Technologia i rozwiązania

Python
Uczenie maszynowe

Sebastian Raschka

Helion

[PACKT] PUBLISHING open source* community experience distilled

Tytuł oryginału: Python Machine Learning

Tłumaczenie: Krzysztof Sawka

ISBN: 978-83-283-3613-1

First published in the English language under the title 'Python Machine Learning - (9781783555130)'.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

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

Printed in Poland.

Spis treści

ROZDZIAŁ 1

Umożliwianie komputerom uczenia się z danych

Moim zdaniem **uczenie maszynowe** (ang. *machine learning*) — dział zajmujący się teorią i praktycznym zastosowaniem algorytmów analizujących dane — stanowi najciekawszą dziedzinę informatyki! Żyjemy w czasach przetwarzania olbrzymiej ilości informacji; za pomocą samouczących się algorytmów będących częścią uczenia maszynowego informacje te są przekształcane w rzeczywistą wiedzę. Dzięki licznym i potężnym bibliotekom o jawnym kodzie źródłowym, które powstały w ostatnich latach, prawdopodobnie teraz jest najlepszy czas, aby zainteresować się uczeniem maszynowym i nauczyć się wykorzystywać potężne algorytmy do wykrywania wzorców w przetwarzanych danych oraz prognozować przyszłe zdarzenia.

W tym rozdziale poznamy podstawowe pojęcia związane z uczeniem maszynowym oraz zdefiniujemy jego różne rodzaje. Dzięki wprowadzeniu podstawowej terminologii wspólnie położymy fundament pod skuteczne wykorzystywanie technik uczenia maszynowego w rozwiązywaniu praktycznych problemów.

Przyjrzymy się teraz następującym zagadnieniom:

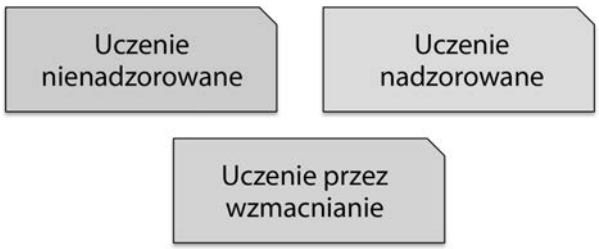
- ogólne pojęcia dotyczące uczenia maszynowego,
- trzy rodzaje uczenia się oraz podstawowa terminologia,
- główne elementy wykorzystywane w skutecznym projektowaniu systemów uczenia maszynowego,
- instalacja oraz konfiguracja środowiska Python pod kątem analizowania danych i uczenia maszynowego.

Tworzenie inteligentnych maszyn służących do przekształcania danych w wiedzę

W erze współczesnej technologii istnieje pewien zasób, którego mamy pod dostatkiem: mianowicie olbrzymie ilości ustrukturyzowanych i nieustrukturyzowanych danych. W drugiej połowie XX wieku uczenie maszynowe wyewoluowało z badań nad **sztuczną inteligencją**, w których projektowano samouczące się algorytmy, zdolne do pozyskiwania wiedzy z informacji oraz tworzenia na ich podstawie prognoz. Dzięki uczeniu maszynowemu nie trzeba zatrudniać ludzi do ręcznego określania reguł oraz tworzenia modeli poprzez analizowanie olbrzymich pokładów danych; omawiana dziedzina wiedzy oferuje efektywniejsze rozwiązanie polegające na stopniowym poprawianiu skuteczności modeli predykcyjnych oraz podejmowaniu decyzji na podstawie analizowanych danych. Uczenie maszynowe nie tylko staje się coraz istotniejszą częścią nauk informatycznych, lecz również odgrywa coraz większą rolę w naszym codziennym życiu. Dzięki metodom opracowanym pod kątem uczenia maszynowego możemy cieszyć się zaawansowanymi filtrami antyspamowymi w poczcie e-mail, wygodnym oprogramowaniem do rozpoznawania mowy i tekstu, rzetelnymi silnikami wyszukiwarek internetowych, wymagającymi programami szachowymi, a także wkrótce (miejmy nadzieję) bezpiecznymi i wydajnymi pojazdami samojezdnymi.

Trzy różne rodzaje uczenia maszynowego

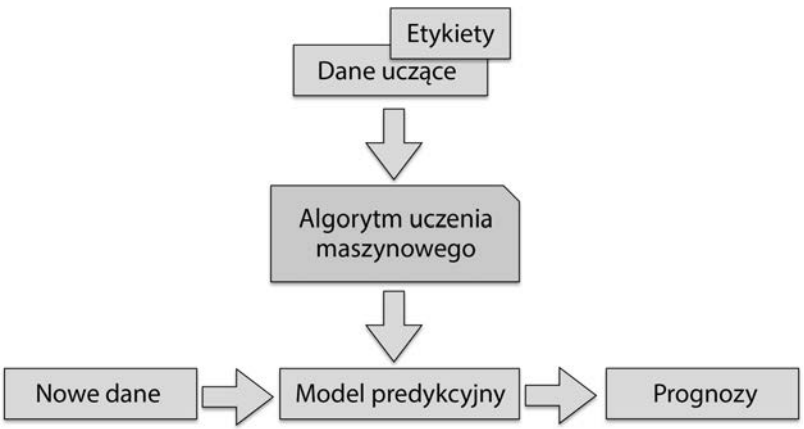
W tym podrozdziale zapoznamy się z trzema odmianami uczenia maszynowego (rysunek 1.1): **uczeniem nadzorowanym**, **uczeniem nienadzorowanym** oraz **uczeniem przez wzmacnianie**. Poznamy podstawowe różnice pomiędzy wspomnianymi typami nauki oraz, korzystając z odpowiednich przykładów, nauczymy się intuicyjnie rozpoznawać, które rodzaje uczenia najlepiej nadają się do rozwiązywania określonych kategorii problemów.



Rysunek 1.1. Podstawowe typy uczenia maszynowego

Prognozowanie przyszłości za pomocą uczenia nadzorowanego

Głównym celem uczenia nadzorowanego (ang. *supervised learning*; rysunek 1.2) jest uczenie modelu za pomocą oznakowanych **danych uczących** (ang. *training data*), co pozwala przewidywać niewidoczne lub wygenerowane w przyszłości informacje. W tym przypadku człon **nadzorowane** odnosi się do zestawu próbek, w których pożądane sygnały wyjściowe (etykiety) są znane.



Rysunek 1.2. Ogólny schemat uczenia nadzorowanego

Za przykład weźmy filtr antyspamowy: możemy trenować dany model, stosując algorytm nadzorowanego uczenia maszynowego wobec treści oznakowanych wiadomości e-mail (poprawnie oznaczonych jak spam lub wartościowe wiadomości), dzięki czemu system jest w stanie przewidywać, czy przychodząca korespondencja zalicza się do jednej z tych dwóch kategorii. Czynność nadzorowanego uczenia przy użyciu dyskretnych **etykiet klas**, zaprezentowana na powyższym przykładzie, nazywana jest również czynnością **klasyfikacji**. Kolejną podkategorią uczenia nadzorowanego jest **regresja**, w której sygnał wyjściowy przyjmuje wartości ciągłe.

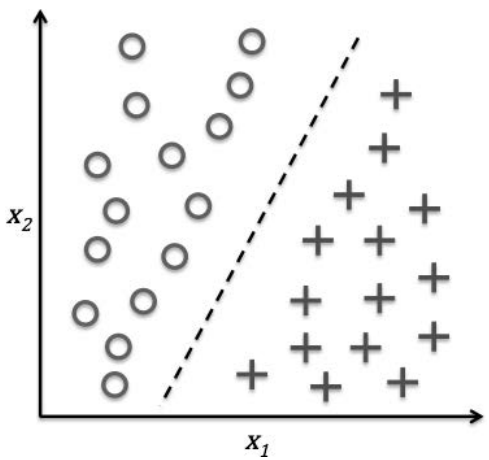
Klasyfikacja — przewidywanie etykiet klas

Klasyfikacja stanowi podkategorię uczenia nadzorowanego służącą do przewidywania etykiet klas w nowych wystąpieniach na podstawie dotychczasowych obserwacji. Etykiety klas to dyskretne, nieuporządkowane wartości, które określają **przynależność** poszczególnych instancji do wyznaczonych grup. Wspomniany wcześniej przykład filtru antyspamowego jest typowym reprezentantem **klasyfikacji binarnej**, w której algorytm uczenia maszynowego uczy się zestawu reguł w celu rozróżniania dwóch możliwych klas: spamu oraz użytecznych wiadomości e-mail.

Jednak zbiór etykiet klas wcale nie musi mieć charakteru binarnego. Model predykcyjny wyuczony za pomocą algorytmu uczenia nadzorowanego może przydzielić dowolną, zaprezentowaną

w zestawie danych uczących etykietę klas do nowego, nieoznakowanego wystąpienia. Klasycznym przykładem takiej **klasyfikacji wieloklasowej** jest rozpoznawanie odręcznego pisma. W tym przypadku możemy stworzyć zestaw danych uczących składający się z wielu próbek odręcznego pisma każdej litery alfabetu. Jeśli użytkownik wprowadzi do urządzenia jakąś odręcznie napisaną literę, nasz model predykcyjny będzie w stanie rozpoznać ten znak z określoną dokładnością. Jeżeli jednak nie dołączymy symboli cyfr arabskich (0 – 9) do danych uczących, to nasz system nie będzie ich rozpoznawał.

Na rysunku 1.3 zaprezentowałem koncepcję klasyfikacji binarnej, w której wykorzystano 30 próbek uczących: 15 próbek zostało oznaczonych jako **klasa negatywna** (kółka), a pozostałym próbkom przydzieliłem etykiety **klasy pozytywnej** (krzyżyki). W takiej sytuacji zbiór danych jest dwuwymiarowy, co oznacza, że każda próbka zawiera dwie wartości: x_1 oraz x_2. Teraz wykorzystujemy algorytm nadzorowanego uczenia maszynowego do nauki reguły — granicy decyzyjnej symbolizowanej przez czarną, przerywaną linię — która rozdziela te dwie klasy oraz klasyfikuje analizowane dane do jednej z tych dwóch kategorii w zależności od wartości parametrów x_1 i x_2.



Rysunek 1.3. Przykład klasyfikacji binarnej

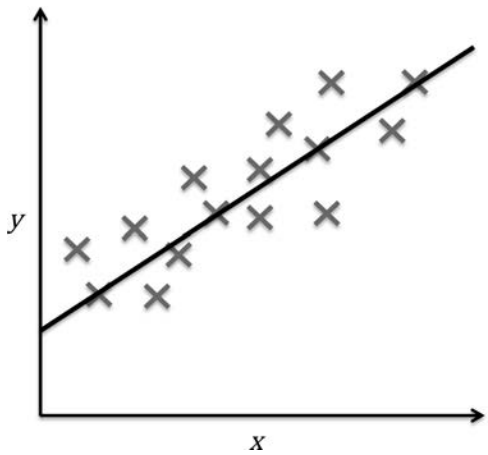
Regresja dla przewidywania wyników ciągłych

Z poprzedniego podrozdziału dowiedzieliśmy się, że klasyfikacja służy do przydzielania kategoryzowanych, nieuporządkowanych etykiet do wystąpień. Drugim rodzajem uczenia nadzorowanego jest prognozowanie wyników ciągłych, czyli tzw. **analiza regresji**. W tym modelu mamy dane zmienne **objaśniające** (prognozujące) oraz ciągłą zmienną **objaśnianą** (prognozowaną), naszym zadaniem natomiast jest odkrycie relacji pomiędzy tymi zmiennymi, co pozwoli przewidywać przyszłe wyniki.

Załóżmy, że interesuje nas prognozowanie wyników egzaminów z matematyki naszych studentów. Jeżeli istnieje związek pomiędzy czasem przeznaczonym na naukę a ocenami, możemy wykorzystać te informacje jako zestaw danych uczących do trenowania modelu do przewidywania ocen przyszłych studentów planujących zdawać testy matematyczne.

Pojęcie regresji zostało ukute przez Francisa Galtona w artykule *Regression Towards Mediocrity in Hereditary Stature* (regresja w kierunku przeciętności w dziedziczeniu postury) z 1886 roku. Autor opisał zjawisko biologiczne, zgodnie z którym zróżnicowanie **wzrostu** w populacji nie ulega zwiększeniu wraz z upływem czasu. Zaobserwował, że wzrost rodziców nie jest przekazywany dzieciom, lecz wzrost potomków zmierza ku średnim wartościom populacji.

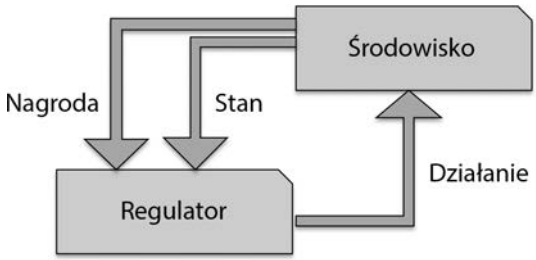
Na rysunku 1.4 zaprezentowana została koncepcja **regresji liniowej**. Mając zmienną objaśniającą x i zmienną objaśnianą y, wyznaczamy dla przykładowych danych prostą przebiegającą w jak najmniejszej odległości od zestawu próbek — uzyskujemy ją najczęściej metodą najmniejszych kwadratów. Możemy teraz wykorzystać punkt przecięcia tej prostej z osiami współrzędnych oraz jej nachylenie do przewidywania wyników pochodzących z nowych danych.



Rysunek 1.4. Przykład regresji liniowej

Rozwiązywanie problemów interaktywnych za pomocą uczenia przez wzmacnianie

Kolejnym rodzajem uczenia maszynowego jest uczenie przez wzmacnianie (ang. *reinforcement learning*). W tym przypadku celem jest utworzenie systemu (**regulatora**, **agenta**), który poprawia własną skuteczność na podstawie interakcji ze **środowiskiem**. Informacje na temat bieżącego stanu środowiska zazwyczaj zawierają także tzw. sygnał **nagrody**, dlatego możemy uznać uczenie przez wzmacnianie jako model powiązany z uczeniem **nadzorowanym**. Jednak w przypadku uczenia przez wzmacnianie sprzężeniem tym nie są poprawne, wzorcowe etykiety lub wartości, lecz wartość skuteczności pomiaru działania przez **funkcję nagrody**. Poprzez oddziaływanie ze środowiskiem regulator może wykorzystywać uczenie przez wzmacnianie do trenowania szeregu działań dążących do maksymalizowania nagrody metodą prób i błędów lub rozważnego planowania (rysunek 1.5).



Rysunek 1.5. Oddziaływania w modelu uczenia przez wzmacnianie

Popularnym przykładem uczenia przez wzmacnianie jest silnik aplikacji szachowej. Regulator wybiera kolejne ruchy figur szachowych na podstawie stanu szachownicy (środowiska), a nagrodę można zdefiniować jako **zwycięstwo** lub **porażkę** na koniec rozgrywki.

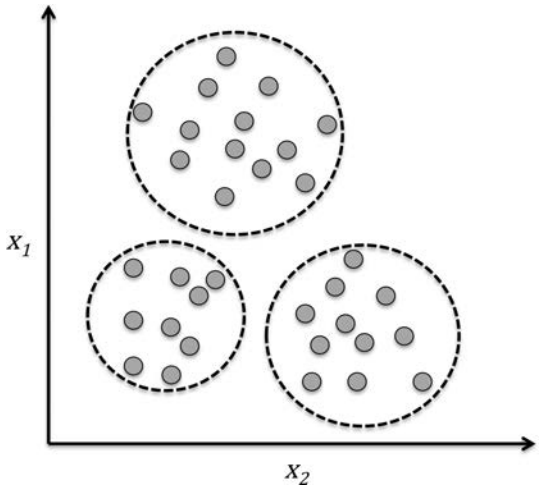
Odkrywanie ukrytych struktur za pomocą uczenia nienadzorowanego

W przypadku uczenia nadzorowanego znamy **właściwą odpowiedź** jeszcze przed rozpoczęciem trenowania danego modelu, z kolei w uczeniu przez wzmacnianie wykorzystujemy regulator do definiowania wartości **nagród** dla poszczególnych działań. Natomiast korzystając z technik uczenia nienadzorowanego (ang. *unsupervised learning*), mamy do czynienia z nieoznakowanymi danymi lub danymi o **nieznanej strukturze**. Dzięki modelom uczenia nienadzorowanego jesteśmy w stanie poznawać strukturę przetwarzanych danych i uzyskiwać użyteczne informacje bez stosowania znanej zmiennej wyjściowej lub funkcji nagrody.

Wyznaczanie podzbiorów za pomocą grupowania

Grupowaniem (**klasteryzacją**, **analizą skupień**; ang. *clustering*) nazywamy technikę badawczą analizy danych pozwalającą na organizowanie zestawów informacji w sensowne podzbiory (**klastry**, **grupy**, **skupienia**) bez uprzedniej wiedzy na temat przydziału grupowego poszczególnych danych. Każdy klaster powstający w wyniku analizy definiuje zbiór obiektów wykazujących między sobą pewne podobieństwa i odróżniających się od elementów umieszczonych w pozostałych grupach, dlatego grupowanie czasami jest nazywane „nienadzorowaną klasyfikacją". Ta technika nadaje się znakomicie do strukturyzowania informacji oraz wyznaczania istotnych powiązań pomiędzy danymi; np. pozwala sprzedawcom odkrywać grupy klientów według ich zainteresowań, co jest wykorzystywane do tworzenia oddzielnych programów marketingowych.

Na rysunku 1.6 pokazuję, w jaki sposób można wykorzystać klasteryzację do zorganizowania danych w trzy oddzielne grupy na podstawie podobieństw cech x_1 i x_2.



Rysunek 1.6. Wyznaczanie osobnych grup za pomocą klasteryzacji

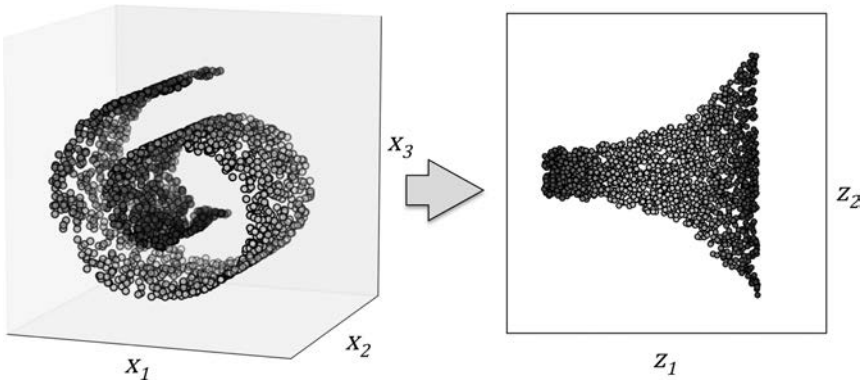
Redukowanie wymiarowości w celu kompresji danych

Kolejną dziedziną uczenia nienadzorowanego jest **redukcja wymiarowości** (ang. *dimensionality reduction*). Często pracujemy z danymi wielowymiarowymi — każda obserwacja daje nam dużą liczbę wartości pomiarowych — co stanowi wyzwanie w przypadku ograniczonej pojemności nośników danych oraz skuteczności obliczeniowej algorytmów uczenia maszynowego. Nienadzorowana redukcja wymiarowości jest stosowana powszechnie we wstępnym przetwarzaniu cech w celu wykluczenia szumu z danych — który może zmniejszać skuteczność predykcji niektórych algorytmów — a do tego kompresuje dane do podprzestrzeni o mniejszej liczbie wymiarów przy zachowaniu większości istotnej informacji.

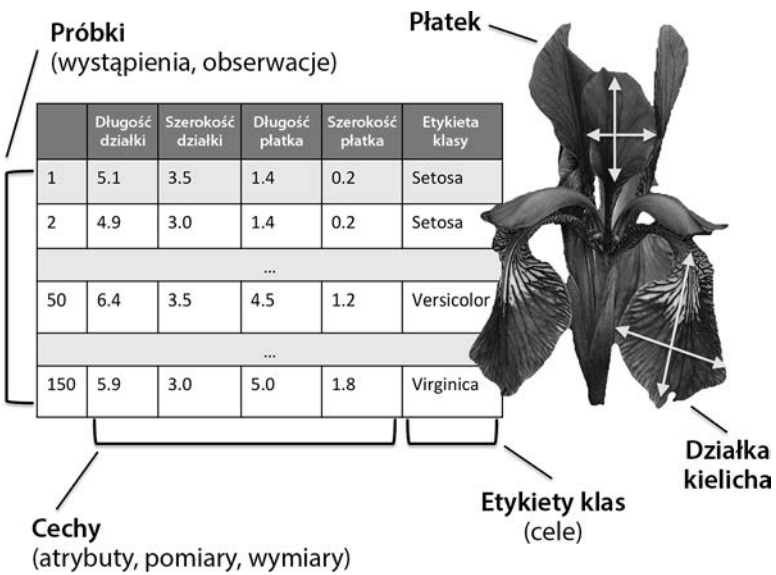
Czasami redukowanie wymiarowości przydaje się również do wizualizacji danych — np. zbiór cech wielowymiarowych można rzutować na jedno-, dwu- lub trójwymiarowe przestrzenie cech w celu wyświetlenia go w formie dwu- lub trójwymiarowego wykresu punktowego albo histogramu. Rysunek 1.7 zawiera przykład zastosowania nieliniowej redukcji wymiarowości do skompresowania wykresu trójwymiarowego do nowej dwuwymiarowej podprzestrzeni cech.

Wprowadzenie do podstawowej terminologii i notacji

Po omówieniu trzech ogólnych kategorii uczenia maszynowego — nadzorowanego, nienadzorowanego oraz przez wzmacnianie — przyszedł czas na zapoznanie się z podstawową terminologią, która będzie wykorzystywana w kolejnych rozdziałach. Na rysunku 1.8 widzimy tabelę stanowiącą fragment zestawu danych **Iris**, będącego klasycznym przykładem w dziedzinie



Rysunek 1.7. Przykład redukowania wymiarowości



	Długość działki	Szerokość działki	Długość płatka	Szerokość płatka	Etykieta klasy
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Rysunek 1.8. Fragment zestawu danych Iris

uczenia maszynowego. Na zbiór tych danych składają się wyniki pomiarów 150 kwiatów kosaćca z trzech różnych gatunków: *Setosa* (kosaciec szczecinkowy), *Versicolor* (kosaciec różnobarwny) oraz *Virginica* (kosaciec wirginijski). Każdy kwiat reprezentuje tu oddzielny wiersz w zbiorze danych, natomiast wyniki pomiarów (w centymetrach) są przechowywane w kolumnach, zwanych także zbiorami cech.

Aby zachować prostotę i czytelność notacji oraz implementacji, wprowadzimy pewne podstawy **algebry liniowej**. W następnych rozdziałach będę opisywać dane w notacji **macierzowej** oraz **wektorowej**. Wykorzystam popularną konwencję, zgodnie z którą każda próbka jest reprezentowana jako osobny wiersz w macierzy cech X, gdzie poszczególne cechy są przechowywane w oddzielnych kolumnach.

Zestaw danych Iris, składający się ze 150 próbek i z 4 cech, możemy zapisać jako macierz o rozmiarze 150×4 lub $\boldsymbol{X} \in \mathbb{R}^{150\times 4}$:

$$\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(150)} & x_2^{(150)} & x_3^{(150)} & x_4^{(150)} \end{bmatrix}$$

W dalszej części książki będziemy wykorzystywać indeks górny (*i*) do określenia *i*-tej próbki uczącej, a indeks dolny (*j*) będzie odnosił się do *j*-tego wymiaru zbioru uczącego.

Pogrubionymi małymi literami będziemy oznaczać wektory: ($\boldsymbol{x} \in \mathbb{R}^{n\times 1}$), z kolei pogrubionymi dużymi literami będziemy określać macierze: ($\boldsymbol{X} \in \mathbb{R}^{n\times m}$). Pojedyncze elementy wektora lub macierzy definiujemy literami oznaczonymi kursywą (odpowiednio: $x^{(n)}$ lub $x_{(m)}^{(n)}$).

Na przykład element x_1^{150} oznacza pierwszy wymiar (czyli długość działki) próbki numer 150. W ten sposób każdy wiersz w tej macierzy cech reprezentuje jedno wystąpienie kwiatu, które można zapisać w postaci czteroelementowego wektora $\boldsymbol{x}^{(i)} \in \mathbb{R}^{1\times 4}$,. $\boldsymbol{x}^{(i)} = \begin{bmatrix} x_1^{(i)} & x_2^{(i)} & x_3^{(i)} & x_4^{(i)} \end{bmatrix}$

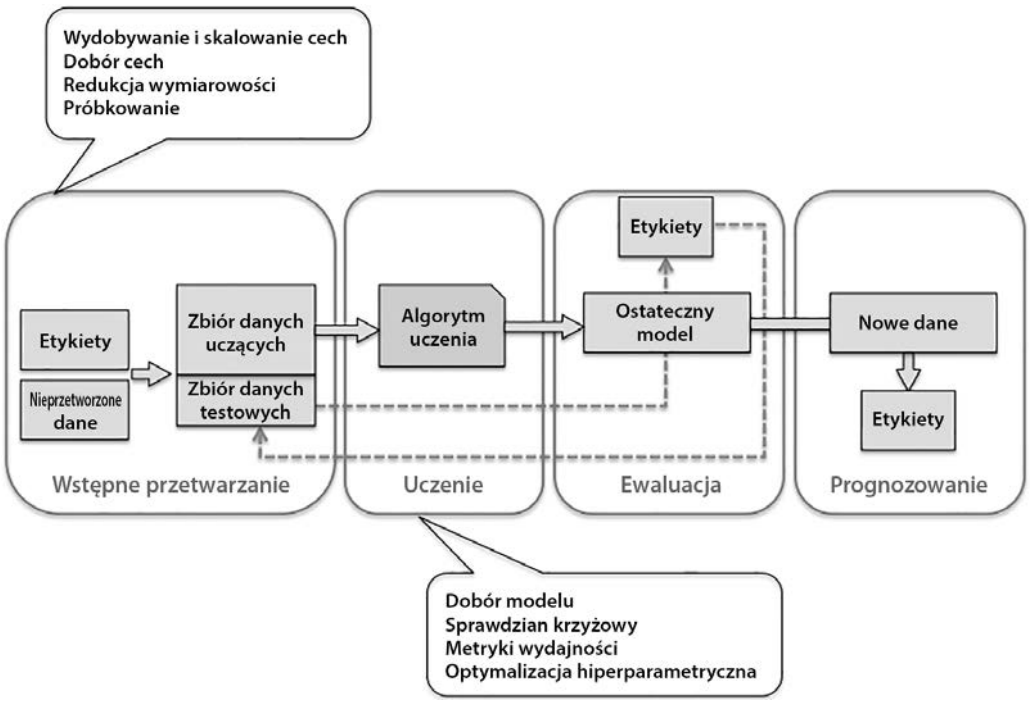
Każdy wymiar cechy jest 150-elementowym wektorem kolumnowym $\boldsymbol{x}_j \in \mathbb{R}^{150\times 1}$, np.:

$$\boldsymbol{x}_j = \begin{bmatrix} x_j^{(1)} \\ x_j^{(2)} \\ \vdots \\ x_j^{(150)} \end{bmatrix}$$

W podobny sposób przechowujemy zmienne docelowe (tutaj: etykiety klas), jako 150-elementowy wektor kolumnowy $\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(150)} \end{bmatrix}$ ($y \in$ {Setosa, Versicolor, Virginica}).

Strategia tworzenia systemów uczenia maszynowego

W poprzednich podrozdziałach omówiliśmy podstawowe pojęcia uczenia maszynowego oraz poznaliśmy trzy jego rodzaje. Przyjrzymy się teraz innym istotnym elementom systemu uczenia maszynowego towarzyszącym algorytmom uczenia. Na rysunku 1.9 zaprezentowałem schemat typowego przebiegu procesów podczas stosowania uczenia maszynowego w **modelowaniu predykcyjnym**. Zagadnienie to rozwiniemy w dalszej części rozdziału.



Rysunek 1.9. Modelowanie predykcyjne

Wstępne przetwarzanie — nadawanie danym formy

Rzadko kiedy nieprzetworzone dane mają postać umożliwiającą wykorzystanie optymalnej skuteczności algorytmu uczącego. Z tego powodu **wstępne przetwarzanie** (ang. *preprocessing*) danych stanowi jeden z najważniejszych etapów każdego rodzaju uczenia maszynowego. Weźmy za przykład omówiony we wcześniejszej części rozdziału zestaw danych Iris — danymi nieprzetworzonymi mogą być w tym przypadku zdjęcia kwiatów, z których chcemy wydobyć jak najwięcej sensownych cech. Do takich cech możemy zaliczyć barwę, odcień, intensywność koloru, wysokość rośliny, a także długość i szerokość elementów anatomicznych kwiatu. Wiele algorytmów uczenia maszynowego wymaga również, aby wybrane cechy były prezentowane w jednakowej skali, co często jest osiągane poprzez ich transformację do zakresu [0, 1] lub do standardowego rozkładu normalnego, ze średnią równą 0 i z wariancją równą 1, jak zostanie to przedstawione w dalszej części książki.

Niektóre wybrane cechy mogą być ze sobą ściśle skorelowane, a przez to w pewnym stopniu nadmiarowe. W takich sytuacjach przydają się techniki redukcji wymiarowości do skompresowania cech w przestrzeń o mniejszej liczbie wymiarów. Zmniejszenie przestrzeni cech pozwala na zaoszczędzenie przestrzeni dyskowej, a także na znaczne przyśpieszenie działania algorytmu uczącego.

Aby się dowiedzieć, czy nasz algorytm uczenia maszynowego działa dobrze nie tylko na zestawie danych uczących, ale i na innych danych, musimy również losowo rozdzielić zbiór danych na osobne zestawy danych uczących i testowych. Zbiór danych uczących służy do trenowania i optymalizowania modelu uczenia maszynowego, natomiast zestaw danych testowych przechowujemy do samego końca procesu i dzięki niemu oceniamy ostateczny model.

Trenowanie i dobór modelu predykcyjnego

W kolejnych rozdziałach przekonamy się, że stworzono wiele różnorodnych algorytmów uczenia maszynowego przeznaczonych do rozwiązywania różnych kategorii problemów. Istotnym faktem, który stanowi sedno słynnego twierdzenia Davida H. Wolperta (*no free lunch theorem*[1]), jest zrozumienie, że nie możemy zmusić systemu do nauki „za darmo" (D.H. Wolpert, *The Lack of A Priori Distinctions Between Learning Algorithms*, 1996; D.H. Wolpert, W.G. Macready, *No Free Lunch Theorems for Optimization*, 1997). Zgodnie z intuicją możemy powiązać tę koncepcję z popularnym powiedzeniem: ***Gdy twoim jedynym narzędziem jest młotek, wszystko zaczyna ci przypominać gwoździe*** (A. Maslow, 1966). Przełóżmy to na przykład: każdy algorytm klasyfikacji zawiera integralne założenia i żaden z modeli klasyfikacji nie przeważa nad innymi, jeżeli nie opracujemy założeń dotyczących danego zadania. W praktyce niezbędne okazuje się porównanie przynajmniej kilku różnych algorytmów w celu wytrenowania i doboru najbardziej skutecznego modelu. Zanim jednak będziemy w stanie porównać różne modele, musimy najpierw ustalić metrykę służącą do pomiaru wydajności. Jedną z najpopularniejszych metryk jest dokładność klasyfikacji, którą definiujemy jako stosunek poprawnie sklasyfikowanych wystąpień do nieprawidłowo określonych instancji.

Możesz w tym momencie zadać rozsądne pytanie: ***skąd mamy wiedzieć, że dany model dobrze się sprawuje wobec ostatecznego zestawu testowego oraz rzeczywistych danych, skoro nie wykorzystujemy danych testowych na etapie doboru systemu, lecz trzymamy je do momentu ewaluacji ostatecznego modelu?*** W celu rozwiązania problemu zdefiniowanego w tym pytaniu można wykorzystać różnorodne techniki sprawdzianu krzyżowego (ang. *cross-validation*), w których zestaw uczący zostaje podzielony na podzbiory uczące i **testowe** (**walidacyjne**), służące do oszacowania **wydajności generalizacji** modelu. Nie możemy również oczekiwać, że domyślne parametry różnych algorytmów uczenia maszynowego znajdujących się w bibliotekach programowych będą od razu zoptymalizowane pod kątem rozwiązania Twojego określonego problemu. Z tego powodu w dalszych rozdziałach książki będziemy często używać **technik optymalizacji** hiperparametrycznej, które pomogą nam poprawić skuteczność modelu. Hiperparametry to parametry, których nie uzyskano z danych, lecz które stanowią elementy regulacyjne modelu, wykorzystywane do poprawienia jego przewidywań — pojęcie to stanie się znacznie bardziej zrozumiałe w dalszej części książki, gdy przejdziemy do praktycznych przykładów.

[1] W wolnym tłumaczeniu: twierdzenie o nieistnieniu darmowych obiadów — *przyp. tłum.*

Ewaluacja modeli i przewidywanie wystąpienia nieznanych danych

Po wybraniu modelu dopasowanego do zestawu danych uczących możemy wykorzystać zbiór danych testowych do oszacowania skuteczności algorytmu wobec nieznanych danych, dzięki czemu będziemy w stanie określić błąd generalizacji. Jeżeli będziemy zadowoleni z jego skuteczności, możemy zacząć używać modelu do przewidywania nowych, przyszłych danych. Należy pamiętać o tym, że parametry wspomnianych wcześniej procedur, takich jak skalowanie cech oraz redukowanie wymiarowości, są określane wyłącznie na podstawie danych uczących, po czym wykorzystywane do przekształcania zbioru testowego oraz wszelkich nowych próbek — skuteczność mierzona jedynie na podstawie wyników z danych testowych może być nazbyt optymistyczna.

Wykorzystywanie środowiska Python do uczenia maszynowego

Python jest jednym z najpopularniejszych języków programowania stosowanych w analizie danych; dzięki temu zawiera olbrzymią bazę dodatkowych bibliotek stworzonych przez wspaniałą społeczność.

Mimo że wydajność języków interpretowanych (do których zalicza się Python) jest w przypadku zadań wymagających dużej mocy obliczeniowej niższa od wydajności języków niższego poziomu, istnieją biblioteki rozszerzeń, takie jak *NumPy* czy *SciPy*, które bazują na implementacjach języków Fortran i C, dzięki czemu uzyskujemy dostęp do szybkich i wektoryzowanych operacji na wielowymiarowych tablicach.

Będziemy najczęściej korzystać z biblioteki *scikit-learn*, która obecnie stanowi jedną z najpopularniejszych i najbardziej przystępnych darmowych bibliotek uczenia maszynowego.

Instalacja pakietów w Pythonie

Omawiane środowisko programistyczne jest dostępne na wszystkie główne systemy operacyjne — Microsoft Windows, Mac OS X i Linuksa — a zarówno jego instalator, jak i dokumentację znajdziesz na oficjalnej stronie Pythona: *https://www.python.org/*.

Ta książka została napisana pod kątem Pythona w wersji co najmniej 3.4.3, natomiast zalecam korzystanie z najbardziej aktualnej implementacji tego środowiska (w wersji 3), chociaż większość przykładów kodu powinna być również kompatybilna z wersją >= 2.7.10. Jeżeli postanowisz uruchamiać zawarte w książce przykłady kodu w wersji 2.7 Pythona, zapoznaj się najpierw z głównymi różnicami pomiędzy obydwiema wersjami środowiska programowania.

Dobre podsumowanie różnic pomiędzy wersjami 2.7 i 3.4 Pythona (w języku angielskim) znajdziesz pod adresem *https://wiki.python.org/moin/Python2orPython3*.

Dodatkowe, wykorzystywane w dalszej części książki pakiety można zainstalować za pomocą aplikacji *pip*, stanowiącej część standardowej biblioteki Pythona od wersji 3.3. Więcej informacji (w języku angielskim) na temat instalatora *pip* znajdziesz pod adresem *https://docs.python.org/3/installing/index.html*.

Po zainstalowaniu środowiska Python dodajemy kolejne pakiety, wpisując następującą komendę w wierszu poleceń:

```
pip install JakiśPakiet
```

Zainstalowane pakiety możemy zaktualizować za pomocą flagi `--upgrade`:

```
pip install JakiśPakiet --upgrade
```

Bardzo polecaną, alternatywną dystrybucją Pythona przeznaczoną do obliczeń naukowych jest Anaconda stworzona przez firmę Continuum Analytics. Jest to bezpłatna dystrybucja — również w przypadku zastosowań komercyjnych — zawierająca wszystkie niezbędne pakiety wykorzystywane w analizie danych, obliczeniach matematycznych oraz inżynierii, dostępne w przyjaznej, międzyplatformowej postaci. Instalator Anaconda znajdziesz pod adresem *https://www.continuum.io/downloads#py34*, z kolei szybkie wprowadzenie do tego środowiska jest dostępne na stronie *https://conda.io/docs/using/cheatsheet.html*.

Po zainstalowaniu Anacondy możemy instalować nowe pakiety Pythona za pomocą następującego polecenia:

```
conda install JakiśPakiet
```

Zainstalowane pakiety aktualizujemy, korzystając z poniższej komendy:

```
conda update JakiśPakiet
```

Przez większość czasu będziemy korzystać z wielowymiarowych tablic biblioteki *NumPy* do przechowywania i przetwarzania danych. Sporadycznie zastosujemy również bibliotekę *pandas* — nakładkę biblioteki NumPy zapewniającą dodatkowe, zaawansowane narzędzia do manipulowania danymi, dzięki czemu praca z tabelarycznymi informacjami będzie jeszcze wygodniejsza. Aby usprawnić proces nauki i zwizualizować dane ilościowe (pozwala to w maksymalnie intuicyjny sposób zrozumieć wykonywane działania), wprowadzimy również do użytku wysoce konfigurowalną bibliotekę *matplotlib*.

Poniżej wymieniam numery wersji głównych pakietów Pythona, które były wykorzystywane w trakcie pisania niniejszej książki. Upewnij się, że masz na swoim komputerze zainstalowane przynajmniej te wersje (lub nowsze), dzięki czemu przykładowy kod będzie działał we właściwy sposób:

- NumPy 1.9.1
- SciPy 0.14.0

- scikit-learn 0.15.2
- matplotlib 1.4.0
- pandas 0.15.2

Podsumowanie

W tym rozdziale zapoznaliśmy się bardzo ogólnie z uczeniem maszynowym i zaznajomiliśmy się z podstawowymi koncepcjami, którym poświęcimy znacznie większą uwagę w kolejnych rozdziałach.

Dowiedzieliśmy się, że na uczenie nadzorowane składają się dwie ważne dziedziny: klasyfikacja i regresja. Modele klasyfikacji pozwalają nam kategoryzować obiekty do znanych klas, natomiast dzięki analizie regresji jesteśmy w stanie prognozować wyniki ciągłe docelowych zmiennych. Uczenie nienadzorowanie nie tylko zapewnia dostęp do przydatnych technik odkrywających struktury nieoznakowanych danych, lecz również pozwala na kompresowanie danych w czasie wstępnego przetwarzania cech.

Przyjrzeliśmy się pobieżnie typowej strategii dopasowywania uczenia maszynowego do zadań problemowych, która stanowi dla nas podstawę do głębszych przemyśleń oraz ukazywania przykładów w dalszej części książki. Na koniec zaś przygotowaliśmy środowisko Pythona i zainstalowaliśmy oraz zaktualizowaliśmy wszystkie pakiety niezbędne do własnoręcznego testowania uczenia maszynowego.

W następnym rozdziale zaimplementujemy jeden z najwcześniejszych algorytmów uczenia maszynowego stosowanych w klasyfikacji, co przygotuje nas do rozdziału 3., „Stosowanie klasyfikatorów uczenia maszynowego za pomocą biblioteki scikit-learn", w którym zapoznamy się z bardziej zaawansowanymi algorytmami dostępnymi w bibliotece o jawnym kodzie źródłowym — scikit-learn. Algorytmy uczenia maszynowego uczą się z danych, dlatego kluczową kwestią jest dostarczanie im użytecznych informacji, zatem w rozdziale 4., „Tworzenie dobrych zbiorów uczących — wstępne przetwarzanie danych", przyjrzymy się istotnym technikom wstępnego przetwarzania danych. W rozdziale 5., „Kompresja danych poprzez redukcję wymiarowości", poznamy metody redukowania wymiarowości, które pomogą nam skompresować zbiór danych do mniejszej przestrzeni cech, co może być korzystne dla szybkości obliczeń. Ważnym aspektem tworzenia modeli uczenia maszynowego jest ocena ich skuteczności i oszacowanie ich zdolności predykcji dla nowych, nieznanych danych. W rozdziale 6., „Najlepsze metody oceny modelu i strojenie parametryczne", nauczymy się wykorzystywać najlepsze rozwiązania umożliwiające strojenie modelu i jego ewaluację. W pewnych przypadkach możemy być niezadowoleni ze skuteczności modelu predykcyjnego pomimo wielu godzin spędzonych na jego dopasowywaniu i testowaniu. Z rozdziału 7., „Łączenie różnych modeli w celu uczenia zespołowego", dowiemy się, w jaki sposób łączyć różne modele uczenia maszynowego, aby uzyskiwać jeszcze potężniejsze systemy prognozujące.

Po omówieniu wszystkich najważniejszych elementów typowego systemu uczenia maszynowego zaimplementujemy model przewidywania emocji na podstawie rozdziału 8., „Wykorzystywanie uczenia maszynowego w analizie sentymentów". Natomiast w rozdziale 9., „Wdrażanie modelu uczenia maszynowego do aplikacji sieciowej", podłączymy ten model pod aplikację sieciową i podzielimy się nim z resztą świata. Następnie w rozdziale 10., „Przewidywanie ciągłych zmiennych docelowych za pomocą analizy regresywnej", wykorzystamy algorytmy uczenia maszynowego do analizy regresywnej pozwalającej na prognozowanie ciągłych zmiennych wyjściowych, a w rozdziale 11., „Praca z nieoznakowanymi danymi — analiza skupień", wprowadzimy algorytmy skupień wyszukujące ukryte struktury wśród danych. Ostatnie dwa rozdziały zostały poświęcone sztucznym sieciom neuronowym służącym do rozwiązywania złożonych problemów, takich jak rozpoznawanie mowy i tekstu — czyli jednym z najbardziej fascynujących zagadnień w świecie uczenia maszynowego.

Python
Uczenie maszynowe

Uczenie maszynowe, zajmujące się algorytmami analizującymi dane, stanowi chyba najciekawszą dziedzinę informatyki. W czasach, kiedy generuje się olbrzymie ilości danych, samouczące się algorytmy maszynowe stanowią wyjątkową metodę przekształcania tych danych w wiedzę. W ten sposób powstało wiele innowacyjnych technologii, a możliwości uczenia maszynowego są coraz większe. Nieocenioną pomoc w rozwijaniu tej dziedziny stanowią liczne nowe biblioteki open source, które pozwalają na budowanie algorytmów w języku Python, będącym ulubionym, potężnym i przystępnym narzędziem naukowców i analityków danych.

Niniejsza książka jest lekturą obowiązkową dla każdego, kto chce rozwinąć swoją wiedzę o danych naukowych i zamierza w tym celu wykorzystać język Python. Przystępnie opisano tu teoretyczne podstawy dziedziny i przedstawiono wyczerpujące informacje o działaniu algorytmów uczenia maszynowego, sposobach ich wykorzystania oraz metodach unikania poważnych błędów. Zaprezentowano również biblioteki Theano i Keras, sposoby przewidywania wyników docelowych za pomocą analizy regresywnej oraz techniki wykrywania ukrytych wzorców metodą analizy skupień. Nie zabrakło opisu technik przetwarzania wstępnego i zasad oceny modeli uczenia maszynowego.

Uczenie maszynowe
— odkryj wiedzę, którą niosą dane!

W książce znajdziesz:

- podstawowe rodzaje uczenia maszynowego i ich zastosowanie
- bibliotekę scikit-learn i klasyfikatory uczenia maszynowego
- sposoby wydajnego łączenia różnych algorytmów uczących
- opis analizy sentymentów — przewidywanie opinii osób na podstawie sposobu pisania

Sebastian Raschka jest ekspertem w dziedzinie analizy danych i uczenia maszynowego. Obecnie przygotowuje doktorat na Michigan State University z metod obliczeniowych w biologii statystycznej. Biegle posługuje się Pythonem. Bierze również udział w różnych projektach open source i wdraża nowe metody uczenia maszynowego. W wolnym czasie pracuje nad modelami predykcyjnymi dyscyplin sportowych. Jeżeli nie siedzi przed monitorem, chętnie uprawia sport.

[PACKT] open source
PUBLISHING
community experience distilled

Helion

księgarnia internetowa
http://helion.pl
zamówienia telefoniczne
0 801 339900
0 601 339900

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Informatyka w najlepszym wydaniu

sięgnij po WIĘCEJ
KOD KORZYŚCI

ISBN 978-83-283-3613-1
9 788328 336131
cena: 69,00 zł