetrycznych kosztów popełniania błędów. Koszt błędnej prognozy przeprowadzonej na podstawie próbki z klasy rzadkiej może być znacznie wyższy niż w przypadku, gdy wykorzystuje się próbkę z klasy większościowej.

Na przykład błędna klasyfikacja zdjęcia rentgenowskiego z komórkami rakowymi jest znacznie bardziej niebezpieczna niż błędna klasyfikacja zdjęcia przedstawiającego zdrowe płuca. Jeśli funkcja straty nie została skonfigurowana w taki sposób, aby uwzględnić tę asymetrię, model będzie tak samo traktował wszystkie próbki. W rezultacie możesz otrzymać model, który radzi sobie równie dobrze zarówno z klasą większościową, jak i mniejszościową, podczas gdy wolałbyś mieć taki, który gorzej obsługuje klasę większościową, ale znacznie lepiej mniejszościową.

Gdy studiowałam, większość zbiorów danych, z jakimi miałam do czynienia, charakteryzowała się mniej lub bardziej zrównoważonymi klasami[28]. Gdy zaczęłam pracować, szokiem było dla mnie to, że niezrównoważenie klas jest normą. W świecie rzeczywistym rzadkie zdarzenia są często bardziej interesujące (lub bardziej niebezpieczne) niż te, które występują regularnie. Wiele projektów dotyczy wykrywania właśnie tych rzadkich zdarzeń.

Klasycznym przykładem problemu z niezrównoważeniem klas jest wykrywanie oszustw. Większość transakcji kartami kredytowymi jest poprawna. Według stanu na 2018 rok 6,8 centów na każde 100 dolarów wydatków posiadacza karty było oszustwem[29]. Innym przykładem takiego zadania jest prognozowanie odchodzenia klientów. Większość klientów prawdopodobnie nie planuje rezygnacji z subskrypcji. Nawet jeśli tacy się trafiają, Twoja firma ma więcej powodów do zmartwień niż algorytmy prognozujące odchodzenie klientów. Inne przykłady obejmują klasyfikację chorób

[27] Właśnie dlatego dokładność jest niewłaściwym wskaźnikiem w przypadku niezrównoważenia klas. Więcej o tym w punkcie „Rozwiązywanie problemu niezrównoważenia klas".

[28] Założyłam, że łatwiej nauczę się teorii związanej z uczeniem maszynowym, jeśli nie będę musiała wymyślać, jak poradzić sobie z niezrównoważeniem klas.

[29] The Nilson Report, *Payment Card Fraud Losses Reach $27.85 Billion*, PR Newswire, 21 listopada 2019, *https://oreil.ly/NM5zo*.



Kup książkę Poleć książkę

(w większości przypadków, na szczęście, ludzie nie mają chorób terminalnych) i klasyfikację CV (98% osób poszukujących pracy jest eliminowanych podczas wstępnego etapu filtrowania CV[30]).

Mniej oczywistym przykładem zadania z niezrównoważonymi klasami jest wykrywanie obiektów (*https://oreil.ly/CGEf5*). Algorytmy wykrywania obiektów generują na obrazie dużą liczbę ograniczonych obszarów, a następnie przewidują, w których z nich najprawdopodobniej znajdują się obiekty. Większość obszarów nie zawiera poszukiwanych obiektów.

Poza przypadkami, w których niezrównoważenie klas jest nieodłącznym elementem problemu, może ono również wynikać z błędów powstałych podczas procesu próbkowania. Załóżmy, że chcemy utworzyć dane treningowe służące do wykrywania, czy e-mail jest spamem. W tym celu postanawiasz wykorzystać wszystkie zanonimizowane wiadomości e-mailowe z firmowej bazy danych. Według firmy Talos Intelligence (stan na maj 2021 roku) prawie 85% wszystkich wiadomości e-mailowych to spam[31]. Jednak zanim poczta dotarła do bazy danych, większość niewłaściwych informacji została odfiltrowanych. Zbiór danych zawiera więc niewiele spamu.

Inną, choć mniej powszechną przyczyną braku zrównoważenia klas są błędy w etykietowaniu. Osoby wykonujące adnotacje mogły źle przeczytać instrukcję, niewłaściwie się do niej zastosować (myśląc, że są tylko dwie klasy, OCENA_POZYTYWNA i OCENA_NEGATYWNA, podczas gdy w rzeczywistości są trzy) lub po prostu popełniły błędy. W każdym przypadku, gdy mamy do czynienia z problemem niezrównoważenia klas, należy zbadać dane, aby zrozumieć jego przyczyny.

## Rozwiązywanie problemu niezrównoważenia klas

Problem niezrównoważenia klas był dokładnie badany przez ostatnie dwie dekady ze względu na jego powszechne występowanie w aplikacjach używanych w świecie rzeczywistym[32]. W zależności od poziomu niezrównoważenia wpływa ono w różny sposób na zadania. Niektóre problemy są bardziej wrażliwe na brak zrównoważenia klas niż inne. Nathalie Japkowicz wykazała, że wrażliwość na brak zrównoważenia wzrasta wraz ze złożonością problemu, a na liniowo odseparowane proste problemy nie mają wpływu żadne poziomy braku zrównoważenia klas[33]. Niezrównoważenie klas w problemach klasyfikacji binarnej jest znacznie łatwiejsza do obsłużenia niż nierównowaga klas w problemach klasyfikacji wieloklasowej. Wan Ding i in. wykazali, że bardzo głębokie sieci neuronowe (przy czym „bardzo głębokie" oznaczało w 2017 roku ponad 10 warstw) radzą sobie znacznie lepiej z niezrównoważonymi danymi niż sieci płytsze[34].

Zaproponowano wiele technik mających na celu złagodzenie efektu niezrównoważenia klas. Ponieważ jednak w międzyczasie sieci neuronowe stały się znacznie większe i głębsze, a także posiadły lepsze umiejętności uczenia się, niektórzy mogliby stwierdzić, że obecnie nie powinno się próbować

[30] *Job Market Expert Explains Why Only 2% of Job Seekers Get Interviewed*, WebWire, 7 stycznia 2014, *https://oreil.ly/UpL8S*.

[31] *Email and Spam Data*, Talos Intelligence, ostatni dostęp: lipiec 2022, *https://oreil.ly/lI5Jr*.

[32] Nathalie Japkowicz i Shaju Stephen, *The Class Imbalance Problem: A Systematic Study*, 2002, *https://oreil.ly/d7lVu*.

[33] Nathalie Japkowicz, *The Class Imbalance Problem: Significance and Strategies*, 2000, *https://oreil.ly/Ma50Z*.

[34] Wan Ding, Dong-Yan Huang, Zhuo Chen, Xinguo Yu i Weisi Lin, *Facial Action Recognition Using Very Deep Networks for Highly Imbalanced Class Distribution*, „2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference" (APSIPA ASC), 2017, *https://oreil.ly/WeW6J*.



Kup książkę Poleć książkę

„naprawiać" niezrównoważenia klas, jeśli takie są właśnie dane w świecie realnym. Dobry model powinien się nauczyć modelować tę nierównowagę. Jego opracowanie może być jednak wyzwaniem, więc wciąż musimy polegać na specjalnych technikach treningowych.

W tym punkcie przeanalizujemy trzy sposoby pozwalające radzić sobie z niezrównoważeniem klas — wybór wskaźnika odpowiedniego dla danego problemu; metody na poziomie danych, czyli zmiana rozkładu danych, aby uczynić go mniej niezrównoważonym; a także metody na poziomie algorytmu, czyli zmiana metody uczenia, aby uczynić ją bardziej odporną na niezrównoważenie klas.

Wdrożenie tych technik może być konieczne, ale niewystarczające. Aby dowiedzieć się więcej, zapoznaj się z dokumentem *Survey on Deep Learning with Class Imbalance* (*https://oreil.ly/9QvBr*) (Johnson i Khoshgoftaar, 2019).

### Użycie właściwego wskaźnika ewaluacyjnego

Najważniejszą rzeczą, którą należy wykonać w przypadku problemu z niezrównoważonymi klasami, jest użycie odpowiednich wskaźników ewaluacyjnych. Niewłaściwe wskaźniki dadzą złe wyobrażenie o działaniu modeli, a także nie będą pomocne podczas ich rozwijania lub wyboru takich, które powinny być odpowiednie dla Twojego zadania.

Najczęściej wykorzystywanymi wskaźnikami służącymi do raportowania wydajności modeli uczenia maszynowego są ogólna dokładność i współczynnik błędów. Jednak nie są one wystarczające w przypadku zadań z niezrównoważonymi klasami, ponieważ traktują wszystkie klasy jednakowo, co oznacza, że zostaną one zdominowane przez wydajność modelu w klasie większościowej. Jest to szczególnie niebezpieczne, gdy klasa większościowa nie jest tym, na czym nam zależy.

Rozważmy zadanie z dwoma etykietami: NOWOTWÓR (klasa pozytywna) i BRAK_PROBLEMU (klasa negatywna), w którym 90% danych ma etykiety BRAK_PROBLEMU. Weźmy pod uwagę dwa modele, A i B, z macierzami błędów przedstawionymi w tabelach 4.4 i 4.5.

*Tabela 4.4. Macierz błędów modelu A. Model A może wykryć 10 na 100 przypadków oznaczonych etykietą NOWOTWÓR*

| Model A | Faktyczny wynik NOWOTWÓR | Faktyczny wynik BRAK_PROBLEMU |
|---|---|---|
| Przewidziano wynik NOWOTWÓR | 10 | 10 |
| Przewidziano wynik BRAK_PROBLEMU | 90 | 890 |

*Tabela 4.5. Macierz błędów modelu B. Model B może wykryć 90 na 100 przypadków oznaczonych etykietą NOWOTWÓR*

| Model B | Faktyczny wynik NOWOTWÓR | Faktyczny wynik BRAK_PROBLEMU |
|---|---|---|
| Przewidziano wynik NOWOTWÓR | 90 | 90 |
| Przewidziano wynik BRAK_PROBLEMU | 10 | 810 |

Prawdopodobnie większość osób stwierdzi, aby prognozy generował model B, ponieważ ma on większą szansę przewidzieć, czy pacjent ma raka. Oba modele mają jednak taką samą dokładność równą 0,9.



Kup książkę Poleć książkę

Lepszym wyborem byłyby wskaźniki, które pomagają zrozumieć wydajność modelu w odniesieniu do konkretnych klas. Dokładność może nadal być dobrym wskaźnikiem, jeśli używasz go oddzielnie dla każdej z klas. Dokładność modelu A dla klasy NOWOTWÓR wynosi 10%, a modelu B — 90%.

F1, precyzja i czułość są wskaźnikami, które w problemach klasyfikacji binarnej określają wydajność modelu w odniesieniu do klasy pozytywnej, ponieważ polegają na wyniku prawdziwie pozytywnym, czyli takim, w którym model poprawnie przewiduje klasę pozytywną[35].

## Precyzja, czułość i F1

W przypadku zadań klasyfikacji binarnej wskaźniki precyzja, czułość i F1 są obliczane przy użyciu liczby wyników prawdziwie pozytywnych, prawdziwie negatywnych, fałszywie pozytywnych i fałszywie negatywnych. Definicje tych pojęć przedstawiono w tabeli 4.6.

*Tabela 4.6. Definicje wyników prawdziwie pozytywnych, prawdziwie negatywnych, fałszywie pozytywnych i fałszywie negatywnych w przypadku zadań klasyfikacji binarnej*

| | Przewidywany wynik pozytywny | Przewidywany wynik negatywny |
|---|---|---|
| Etykieta pozytywna | Wynik prawdziwie pozytywny (trafienie) | Wynik fałszywie negatywny (błąd II rodzaju, pomyłka) |
| Etykieta negatywna | Wynik fałszywie pozytywny (błąd I rodzaju, fałszywy alarm) | Wynik prawdziwie negatywny (poprawne odrzucenie) |

precyzja = wyniki prawdziwie pozytywne : (wyniki prawdziwie pozytywne + wyniki fałszywie pozytywne)

czułość = wyniki prawdziwie pozytywne : (wyniki prawdziwie pozytywne + wyniki fałszywie negatywne)

F1 = 2 · precyzja · czułość : (precyzja + czułość)

F1, precyzja i czułość są wskaźnikami asymetrycznymi, co oznacza, że ich wartości zmieniają się w zależności od tego, która klasa zostanie uznana za pozytywną. Jeśli założymy, że klasa NOWOTWÓR jest pozytywna, wskaźnik F1 dla modelu A wyniesie 0,17. Jeśli jednak za klasę pozytywną uznamy BRAK_PROBLEMU, wskaźnik F1 będzie równy 0,95. W tabeli 4.7 przedstawiono wartości wskaźników dla modeli A i B przy założeniu, że klasa NOWOTWÓR jest pozytywna.

*Tabela 4.7. Oba modele charakteryzują się taką samą dokładnością, mimo że jeden z nich jest wyraźnie lepszy*

| | NOWOTWÓR (1) | BRAK_PROBLEMU (2) | Dokładność | Precyzja | Czułość | F1 |
|---|---|---|---|---|---|---|
| Model A | 10/100 | 890/900 | 0,9 | 0,5 | 0,1 | 0,17 |
| Model B | 90/100 | 810/900 | 0,9 | 0,5 | 0,9 | 0,64 |

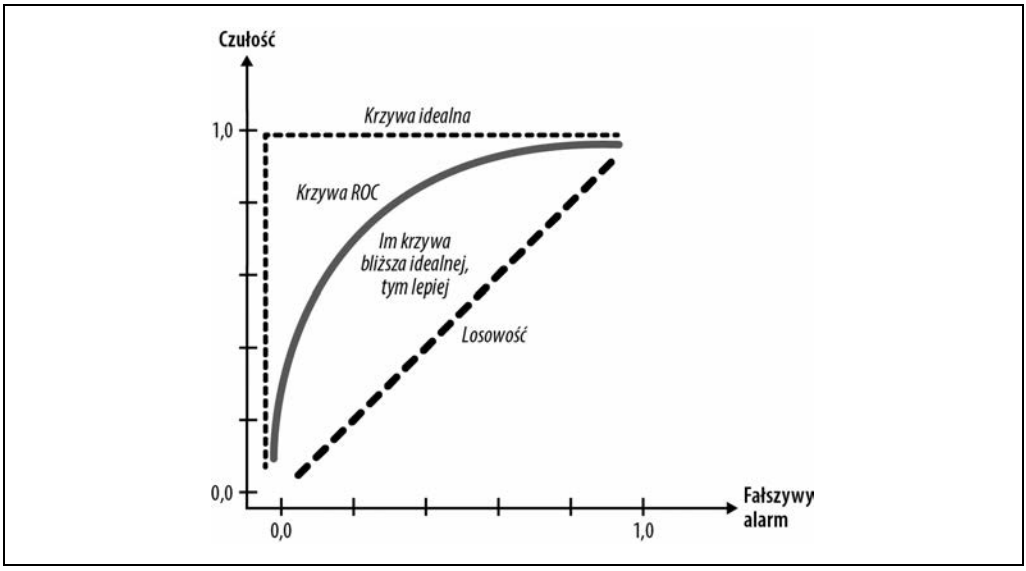
[35] W funkcji `scikit-learn.metrics.f1_score` wartość `pos_label` jest domyślnie ustawiona na 1, ale jeśli chciałbyś, by wartość 0 była Twoją pozytywną etykietą, możesz zmienić `pos_label` na 0 (stan na lipiec 2021).



Kup książkę Poleć książkę

Wiele zagadnień związanych z klasyfikacją może być modelowanych za pomocą problemów regresyjnych. Model generuje prawdopodobieństwo, a na jego podstawie próbka zostaje sklasyfikowana. Na przykład jeśli wartość jest większa niż 0,5, mamy etykietę pozytywną, a jeśli jest mniejsza lub równa 0,5, negatywną. Oznacza to, że można tak dobrać próg, aby zwiększyć **współczynnik wyników prawdziwie pozytywnych** (czyli wskaźnik **czułości**), jednocześnie zmniejszając **współczynnik wyników fałszywie pozytywnych** (znany również jako **prawdopodobieństwo fałszywego alarmu**) — i odwrotnie. Możemy więc dla różnych progów wykreślić współczynnik wyników prawdziwie pozytywnych względem współczynnika wyników fałszywie pozytywnych. Wykres ten znany jest jako **krzywa ROC** (ang. *receiver operating characteristics* — charakterystyka operacyjna odbiornika). Gdy model jest doskonały, czułość wynosi 1,0, a krzywa jest prostą linią na górze wykresu. Pokazuje ona, jak zmienia się wydajność modelu w zależności od progu, i pomaga w wybraniu takiego, który jest najlepszy. Im kształt krzywej jest bardziej podobny do linii idealnej, tym lepsza wydajność modelu.

Zmienia się także wielkość pola pod krzywą (AUC — ang. *area under the curve*) ROC. Ponieważ im krzywa bardziej podobna do linii idealnej, tym lepiej, więc tym samym im pole pod krzywą większe, tym lepiej (rysunek 4.9).




*Rysunek 4.9. Krzywa ROC*

Podobnie jak wskaźniki F1 i czułość, również krzywa ROC dotyczy tylko klasy pozytywnej i nie pokazuje, jak dobrze model radzi sobie z klasą negatywną. Davis i Goadrich zasugerowali, że należy wykreślić wykres precyzji w funkcji czułości, i nazwali go krzywą precyzja-czułość. Według nich taka krzywa daje bardziej informacyjny obraz wydajności algorytmu w zadaniach z dużym niezrównoważeniem klas[36].

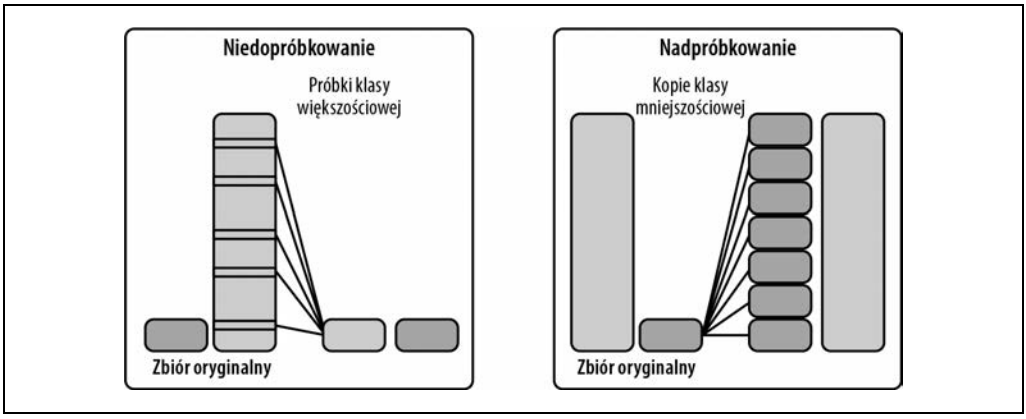
[36] Jesse Davis i Mark Goadrich, *The Relationship Between Precision-Recall and ROC Curves*, *Proceedings of the 23rd International Conference on Machine Learning*, 2006, *https://oreil.ly/s40F3*.



Kup książkę Poleć książkę

## Metody na poziomie danych — ponowne próbkowanie

Metody na poziomie danych modyfikują rozkład danych treningowych w celu zmniejszenia poziomu nierównowagi, a przez to ułatwiają proces uczenia modelu. Często używaną rodziną technik jest ponowne próbkowanie. Obejmuje ono **nadpróbkowanie** (ang. *oversampling*), czyli dodawanie większej liczby przypadków z klas mniejszościowych, oraz **niedopróbkowanie** (ang. *undersampling*), czyli usuwanie przypadków z klas większościowych. Najprostszym sposobem niedopróbkowania jest losowe usunięcie instancji z klasy większościowej, natomiast nadpróbkowania jest losowe tworzenie kopii klasy mniejszościowej, aż uzyskamy współczynnik, z którego jesteśmy zadowoleni. Na rysunku 4.10 pokazano schemat operacji nadpróbkowania i niedopróbkowania.




*Rysunek 4.10. Ilustracja działania metod niedopróbkowania i nadpróbkowania. Źródło: utworzono na podstawie rysunku autorstwa Rafaela Alencara*[37]

Popularną metodą niedopróbkowania danych o niewielkiej liczbie wymiarów, która została opracowana w 1976 roku, są tak zwane **połączenia Tomka** (ang. *Tomek links*).[38] Używając tej techniki, pobieramy pary próbek z przeciwnych klas, które znajdują się blisko siebie, a następnie z każdej z nich usuwamy próbkę klasy większościowej.

Taka metoda sprawia, że granica decyzyjna staje się bardziej przejrzysta, a modele mogą ją łatwiej poznać. Jednak model staje się jednocześnie mniej odporny, ponieważ nie poznaje subtelności realnej granicy decyzyjnej.

Często wykorzystywaną metodą nadpróbkowania danych o niewielkiej liczbie wymiarów jest SMOTE (ang. *synthetic minority oversampling technique — generowanie próbek syntetycznych z klasy mniejszościowej*).[39] Syntetyzuje ona nowe próbki poprzez próbkowanie wypukłych kombinacji istniejących punktów danych w obrębie klasy mniejszościowej[40].

[37] Rafael Alencar, *Resampling Strategies for Imbalanced Datasets*, Kaggle, *https://oreil.ly/p8Whs*.

[38] Ivan Tomek, *An Experiment with the Edited Nearest-Neighbor Rule*, „IEEE Transactions on Systems, Man, and Cybernetics", czerwiec 1976, s. 448 – 452, *https://oreil.ly/JCxHZ*.

[39] N.V. Chawla, K.W. Bowyer, L.O. Hall i W.P. Kegelmeyer, *SMOTE: Synthetic Minority Over-sampling Technique*, „Journal of Artificial Intelligence Research", 16, 2002, s. 341 – 378, *https://oreil.ly/f6y46*.

[40] „Wypukły" oznacza w przybliżeniu w tym kontekście „liniowy".



Kup książkę Poleć książkę

Zarówno metoda SMOTE, jak i połączenia Tomka okazały się skuteczne jedynie w przypadku danych o niewielkiej liczbie wymiarów. Wiele zaawansowanych technik ponownego próbkowania, takich jak Near-Miss i selekcja jednostronna[41], wymaga obliczenia odległości między instancjami lub między instancjami a granicami decyzyjnymi, co może być kosztowne lub niewykonalne w przypadku danych wielowymiarowych lub wielowymiarowej przestrzeni cech, co charakteryzuje duże sieci neuronowe.

Po przeprowadzeniu procesu ponownego próbkowania nigdy nie należy ewaluować modelu za pomocą uzyskanych danych, ponieważ zostanie on do nich nadmierne dopasowany.

Niedopróbkowanie grozi utratą ważnych danych z powodu ich usunięcia. Nadpróbkowanie wiąże się z ryzykiem nadmiernego dopasowania do danych treningowych, zwłaszcza jeśli dodane kopie klasy mniejszościowej są replikami istniejących danych. W celu uniknięcia tych zagrożeń opracowano wiele wyrafinowanych technik próbkowania.

Jedną z nich jest **uczenie dwufazowe** (ang. *two-phase learning*)[42]. Najpierw trenuje się model za pomocą ponownie spróbkowanych danych. Dane wynikowe można uzyskać przez losowe niedopróbkowanie dużych klas, by każda z nich miała tylko *N* instancji. Następnie model jest trenowany przy użyciu danych oryginalnych.

Inną wykorzystywaną techniką jest próbkowanie dynamiczne, czyli nadpróbkowanie klas o niskiej wydajności i niedopróbkowanie klas o wysokiej wydajności podczas procesu trenowania. Metoda ta, wprowadzona przez Pouyanfara i in.[43], ma na celu udostępnienie modelowi mniej wiedzy, którą już zdobył, a więcej tego, czego jeszcze nie wie.

### Metody na poziomie algorytmu

Metody na poziomie danych ograniczają problemy związane z niezrównoważeniem klas poprzez zmianę rozkładu danych treningowych, natomiast metody na poziomie algorytmu utrzymują rozkład danych treningowych w stanie nienaruszonym, ale zmieniają sam algorytm, aby uczynić go bardziej odpornym.

Ponieważ procesem uczenia steruje funkcja straty (lub funkcja kosztu), na jej dostosowywaniu jest opartych wiele metod działających na poziomie algorytmu. Kluczowa idea polega na tym, że jeśli istnieją dwie instancje, $x_1$ i $x_2$, a strata wynikająca z błędnej prognozy jest dla $x_1$ większa niż dla $x_2$, model będzie przedkładał generowanie poprawnych prognoz dla $x_1$ nad poprawne przewidywanie dla $x_2$.

---

[41] Jianping Zhang i Inderjeet Mani, *kNN Approach to Unbalanced Data Distributions: A Case Study involving Information Extraction*, Workshop on Learning from Imbalanced Datasets II, ICML, Washington 2003, *https://oreil.ly/qnpra*; Miroslav Kubat i Stan Matwin, *Addressing the Curse of Imbalanced Training Sets: One-Sided Selection*, 2000, *https://oreil.ly/8pheJ*.

[42] Hansang Lee, Minseok Park i Junmo Kim, *Plankton Classification on Imbalanced Large Scale Database via Convolutional Neural Networks with Transfer Learning*, „2016 IEEE International Conference on Image Processing (ICIP)", 2016, *https://oreil.ly/YiA8p*.

[43] Samira Pouyanfar, Yudong Tao, Anup Mohan, Haiman Tian, Ahmed S. Kaseb, Kent Gauen, Ryan Dailey i in., *Dynamic Sampling in Convolutional Neural Networks for Imbalanced Data Classification*, „2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)", 2018, *https://oreil.ly/D3Ak5*.



Kup książkę Poleć książkę

Poprzez przypisanie większej wagi instancjom treningowym, na których nam zależy, możemy sprawić, że model skoncentruje się bardziej na ich uczeniu się.

Niech $L(x;\theta)$ będzie stratą spowodowaną przez instancję $x$ dla modelu z zestawem parametrów $\theta$. Strata modelu jest często definiowana jako średnia strata spowodowana przez wszystkie instancje. Niech $N$ oznacza całkowitą liczbę próbek treningowych.

$$L(X;\theta) = \Sigma_x \frac{1}{N} L(x;\theta)$$

Ta funkcja straty jednakowo wartościuje stratę spowodowaną przez każdą z instancji, nawet jeśli niektóre błędne prognozy dla pewnych instancji mogą być znacznie kosztowniejsze od innych. Istnieje wiele sposobów modyfikacji funkcji kosztu. W tym podpunkcie przeanalizujemy trzy z nich, a zaczniemy od **uczenia zależnego od kosztów** (ang. *cost-sensitive learning*).

**Uczenie zależne od kosztów** Charles Elkan zauważył, że błędne klasyfikowanie różnych klas jest związane z różnymi kosztami. W związku z tym w 2001 roku zaproponował metodę uczenia zależnego od kosztów, w którym indywidualna funkcja straty jest modyfikowana tak, aby uwzględnić ten zmieniający się koszt[44]. Metoda wykorzystuje macierz kosztów do określenia parametru $C_{ij}$, czyli kosztu w przypadku, gdy klasa $i$ została sklasyfikowana jako klasa $j$. Jeśli $i = j$, otrzymujemy poprawną klasyfikację, a jej koszt wynosi zwykle 0. W przeciwnym razie mamy do czynienia z klasyfikacją błędną. Jeśli klasyfikowanie przykładów WYNIK_POZYTYWNY jako WYNIK_NEGATYWNY jest dwa razy bardziej kosztowne niż odwrotnie, możesz uczynić parametr $C_{10}$ dwukrotnie większym od $C_{01}$.

Jeśli więc masz dwie klasy, WYNIK_POZYTYWNY i WYNIK_NEGATYWNY, macierz kosztów mogłaby wyglądać tak, jak przedstawiono w tabeli 4.8.

*Tabela 4.8. Przykład macierzy kosztów*

| | Faktyczny WYNIK_NEGATYWNY | Faktyczny WYNIK_POZYTYWNY |
|---|---|---|
| Prognozowano WYNIK_NEGATYWNY | $C(0,0) = C_{00}$ | $C(1,0) = C_{10}$ |
| Prognozowano WYNIK_POZYTYWNY | $C(0,1) = C_{01}$ | $C(1,1) = C_{11}$ |

Strata spowodowana przez instancję $x$ klasy $i$ stanie się średnią ważoną wszystkich możliwych klasyfikacji instancji $x$.

$$L(x;\theta) = \Sigma_j C_{ij} P(j \mid x;\theta)$$

Problem z tą funkcją straty polega na tym, że należy samodzielnie określić macierz kosztów, która się różni w zależności od zadania i jego skali.

[44] Charles Elkan, *The Foundations of Cost-Sensitive Learning*, „Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)", 2001, *https://oreil.ly/WGq5M*.



Kup książkę Poleć książkę

**Strata klasy zrównoważonej** W przypadku modelu wytrenowanego na niezrównoważonym zbiorze danych może się zdarzyć, że będzie się on skłaniał ku klasom większościowym i dokonywał błędnych prognoz dotyczących klas mniejszościowych. Co by się stało, gdybyśmy ukarali model za dokonywanie takich błędnych przewidywań, a przez to skorygowali jego sposób działania?

Najprostsze rozwiązanie polega na tym, by wagę każdej klasy uczynić odwrotnie proporcjonalną do liczby próbek, które się w niej zawierają. Przy takim założeniu rzadsze klasy będą miały większe wagi. W poniższym równaniu $N$ oznacza całkowitą liczbę próbek treningowych:

$$W_i = \frac{N}{\text{liczba próbek w klasie } i}$$

Strata spowodowana przez instancję $x$ klasy $i$ jest wyznaczana za pomocą poniższego wzoru. Funkcja Strata($x$, $j$) wyznacza stratę w przypadku, gdy $x$ zostanie sklasyfikowana jako klasa $j$. Może to być entropia krzyżowa lub dowolna inna funkcja straty.

$$L(x, \theta) = W_i \Sigma_j P(j \mid x; \theta) \text{Strata}(x, j)$$

Bardziej wyrafinowana wersja funkcji straty może uwzględniać nakładanie się istniejących próbek. Przykładem takiej funkcji jest strata klasy zrównoważonej oparta na efektywnej liczbie próbek[45].

**Strata ogniskowa** Niektóre przykłady danych są łatwiejsze do rozpoznania niż inne, a model szybko się uczy je klasyfikować. Chcemy więc zmotywować model do skoncentrowania się na uczeniu takich próbek, z którymi wciąż ma problem. Może dopasowalibyśmy stratę w taki sposób, że w przypadku, gdyby poprawne sklasyfikowanie próbki było mniej prawdopodobne, przypisalibyśmy jej większą wagę? W taki właśnie sposób działa **strata ogniskowa** (ang. *focal loss*)[46]. Na rysunku 4.11 przedstawiono równanie straty ogniskowej i jej wydajność w porównaniu ze stratą entropii krzyżowej.

W praktyce okazało się, że rozwiązanie problemu niezrównoważenia klas można osiągnąć za pomocą zespołów[47]. W tym rozdziale nie będziemy jednak analizować tej techniki, ponieważ niezrównoważenie klas nie jest zazwyczaj związane ze stosowaniem zespołów. Metody zespołowe zostaną omówione w rozdziale 6.

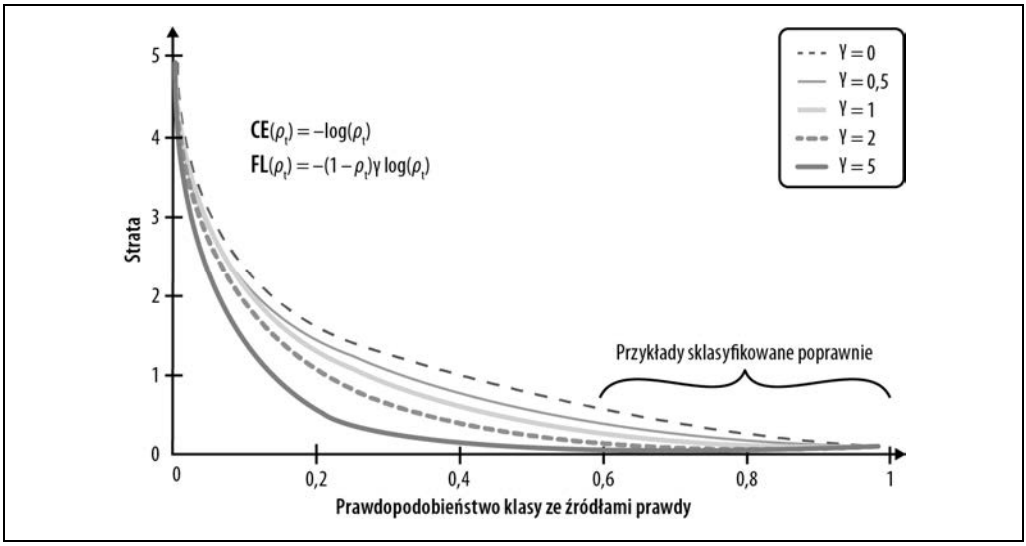
[45] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song i Serge Belongie, *Class-Balanced Loss Based on Effective Number of Samples*, „Proceedings of the Conference on Computer Vision and Pattern", 2019, *https://oreil.ly/jCzGH*.

[46] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He i Piotr Dollár, *Focal Loss for Dense Object Detection*, arXiv, 7 sierpnia 2017, *https://oreil.ly/Km2dF*.

[47] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince i Francisco Herrera, *A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches*, „IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)", 42, nr 4, lipiec 2012, s. 463 – 484, *https://oreil.ly/1ND4g*.



Kup książkę Poleć książkę




*Rysunek 4.11. Model wytrenowany przy użyciu straty ogniskowej (FL) wykazuje mniejsze wartości straty w porównaniu z modelem wytrenowanym z wykorzystaniem straty entropii krzyżowej (CE). Źródło: utworzono na podstawie rysunku autorstwa Lin i in.*

# Generowanie sztucznych danych

Generowanie sztucznych danych to zestaw metod, które są wykorzystywane do zwiększenia ilości danych treningowych. Tradycyjnie techniki te są stosowane w zadaniach, w których ilość danych treningowych jest ograniczona, na przykład w obrazowaniu medycznym. Jednak w ciągu ostatnich kilku lat okazało się, że są one przydatne nawet wtedy, gdy mamy dużo danych. Dodatkowe dane mogą sprawić, że modele staną się bardziej odporne na zakłócenia, a nawet na podstępny atak.

Generowanie sztucznych danych stało się standardową procedurą w przypadku wielu problemów dotyczących widzenia komputerowego, a także znajduje zastosowanie w przetwarzaniu języka naturalnego (NLP). Techniki generowania danych zależą w dużej mierze od formatu danych, ponieważ modyfikowanie obrazu różni się od modyfikowania tekstu. W tym rozdziale omówimy trzy główne metody generowania sztucznych danych: proste transformacje zachowujące etykiety, perturbacje, które są terminem oznaczającym „dodawanie zakłóceń", a także synteza danych. Dla każdej z metod przeanalizujemy przykłady dotyczące widzenia komputerowego, jak i przetwarzania języka naturalnego.

## Proste transformacje zachowujące etykiety

Najprostszą techniką zwiększania liczby danych w przypadku widzenia komputerowego jest losowa modyfikacja obrazu z zachowaniem jego etykiety. Można stosować takie operacje jak przycięcie, obrócenie, odwrócenie (poziomo lub pionowo), wyczyszczenie części obrazu i inne. Ma to sens, ponieważ obrócony obraz psa nadal przedstawia psa. Popularne biblioteki uczenia maszynowego takie jak PyTorch, TensorFlow i Keras wspierają generowanie dodatkowych obrazów.



Kup książkę Poleć książkę

Krizhevsky i in. opublikowali legendarny już artykuł o sieci AlexNet. Piszą oni, że „przekształcone obrazy są generowane w kodzie Pythona przy użyciu CPU, natomiast GPU trenuje model, wykorzystując poprzednią paczkę obrazów. Tak więc te metody generowania sztucznych danych nie obciążają procesora”[48].

W przypadku przetwarzania języka naturalnego można losowo zastąpić dane słowo podobnym przy założeniu, że nie zmieni to znaczenia zdania lub wyrażonych w nim emocji (tabela 4.9). Podobne słowa można znaleźć w słowniku synonimów albo poprzez wyszukanie w przestrzeni osadzania takich, których osadzenia są bliskie sobie.

*Tabela 4.9. Trzy zdania wygenerowane na podstawie oryginalnego*

| Zdanie oryginalne | Jestem tak szczęśliwy, że cię spotkałem. |
|---|---|
| Zdania wygenerowane | Jestem tak *zadowolony*, że cię spotkałem. |
| | Jestem tak szczęśliwy, że cię *zobaczyłem*. |
| | Jestem *bardzo* szczęśliwy, że cię spotkałem. |

Ten sposób generowania sztucznych danych pozwala na szybkie podwojenie, a nawet potrojenie ilości danych treningowych.

## Perturbacja

Perturbacja jest również operacją zachowującą etykiety. Ponieważ jest ona czasami używana do oszukiwania modeli w celu wygenerowania błędnych prognoz, stwierdziłam, że zasługuje na oddzielną analizę.

Ogólnie rzecz ujmując, sieci neuronowe są wrażliwe na zakłócenia. W przypadku widzenia komputerowego oznacza to, że dodanie niewielkiej ilości szumu do obrazu może spowodować, że sieć neuronowa źle go sklasyfikuje. Su i in. wykazali, że 67,97% naturalnych obrazów w zestawie danych testowych Kaggle CIFAR-10 oraz 16,04% obrazów testowych ImageNet może zostać błędnie sklasyfikowanych poprzez zmianę tylko jednego piksela (patrz rysunek 4.12)[49].

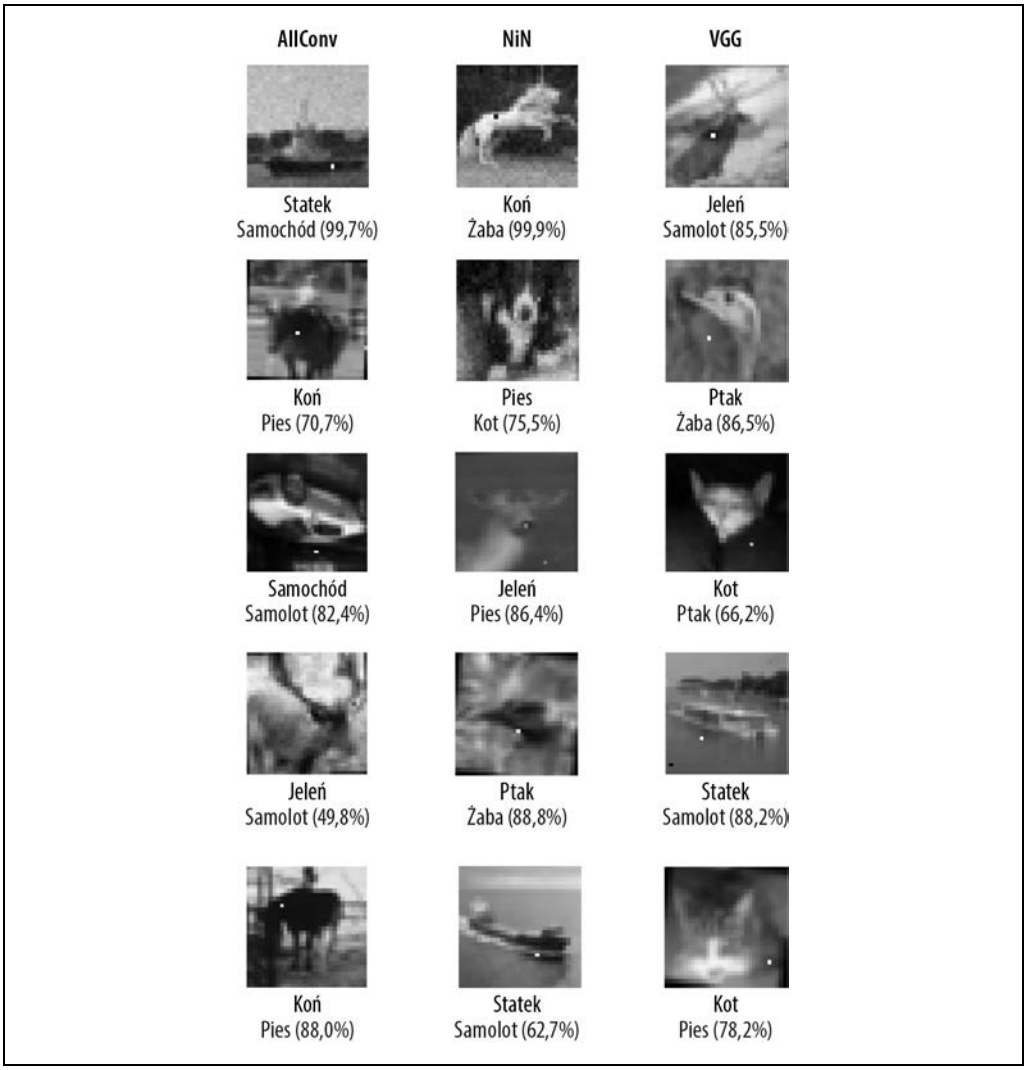
Używanie błędnych danych, aby zmusić sieć neuronową do wygenerowania błędnych prognoz, jest nazywane podstępnym atakiem. Dodawanie zakłóceń do próbek to powszechnie stosowana technika tworzenia danych wykorzystywanych w takich atakach. Podstępny atak jest szczególnie wydajny w przypadku wysokiej rozdzielczości obrazów.

[48] Alex Krizhevsky, Ilya Sutskever i Geoffrey E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, 2012, *https://oreil.ly/aphzA*.

[49] Jiawei Su, Danilo Vasconcellos Vargas i Sakurai Kouichi, *One Pixel Attack for Fooling Deep Neural Networks*, „IEEE Transactions on Evolutionary Computation”, 23, nr 5, 2019, s. 828 – 841, *https://oreil.ly/LzN9D*.



Kup książkę Poleć książkę




*Rysunek 4.12. Zmiana jednego piksela może spowodować, że sieć neuronowa zwróci błędne prognozy. W doświadczeniu użyto modeli AllConv, NiN i VGG. Oryginalne etykiety wygenerowane przez te modele znajdują się powyżej etykiet uzyskanych po zmianie jednego piksela. Źródło: Su i in.*[50]

Dodanie zakłóconych próbek do danych treningowych może pomóc modelom rozpoznać słabe punkty w poznanych granicach decyzyjnych i poprawić wydajność[51]. Takie próbki mogą być tworzone zarówno przez dodanie losowego szumu, jak i przez strategię wyszukiwania. Moosavi-Dezfooli

[50] Su i in., *One Pixel Attack*.

[51] Ian J. Goodfellow, Jonathon Shlens i Christian Szegedy, *Explaining and Harnessing Adversarial Examples*, arXiv, 20 marca 2015, *https://oreil.ly/9v2No*; Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville I Yoshua Bengio, *Maxout Networks*, arXiv, 18 lutego 2013, *https://oreil.ly/L8mch*.



Kup książkę Poleć książkę

i in. zaproponowali algorytm o nazwie DeepFool, który znajduje minimalną możliwą ilość zakłóceń potrzebną do nieprawidłowej klasyfikacji obrazu, co do której model ma wysoką pewność[52]. Taka metoda jest nazywana antagonistycznym generowaniem sztucznych danych[53].

Jest ona mniej popularna w przypadku problemów dotyczących przetwarzania języka naturalnego (na obrazie niedźwiedzia z losowo dodanymi pikselami nadal można rozpoznać niedźwiedzia, ale dodanie losowych znaków do losowego zdania prawdopodobnie spowoduje, że stanie się ono niezrozumiałe). Perturbacje są jednak używane w celu uczynienia modeli bardziej odpornymi. Jednym z najbardziej znanych przykładów jest algorytm BERT, w którym model wybiera losowo 15% ze wszystkich tokenów z każdej sekwencji, a następnie decyduje się zastąpić 10% wybranych losowymi słowami. Weźmy na przykład pod uwagę zdanie: „Mój pies jest kudłaty". Model może zastąpić słowo „kudłaty" losowo wybranym słowem „jabłko". Otrzymamy więc zdanie: „Mój pies jest jabłkiem". Jak widać, zastąpienie 1,5% wszystkich tokenów może spowodować wygenerowanie nonsensownej konstrukcji. Badania ablacyjne pokazują, że w wyniku losowego zastępowania osiąga się niewielki wzrost wydajności[54].

W rozdziale 6. zostanie przeanalizowane, w jaki sposób można używać perturbacji nie tylko w celu poprawy wydajności modelu, ale także by móc ją oceniać.

## Synteza danych

Ponieważ gromadzenie danych jest kosztowne i powolne, a do tego wiąże się z wieloma potencjalnymi problemami dotyczącymi prywatności, idealnie byłoby, gdybyśmy mogli całkowicie pominąć ten proces, a modele trenować za pomocą zsyntetyzowanych danych. Mimo że wciąż nie ma możliwości syntezy wszystkich danych treningowych, potrafimy już syntetyzować niektóre z nich w celu zwiększenia wydajności modelu.

W przypadku przetwarzania języka naturalnego podczas uruchamiania modelu można wykorzystywać szablony. Jeden z zespołów, z którym współpracowałam, używał szablonów, aby przygotować dane treningowe dla konwersacyjnej aplikacji sztucznej inteligencji (chatbota). Szablon mógłby wyglądać następująco: „Znajdź restaurację [RESTAURACJA] w promieniu [LICZBA] kilometrów od [LOKALIZACJA]" (patrz tabela 4.10). Mając dla każdego miasta listę wszystkich możliwych restauracji i lokalizacji (dom, biuro, punkty orientacyjne, dokładne adresy), a także posługując się rozsądnymi odległościami (prawdopodobnie nigdy nie chciałbyś szukać restauracji oddalonych więcej niż 1000 kilometrów), możesz na podstawie takiego szablonu wygenerować tysiące zapytań treningowych.

[52] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi i Pascal Frossard, *DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks*, „Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", 2016, *https://oreil.ly/dYVL8*.

[53] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama i Shin Ishii, *Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning*, „IEEE Transactions on Pattern Analysis and Machine Intelligence", 2017, *https://oreil.ly/MBQeu*.

[54] Devlin i in., *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.



Kup książkę Poleć książkę

*Tabela 4.10. Trzy zdania wygenerowane na podstawie szablonu*

| Szablon | Znajdź restaurację [RESTAURACJA] w promieniu [LICZBA] kilometrów od [LOKALIZACJA]. |
|---|---|
| Wygenerowane zdania | Znajdź restaurację *wietnamską* w promieniu *2* kilometrów od *mojego biura*. |
| | Znajdź restaurację *włoską* w promieniu 5 kilometrów od *mojego domu*. |
| | Znajdź restaurację *meksykańską* w promieniu 3 kilometrów od *dworca kolejowego w Katowicach*. |

Prostym sposobem syntetyzowania nowych danych w przypadku widzenia komputerowego jest połączenie istniejących przykładów z etykietami dyskretnymi w celu wygenerowania etykiet ciągłych. Rozważmy zadanie klasyfikacji obrazów z dwoma możliwymi etykietami: PIES (zakodowana jako 0) i KOT (zakodowana jako 1). Z przykładu $x_1$ z etykietą PIES i przykładu $x_2$ z etykietą KOT można wygenerować następujący przykład $x'$:

$$x' = \gamma x_1 + (1 - \gamma)x_2$$

Etykieta $x'$ jest kombinacją etykiet $x_1$ i $x_2$: $\gamma \cdot 0 + (1 - \gamma) \cdot 1$. Wykorzystana metoda jest nazywana **mixupem**. Autorzy wykazali, że mixup poprawia poziom uogólnienia modeli, utrudnia zapamiętywanie uszkodzonych etykiet, zwiększa ich odporność na podstępne przykłady i stabilizuje trening generatywnych sieci przeciwstawnych[55].

Wykorzystanie sieci neuronowych do syntezy danych treningowych to ekscytujące rozwiązanie, które jest aktywnie badane. Nie zostało ono jednak jeszcze powszechnie wdrożone w środowiskach produkcyjnych. Sandfort i in. wykazali, że po dodaniu do oryginalnych danych treningowych obrazów wygenerowanych za pomocą algorytmu CycleGAN znacząco wzrosła wydajność modelu w zadaniach segmentacji tomografii komputerowej[56].

Aby dowiedzieć się więcej o generowaniu sztucznych danych w widzeniu komputerowym, zapoznaj się z kompleksową analizą *A Survey on Image Data Augmentation for Deep Learning* (*https://oreil.ly/3TUpK*) (Shorten i Khoshgoftaar, 2019).

# Podsumowanie

Dane treningowe wciąż stanowią podstawę nowoczesnych algorytmów uczenia maszynowego. Bez względu na to, jak mądre są Twoje algorytmy, nie będą poprawnie działać, jeśli zostaną z nimi użyte błędne dane treningowe. Warto zainwestować czas i wysiłek, aby zebrać i utworzyć takie dane treningowe, które pozwolą Twoim algorytmom nauczyć się czegoś sensownego.

W tym rozdziale przeanalizowaliśmy wiele sposobów tworzenia danych treningowych. Najpierw omówiliśmy różne metody generowania próbek, zarówno próbkowanie nieprobabilistyczne, jak i losowe, które mogą pomóc w doborze odpowiednich danych dla modelu.

[55] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin i David Lopez-Paz, *mixup: Beyond Empirical Risk Minimization*, „ICLR 2018", *https://oreil.ly/lIM5E*.

[56] Veit Sandfort, Ke Yan, Perry J. Pickhardt i Ronald M. Summers, *Data Augmentation Using Generative Adversarial Networks (CycleGAN) to Improve Generalizability in CT Segmentation Tasks*, „Scientific Reports" 9, nr 1, 2019, str. 16884, *https://oreil.ly/TDUwm*.



Kup książkę Poleć książkę

Większość używanych obecnie metod uczenia maszynowego stanowią algorytmy nadzorowane, więc uzyskanie etykiet jest integralną częścią tworzenia danych treningowych. Wiele zadań, takich jak szacowanie czasu dostawy lub systemy rekomendacji, wykorzystuje etykiety naturalne. Pojawiają się one zazwyczaj z opóźnieniem, a czas, jaki upływa od momentu podania prognozy do momentu przekazania informacji zwrotnej, jest zwany długością pętli sprzężenia zwrotnego. Etykiety naturalne są dość powszechnie spotykane w środowisku produkcyjnym, co może oznaczać, że firmy wolą rozpoczynać wdrażanie uczenia maszynowego od zadań z takimi etykietami.

W przypadku problemów, które nie wykorzystują etykiet naturalnych, przedsiębiorstwa zazwyczaj polegają na osobach komentujących dane. Etykietowanie ręczne ma jednak wiele wad. Na przykład może być drogie i powolne. Istnieją alternatywne rozwiązania pozwalające na radzenie sobie z brakiem etykiet ręcznych. Przeanalizowaliśmy takie z nich jak nadzór słaby, półnadzór, uczenie transferowe i uczenie aktywne.

Algorytmy uczenia maszynowego działają lepiej w sytuacjach, gdy rozkład danych jest bardziej zrównoważony. Mają jednak problemy w przypadku, gdy klasy są mocno niezrównoważone. Niestety jest to sytuacja często spotykana w świecie rzeczywistym. W dalszej części rozdziału wyjaśniliśmy, dlaczego niezrównoważenie klas utrudnia algorytmom ML uczenie się. Omówiliśmy również różne techniki radzenia sobie z niezrównoważeniem klas, takie jak wybór odpowiedniego wskaźnika, ponowne próbkowanie danych, czy też modyfikacja funkcji straty w celu zachęcenia modelu do zwrócenia uwagi na pewne próbki.

Rozdział zakończyliśmy analizą technik sztucznego generowania danych, które mogą być wykorzystane do poprawy wydajności modelu i poziomu jego uogólnienia zarówno w przypadku problemów związanych z widzeniem komputerowym, jak i przetwarzaniem języka naturalnego.

Po uzyskaniu danych treningowych należy wyodrębnić z nich cechy, co umożliwi wytrenowanie modeli uczenia maszynowego. To zagadnienie zostanie przeanalizowane w następnym rozdziale.



Kup książkę Poleć książkę

Kup książkę Poleć książkę


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

# Wdrażaj i skaluj modele tak, by uzyskiwać najlepsze wyniki!

Systemy uczenia maszynowego (ML) charakteryzują się złożonością i unikatowością. Zmiana w jednym z wielu komponentów może istotnie wpłynąć na całość. Zastosowane w modelach dane diametralnie różnią się od siebie w poszczególnych przypadkach użycia. To wszystko sprawia, że bardzo trudno jest stworzyć taki system, jeśli każdy komponent zostaje zaprojektowany oddzielnie. Aby zbudować aplikację korzystającą z ML i nadającą się do wdrożenia w środowisku produkcyjnym, konieczne jest podejmowanie decyzji projektowych z uwzględnieniem cech systemu jako całości.

To książka przeznaczona dla inżynierów, którzy chcą stosować systemy uczenia maszynowego do rozwiązywania rzeczywistych problemów biznesowych. Zaprezentowano w niej systemy ML używane w szybko rozwijających się startupach, a także przedstawiono holistyczne podejście do ich projektowania — z uwzględnieniem różnych komponentów systemu i celów osób zaangażowanych w proces. Dużo uwagi poświęcono analizie decyzji projektowych, dotyczących między innymi sposobu tworzenia i przetwarzania danych treningowych, wyboru wskaźników, częstotliwości ponownego treningu modelu czy techniki monitorowania pracy aplikacji. Zaprezentowana tu koncepcja iteracyjna natomiast pozwala na uzyskanie pewności, że podejmowane decyzje są optymalne z punktu widzenia pracy całości systemu. Co ważne, poszczególne zagadnienia zostały zilustrowane rzeczywistymi studiami przypadków.

## W książce między innymi:

- wybór wskaźników właściwych dla danego problemu biznesowego
- automatyzacja ciągłego rozwoju, ewaluacji, wdrażania i aktualizacji modeli
- szybkie wykrywanie i rozwiązywanie problemów podczas wdrożenia produkcyjnego
- tworzenie wszechstronnej platformy ML
- odpowiedzialne tworzenie systemów ML

**Chip Huyen** zajmowała się tworzeniem i wdrażaniem systemów ML dla takich firm jak NVIDIA, Netflix czy Snorkel AI. Brała też udział w projektowaniu Claypot AI, działającej w czasie rzeczywistym platformy do uczenia maszynowego. Jest autorką kursu CS 329S dotyczącego projektowania systemów uczenia maszynowego, dostępnego na Uniwersytecie Stanforda.

---

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-283-9912-9
9 788328 399129
Cena: 89,00 zł