TECHNOLOGY IN ACTION™

Uczenie maszynowe na Raspberry Pi

Eksperymentowanie z danymi i rozpoznawaniem obrazów

Donald J. Norris

Apress®

Apress®

Donald J. Norris

Uczenie maszynowe na Raspberry Pi

Eksperymentowanie z danymi i rozpoznawaniem obrazów

Przekład: Maria Chaniewska

APN Promise, Warszawa 2020

Uczenie maszynowe na Raspberry Pi. Eksperymentowanie z danymi i rozpoznawaniem obrazów

First published in English under the title
Machine Learning with the Raspberry Pi: Experiments with Data and Computer Vision
by Donald J. Norris

Copyright © 2020 by Donald J. Norris

This edition has been translated and published under licence from APress Media, LLC, part of Springer Nature.

APress Media, LLC, part of Springer Nature takes no responsibility and shall not be made liable for the accuracy of the translation.

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 publisher.

Polish language edition published by APN PROMISE S.A., Copyright © 2020

Autoryzowany przekład z wydania w języku angielskim, zatytułowanego: PMachine Learning with the Raspberry Pi: Experiments with Data and Computer Vision by Donald J. Norris, opublikowanego przez APress Media, LLC, oddział Springer Nature.

Wszystkie prawa zastrzeżone. Żadna część niniejszej książki nie może być powielana ani rozpowszechniana w jakiejkolwiek formie i w jakikolwiek sposób (elektroniczny, mechaniczny), włącznie z fotokopiowaniem, nagrywaniem na taśmy lub przy użyciu innych systemów bez pisemnej zgody wydawcy.

APN PROMISE SA, ul. Domaniewska 44a, 02-672 Warszawa
tel. +48 22 35 51 600, fax +48 22 35 51 699
e-mail: mspress@promise.pl

Książka ta przedstawia poglądy i opinie autorów. Przykłady firm, produktów, osób i wydarzeń opisane w niniejszej książce są fikcyjne i nie odnoszą się do żadnych konkretnych firm, produktów, osób i wydarzeń, chyba że zostanie jednoznacznie stwierdzone, że jest inaczej. Ewentualne podobieństwo do jakiejkolwiek rzeczywistej firmy, organizacji, produktu, nazwy domeny, adresu poczty elektronicznej, logo, osoby, miejsca lub zdarzenia jest przypadkowe i niezamierzone.

Wszystkie znaki towarowe występujące w książce mogą być własnością ich odnośnych właścicieli.

APN PROMISE SA dołożyła wszelkich starań, aby zapewnić najwyższą jakość tej publikacji. Jednakże nikomu nie udziela się rękojmi ani gwarancji.
APN PROMISE SA nie jest w żadnym wypadku odpowiedzialna za jakiekolwiek szkody będące następstwem korzystania z informacji zawartych w niniejszej publikacji, nawet jeśli APN PROMISE została powiadomiona o możliwości wystąpienia szkód.

ISBN: 978-83-7541-418-9 (druk), 978-83-7541-421-9 (ebook)

Przekład: Maria Chaniewska
Redakcja: Marek Włodarz
Korekta: Ewa Swędrowska
Skład i łamanie: MAWart Marek Włodarz

Spis treści

O autorze

Donald J. Norris jest zapalonym hobbystą i majsterkowiczem. Jest także inżynierem elektroniki ze stopniem naukowym z zarządzania produkcją. Don jest emerytowanym pracownikiem rządowej służby cywilnej w Marynarce Wojennej USA, gdzie specjalizował się w akustyce i cyfrowym przetwarzaniu sygnałów. Ma także ponad dwanaście lat doświadczenia jako profesjonalny programista używający języków C, C#, C++, Python i Java oraz 5 lat doświadczenia jako certyfikowany konsultant zabezpieczeń IT.

O recenzencie technicznym

Ahmed Fawzy Gad jest inżynierem zajmującym się uczeniem maszynowym posiadającym stopień licencjata i magistra technologii informacyjnej. Ahmed jest asystentem dydaktycznym i naukowcem, który interesuje się uczeniem maszynowym/głębokim, komputerowym rozpoznawaniem obrazów oraz Pythonem. Jest recenzentem technicznym uczenia maszynowego i konsultantem, który pomaga innym osobom w realizacji projektów. Ahmed współtworzy pisemne samouczki i artykuły na wielu blogach w tym Paperspace, Real Python, KDnuggets, Heartbeat i Towards Data Science.

Ahmed jest autorem trzech książek: *TensorFlow: A Guide to Build Artificial Neural Networks Using Python* (TensorFlow: Przewodnik budowania sztucznych sieci neuronowych w języku Python), wydawnictwo Lambert, 2017 r., *Practical Computer Vision Applications Using Deep Learning with CNNs* (Praktyczne aplikacje komputerowego rozpoznawania obrazów przy użyciu uczenia głębokiego w sieciach CNN), wydawnictwo Apress, 2018 r. oraz *Building Android Apps in Python Using Kivy with Android Studio* (Budowanie aplikacji Android w języku Python przy użyciu Kivy i Android Studio), wydawnictwo Apress, 2019 r.

Zachęca do kontaktów przez LinkedIn (linkedin.com/in/AhmedFGad), Facebook (fb.com/AhmedFGadd) i e-mail (ahmed.f.gad@gmail.com).

Przykłady kodu

Wszystkie pliki skryptów przedstawione w książce dostępne są w witrynie wydawcy pod adresem:

`https://ksiazki.promise.pl/produkt/uczenie-maszynowe-na-raspberry-pi`

Ponieważ książka drukowana jest w wersji czarno-białej, udostępniliśmy tam również plik PDF zawierający kolorowe wersje tych ilustracji, których czytelność bez koloru jest niewystarczajaca.

Rozdział 1

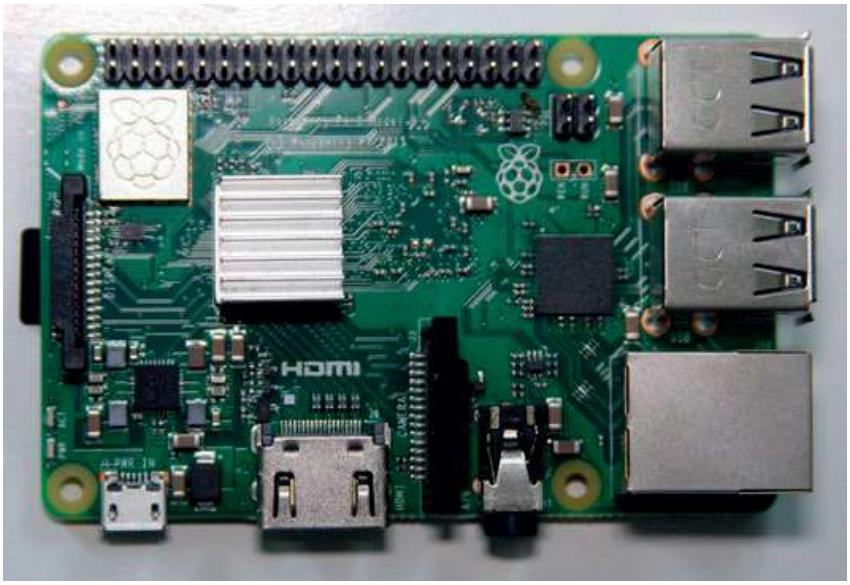
Wprowadzenie do uczenia maszynowego (ML) na Raspberry Pi (RasPi)

Ten rozdział zawiera wprowadzenie do platformy RasPi i uczenia maszynowego. Na początek zostanie opisana platforma sprzętowa, RasPi, na której zostaną przeprowadzone wszystkie demonstracje w tej książce. Później nastąpi wprowadzenie do koncepcji uczenia maszynowego oraz wyjaśnienie, dlaczego jest tak ekscytującą i szybko rozwijającą się dziedziną nauki.

Wprowadzenie do RasPi

Do przeprowadzenia demonstracji z tej książki będzie potrzebna płytka RasPi. W kilku następnych podrozdziałach pokażę, jak skonfigurować model RasPi 3 B lub B+ jako stację roboczą, na której będą wykonywane skrypty i programy wymagane w różnych demonstracjach uczenia maszynowego. Rysunek 1-1 przedstawia model RasPi 3 B+, który jest używany w tej książce.

Modele RasPi 3 B i B+ różnią się kilkoma szczegółami. Zasadniczo są podobne, ale w porównaniu z modelem B model B+ ma trochę większą prędkość taktowania procesora oraz kilka poprawek funkcji bezprzewodowych. Żadne z tych ulepszeń nie wpływa istotnie na działanie projektów z tej książki, jeśli użyjesz modelu B zamiast B+.



Rysunek 1-1. *Model Raspberry Pi 3 B+*

Nie będę omawiać budowy jednopłytkowego komputera RasPi, ponieważ zostało to już wyczerpująco opisane w wielu dostępnych książkach i blogach. Jak wspomniałem wcześniej, używałem modelu RasPi 3 B+ w konfiguracji stacji roboczej. W tej konfiguracji do płytki RasPi podłączono klawiaturę USB, mysz USB i monitor HDMI. W mojej konfiguracji płytka RasPi jest zasilana zasilaczem 2,2 A, 5 V ze złączem micro USB.

RasPi nie używa mechanicznego dysku twardego do implementacji systemu plików zawierającego system operacyjny (OS). Wszystkie dotychczasowe wersje RasPi wykorzystują podłączaną kartę micro SD, która służy za pamięć masową. Chociaż jest możliwe podłączenie tradycyjnego dysku twardego do RasPi, służyłby on tylko jako zapasowe urządzenie pamięci, a nie jako główna pamięć na system operacyjny ani partycja rozruchowa. Dalej pokażę, jak pobrać i zainstalować system operacyjny na karcie micro SD, aby umożliwić działanie RasPi jako funkcjonalnego mikrokontrolera ML.

Niewątpliwie najprostszym sposobem rozpoczęcia pracy jest zakup wstępnie zaprogramowanej karty micro SD. Takie karty są gotowe do użytku i wymagają tylko skonfigurowania, aby odpowiadały konfiguracji konkretnej stacji roboczej, w tym sieci WiFi. Proces konfiguracji WiFi zostanie opisany dalej w tym podrozdziale, ale najpierw chcę opisać sposób tworzenia własnej karty micro SD w razie potrzeby.

Oprogramowanie karty micro SD do załadowania jest nazywane obrazem systemu Raspbian (Raspbian Image) i jest dostępne bezpłatnie z wielu witryn internetowych, w tym zalecanej witryny fundacji Raspberry Pi pod adresem `raspberrypi.org`.

Najnowszy obraz systemu operacyjnego jest zawsze dostępny z sekcji Downloads tej witryny. Istnieją dwa typy obrazu systemu operacyjnego do pobrania. Pierwszy typ ma nazwę NOOBS, która jest skrótem od „New Out of the Box Software". Dostępne są dwie wersje obrazu NOOBS. Jedna nazywa się NOOBS, a druga NOOBS Lite. W czasie pisania tej książki obie wersje były rozpoznawane jako v3.0.0. NOOBS zawiera łatwy instalator systemu operacyjnego, który zawiera Raspbian OS, a także inny popularny system operacyjny LibreELEC. Ponadto wersja NOOBS umożliwia wybór alternatywnych systemów operacyjnych, które są następnie pobierane z Internetu i instalowane. NOOBS Lite zawiera ten sam instalator systemu operacyjnego, ale bez wstępnie załadowanego systemu Raspbian i bez opcji LibreELEC. Jednak ta wersja zapewnia to samo menu wyboru systemu operacyjnego, które pozwala na pobranie i zainstalowanie obrazu systemu Raspbian i innych systemów operacyjnych.

Obrazy NOOBS i NOOBS Lite są tylko zbiorami plików i podkatalogów, które można pobrać przy użyciu aplikacji BitTorrent lub po prostu jako surowy plik Zip. Pliki pobierane z BitTorrent i Zip mają rozmiar około 1,2 GB. Wyodrębniony obraz ma rozmiar 1,36 GB, ale ostateczny rozmiar po zainstalowaniu przekracza 4 GB. Oznacza to, że na ostateczny obraz systemu będzie potrzebna karta micro SD o rozmiarze co najmniej 8 GB. Jednak w celu replikacji wszystkich demonstracji ML z tej książki stanowczo polecam kartę micro SD o rozmiarze co najmniej 16 GB, klasy 10, aby zapewnić mnóstwo dostępnej pamięci, a także maksymalizować przepustowość danych działającego komputera RasPi.

Drugi typ obrazu pozwala na bezpośrednie pobranie systemu operacyjnego. Obecnie dostępnym obrazem jest dystrybucja systemu Raspbian Linux o nazwie kodowej Stretch. Ta wersja systemu Raspbian może być pobrana przy użyciu BitTorrent lub jako plik Zip z końcowym obrazem o rozmiarze podobnym do obrazu NOOBS.

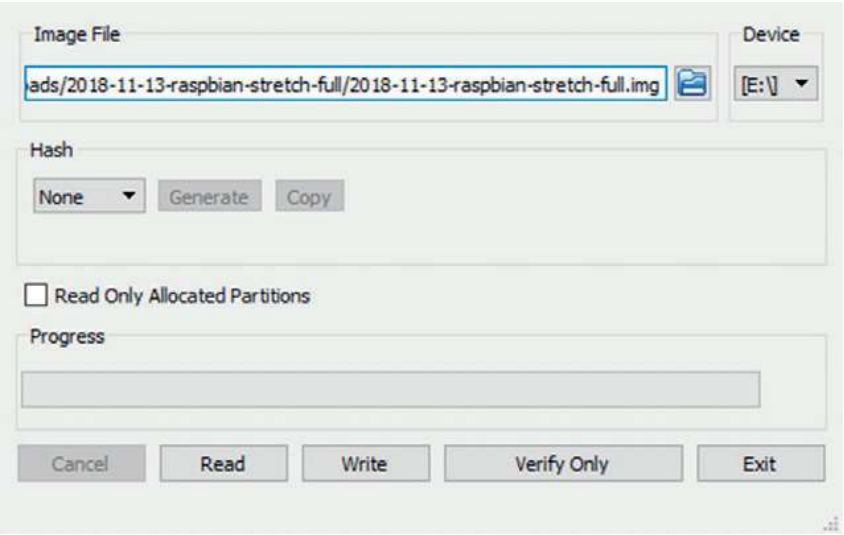
Karta micro SD musi być skonfigurowana po pobraniu obrazu. Omówię tylko bezpośredni typ pobierania systemu Raspbian, ponieważ wierzę, że większość Czytelników tej książki ma wystarczające doświadczenie z podstawowymi operacjami komputerowymi, a także z płytką RasPi, aby zdecydować się na podejście z bezpośrednim pobieraniem.

Zapisywanie obrazu systemu Raspbian na kartę micro SD

Karta micro SD nie wymaga formatowania przed zapisaniem obrazu. Ta część procesu jest wykonywana automatycznie przez aplikację, która zapisuje obraz na kartę. Wystarczy tylko skonfigurować odpowiednią aplikację na wykorzystywanym komputerze. W przypadku komputera z systemem Windows stanowczo zalecam użycie programu Win32DiskImager, dostępnego pod adresem

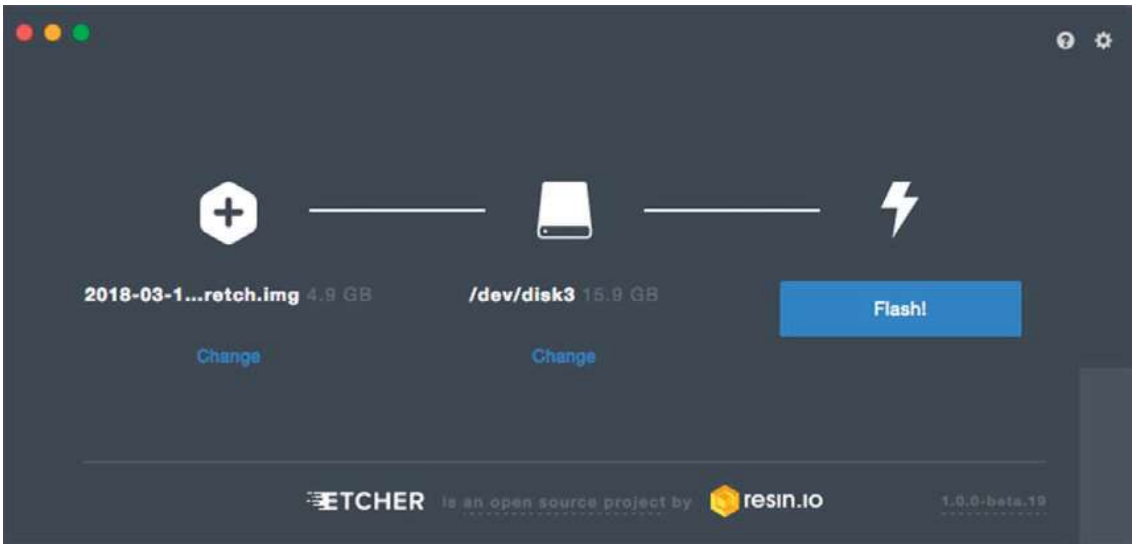
```
https://sourceforge.net/projects/win32diskimager/files/latest/download
```

Pobrany plik to archiwum Zip, które trzeba rozpakować przed użyciem. Następnie wystarczy uruchomić aplikację, wybrać lokalizację obrazu dysku, a także wybrać literę pliku logicznego karty micro SD. Rysunek 1-2 przedstawia mój ekran konfiguracji użytej do zapisania wersji Raspbian Stretch na karcie micro SD na komputerze z systemem Windows.



Rysunek 1-2. *Zrzut ekranu programu Win32DiskImager*

Użytkownikom systemu Mac polecam użycie programu Etcher do zapisu obrazu dysku. Jest on dostępny pod adresem `https://etcher.io/`. Ta aplikacja działa podobnie do programu Win32DiskImager. Rysunek 1-3 przedstawia zrzut ekranu tego programu działającego na moim komputerze MacBook Pro.



Rysunek 1-3. *Zrzut ekranu programu Etcher*

Po zapisaniu obrazu systemu operacyjnego na karcie micro SD trzeba go skonfigurować. Podzieliłem proces konfiguracji na dwa etapy. Pierwszy skupia się na tym, co uznałem za konfigurację obowiązkową tym sensie, że jej przeprowadzenie jest konieczne, aby system operacyjny działał zgodnie z oczekiwaniami w danej sytuacji. Drugi etap konfiguracji to „strojenie" już ogólnie skonfigurowanego systemu operacyjnego w celu dostosowania do konkretnych potrzeb.

Uwaga Proces konfiguracji RasPi jest dynamiczny i stale ewoluuje. Niniejszym przyznaję, że poniższe instrukcje, chociaż zasadne w czasie pisania tej książki, mogą być nieaktualne, kiedy będziesz próbować je wykonać. Wynika to po prostu z natury oprogramowania open source. Jednak jestem przekonany, że bez względu na zastosowane procedury, będą przejrzyste i proste do przeprowadzenia.

Obowiązkowa konfiguracja

Rysunek 1-4 przedstawia pierwszy ekran po rozruchu RasPi.

Rysunek 1-4. *Początkowy zrzut ekranu konfiguracji*

Trzeba kliknąć przycisk Next (Dalej), aby rozpocząć proces konfiguracji, zupełnie tak, jak na rysunku. Natychmiast pojawi się ekran z rysunku 1-5 pokazujący domyślny kraj, język i strefę czasową.

Rysunek 1-5. *Domyślne okno dialogowe* ***Set Country*** *(Ustaw kraj)*

Ważne jest, aby przynajmniej wybrać odpowiedni kraj i język, ponieważ w przeciwnym razie pojawią się trudności z wprowadzaniem skryptów lub programów z powodu

konfliktów między konfiguracją fizycznej klawiatury a wprowadzanymi znakami. Menu strefy czasowej również zostanie automatycznie dostosowane, aby odzwierciedlić strefy czasowe dostępne w wybranym kraju.

Rysunek 1-6 przedstawia to pole po dokonaniu przeze mnie odpowiednich wyborów.

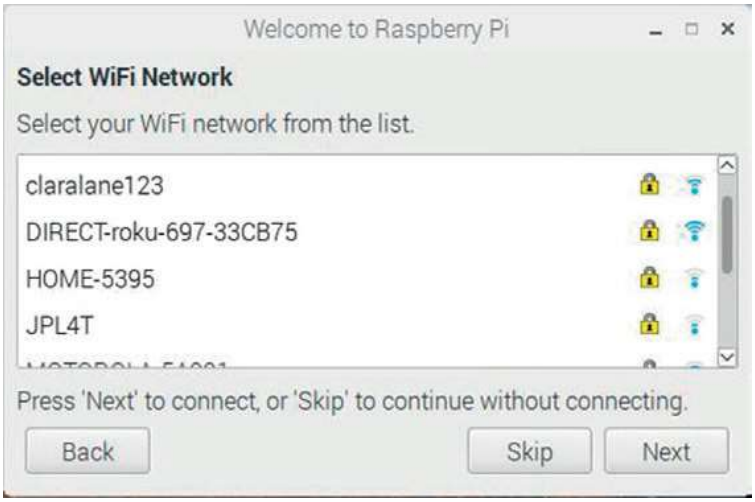
Rysunek 1-6. *Dostosowane okno dialogowe* ***Set Country***

Kliknięcie przycisku Next przywołuje okno dialogowe **Change Password** (Zmień hasło) pokazane na rysunku 1-7.

Rysunek 1-7. *Okno dialogowe* ***Change Password***

Zmiana domyślnego hasła *raspberry*, które jest prawdopodobnie powszechnie znane, powinno poprawić bezpieczeństwo systemu. Ten wybór należy całkowicie do Ciebie

i nie ma wpływu na przebieg jakichkolwiek demonstracji z tej książki. Koniecznie zapamiętaj hasło. W przeciwnym razie będzie wymagana instalacja obrazu od nowa. Nie wydaje mi się, aby był jakiś łatwy sposób odzyskania zapomnianego hasła do systemu operacyjnego Raspbian. Jeśli zdecydujesz się nie zmieniać hasła, po prostu kliknij przycisk Next, a powinno się pojawić okno dialogowe **Select WiFi Network** (Wybierz sieć WiFi). Rysunek 1-8 przedstawia to okno dialogowe, które pojawiło się u mnie po kliknięciu przycisku.

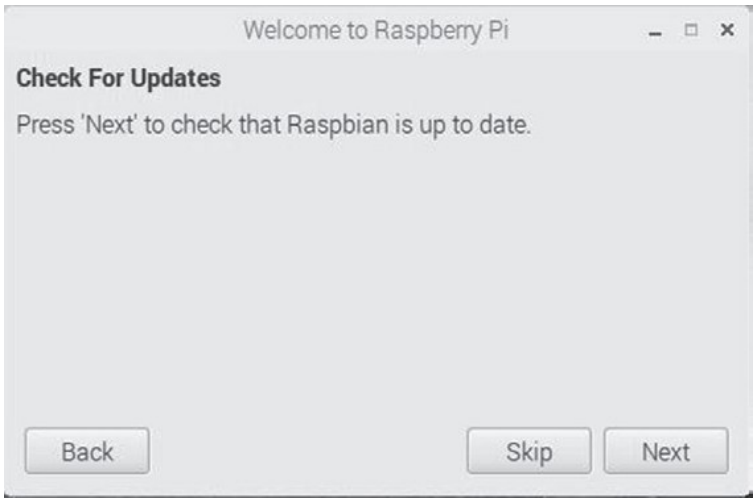


Rysunek 1-8. *Okno dialogowe **Select WiFi Network***

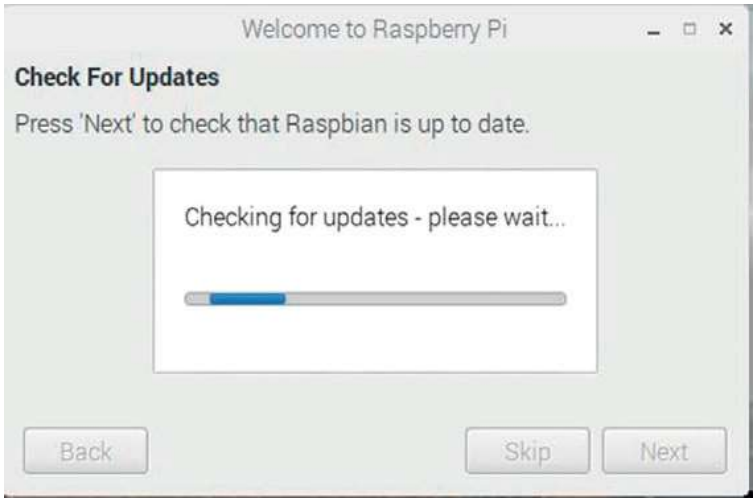
Trzeba kliknąć odpowiedni identyfikator SSID sieci WiFi w celu ustanowienia łącza komunikacyjnego WiFi. Pojawi się kolejne okno dialogowe z monitem o naciśnięcie przycisku na fizycznym routerze lub wprowadzenie hasła skojarzonego z wybranym identyfikatorem SSID sieci WiFi. Zdecydowałem się, aby nie pokazywać tego szczególnego okna dialogowego z oczywistych względów bezpieczeństwa. Kliknięcie przycisku Next spowoduje przywołanie okna dialogowego **Check For Updates** (Sprawdź aktualizacje), pokazanego na rysunku 1-9.

Nie da się sprawdzić aktualizacji bez wcześniejszego skonfigurowania sieci WiFi. W istocie nie jestem pewien, czy to okno dialogowe w ogóle się pojawi, jeśli nie ma działającej konfiguracji łącza WiFi. Przy założeniu, że faktycznie zostało skonfigurowane łącze WiFi, kliknięcie przycisku Next spowoduje, że RasPi połączy się z Internetem i sprawdzi status obecnie zainstalowanego oprogramowania zawartego w nowym obrazie. Jednak nie musisz uruchamiać sprawdzania w tym momencie konfiguracji, ponieważ wkrótce pokażę, jak dokonać aktualizacji przy użyciu polecenia okna terminala.

Wybór należy do Ciebie. W rzeczywistości żadna z tych opcji nie zaszkodzi, a jedynym kosztem jest dodatkowy czas procesu konfiguracji. Jeśli chcesz używać ręcznego procesu aktualizacji, po prostu kliknij przycisk Skip (Pomiń), a w przeciwnym razie kliknij przycisk Next. Rysunek 1-10 przedstawia zmianę wyglądu okna dialogowego **Check For Updates** po kliknięciu przycisku Next.



Rysunek 1-9. *Okno dialogowe* ***Check For Updates***



Rysunek 1-10. *Aktywne okno dialogowe* ***Check For Updates***

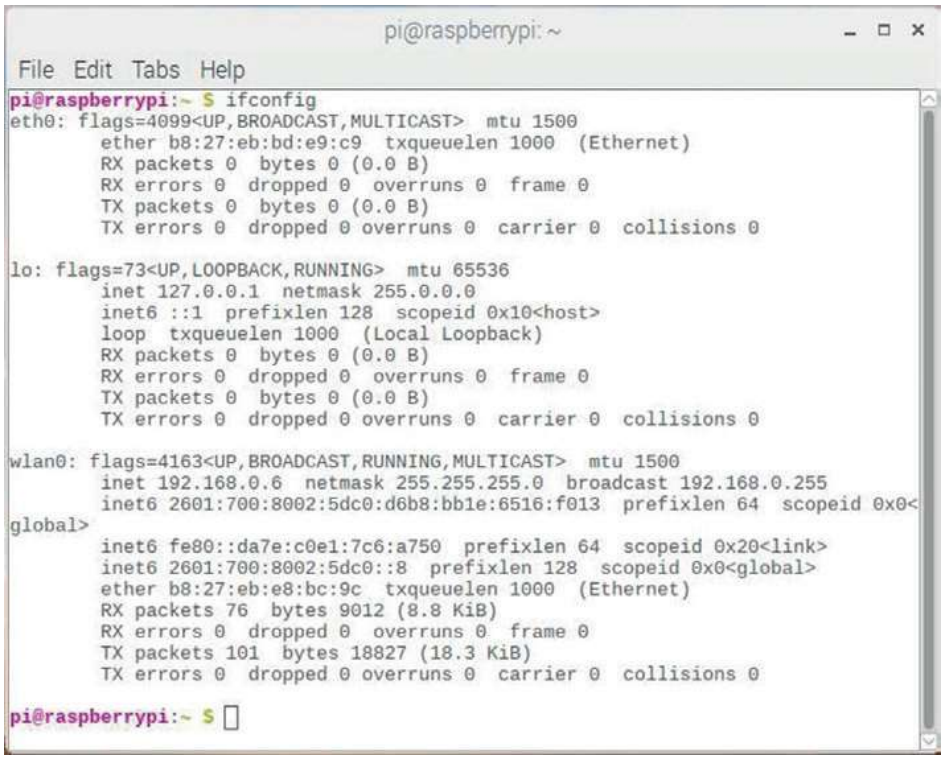
Pasek aktywności pozostanie aktywny przez kilka minut, w zależności od liczby wykrytych aktualizacji. Po ukończeniu aktualizacji pojawi się okno z informacją, że proces konfiguracji został prawie ukończony i potrzeba kliknąć przycisk Reboot (Uruchom

ponownie), aby ukończyć proces. Sugeruję, aby to zrobić i nie zapomnieć o wprowadzeniu nowego hasła, jeśli zostało zmienione.

Teraz polecam wprowadzenie następującego polecenia do okna terminala, aby sprawdzić stan połączenia WiFi:

```
Ifconfig
```

Rysunek 1-11 przedstawia wyniki tego polecenia w moim systemie RasPi.



Rysunek 1-11. *Ekran polecenia ifconfig*

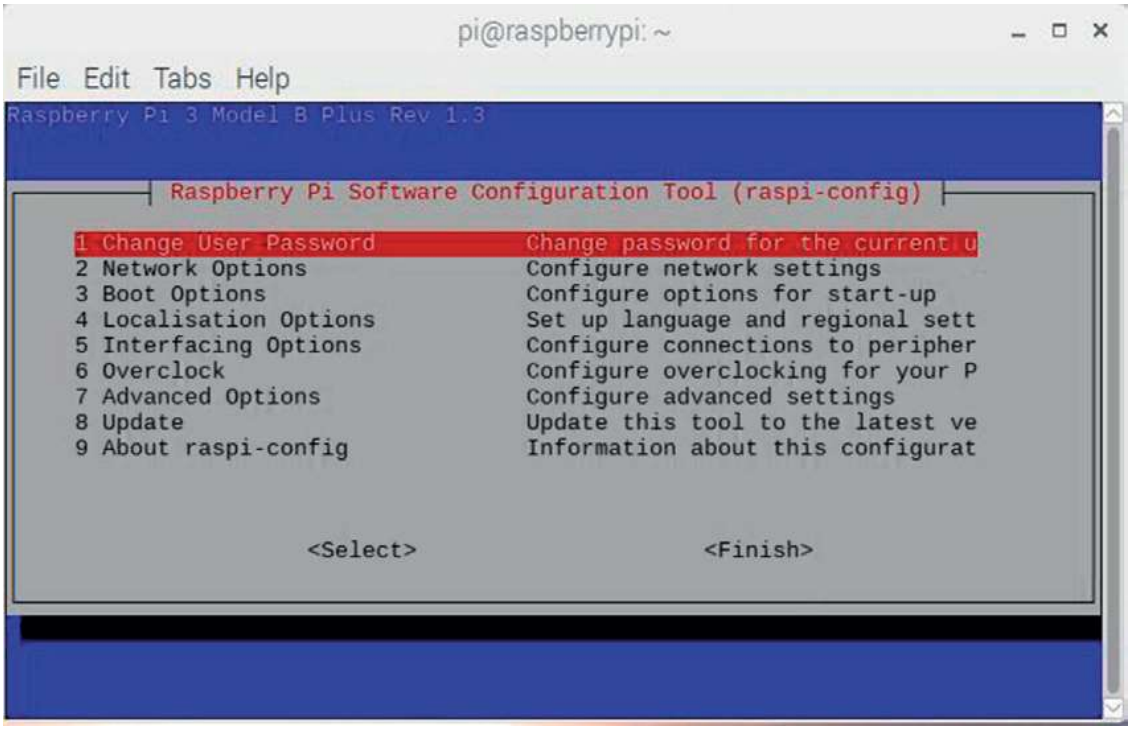
W sekcji wlan0 powinno być widoczne, że lokalny adres IP 192.168.0.6 został przypisany do RasPi przez domowy router WiFi. To przypisanie potwierdza, że RasPi jest w stanie połączyć się z Internetem. Sprawdź, czy router domowy ma skonfigurowany protokół DHCP, jeśli nie zobaczysz adresu IP podobnego to tego pokazanego na rysunku.

Opcjonalna konfiguracja

Proces konfiguracji opcjonalnej wykorzystuje narzędzie o nazwie raspi-config. To narzędzie jest dostarczane w początkowo pobranym obrazie. Aby uruchomić narzędzie raspi-config, można otworzyć okno terminala i wpisać następujące polecenie:

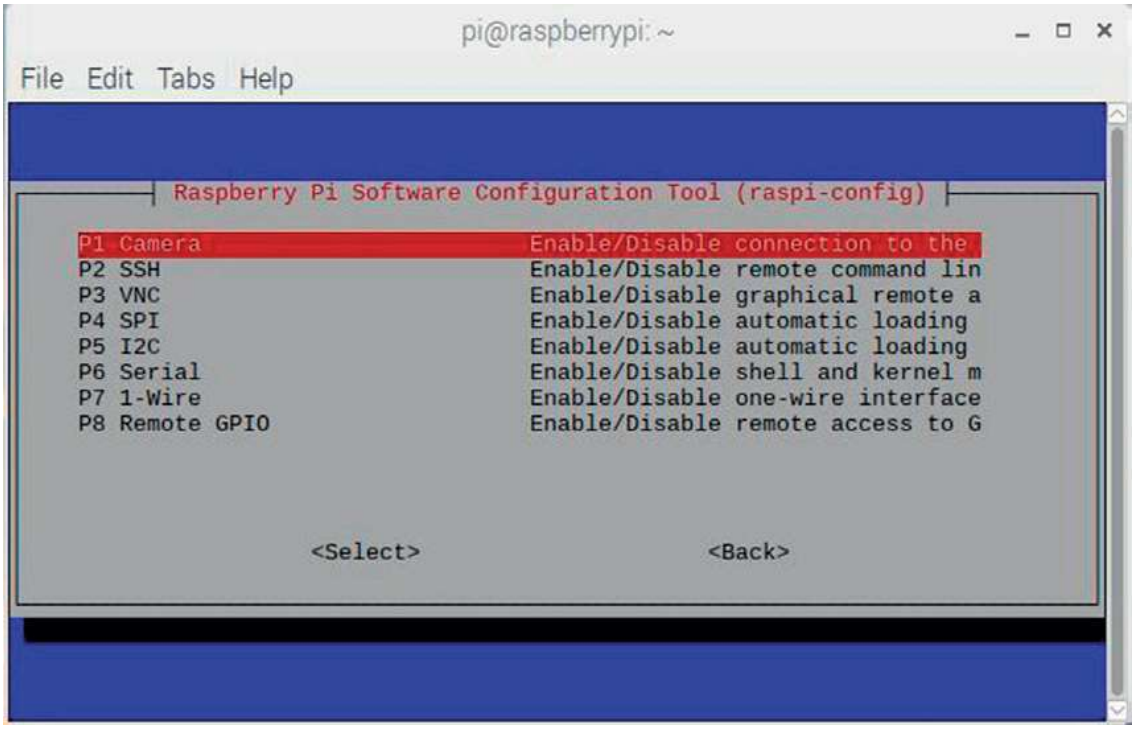
```
sudo raspi-config
```

Rysunek 1-12 przedstawia ekran startowy narzędzia raspi-config.



Rysunek 1-12. *Ekran startowy narzędzia raspi-config*

Wybranie z menu pozycji Interfacing Options (Opcje interfejsów) spowoduje pokazanie menu podrzędnego, jak na rysunku 1-13.



Rysunek 1-13. *Menu Interfacing Options*

To menu ma osiem opcji wyboru, jak widać na rysunku. Włączenie opcji będzie zależeć od typu urządzeń używanych w systemie RasPi. Zalecam włączenie następujących opcji w celu dopasowania konfiguracji do demonstracji i procedur opisanych w tej książce:

- Camera (kamera)
- SSH
- SPI
- I2C
- Serial (szeregowy)
- 1-Wire

Opcje interfejsów można łatwo dodawać lub usuwać w dowolnym momencie, wystarczy ponownie uruchomić narzędzie raspi-config. W każdym razie dodanie opcji interfejsu tylko minimalnie zwiększa rozmiar całego systemu operacyjnego. Zauważ także, że włączenie interfejsu powoduje tylko wywołanie skojarzonych sterowników dla tego konkretnego urządzenia.

Znów trzeba będzie ponownie uruchomić RasPi, aby wdrożyć te opcjonalne ustawienia konfiguracyjne. Wprowadź następujące polecenie w oknie terminala, aby ponownie uruchomić komputer:

```
sudo reboot
```

W tym momencie mamy pomyślnie zainstalowany i skonfigurowany system RasPi. Następnie trzeba zaktualizować i uaktualnić system, aby zapewnić, że jest zainstalowana najnowsza wersja systemu operacyjnego Raspbian.

Aktualizowanie i uaktualnianie dystrybucji systemu Raspbian

Dystrybucja Raspbian systemu Linux jest nieustannie poprawiana, jak wspomniałem wcześniej. Bardzo łatwo można zapewnić, że mamy najnowszą zaktualizowaną i uaktualnioną dystrybucję po ustanowieniu łączności internetowej. Wprowadź następujące polecenie w oknie terminala, aby zaktualizować zainstalowany system operacyjny:

```
sudo apt-get update
```

Akcja aktualizacji (update) zmienia listę pakietów wewnątrz systemu, aby pasowała do bieżącej listy pakietów online. W rzeczywistości nie zmienia ona żadnego z zainstalowanych pakietów, nawet jeśli są przestarzałe lub nieaktualne. Faktyczne zmiany są realizowane dopiero po wprowadzeniu w oknie terminala następującego polecenia:

```
sudo apt-get upgrade
```

Jeśli oryginalnie zainstalowana dystrybucja nie jest zbyt stara, aktualizacja (update) odbywa się dość szybko. Jednak uaktualnienie (upgrade) może zająć dość dużo czasu w przypadku obecności wielu nieaktualnych pakietów.

Wystarczy pamiętać, aby zawsze wykonywać najpierw akcję update, a potem upgrade. Wszystkie projekty w tej książce zostały utworzone przy użyciu zaktualizowanej i uaktualnionej dystrybucji Stretch Raspbian. Zauważyłem, że brak aktualizacji i uaktualnienia może czasami prowadzić do dziwnych błędów i awarii systemu, które są nieoczekiwane i kłopotliwe.

W tym momencie procesu instalacji i konfiguracji system RasPi powinien być już w pełni funkcjonalny. Nadszedł czas, aby wprowadzić koncepcję środowiska wirtualnego Pythona, przed przejściem do omówienia uczenia maszynowego.

Środowisko wirtualne Pythona

Ten podrozdział zawiera odpowiedzi na dwa pytania:

1. Co to jest środowisko wirtualne Pythona?
2. Dlaczego jest potrzebne?

Najpierw zajmę się drugim pytaniem. Działanie Pythona, jak wielu podobnych języków zorientowanych obiektowo, zależy od wielu bibliotek pomocniczych i procedur. W Pythonie te biblioteki są nazwane zależnościami i są przechowywane w jednym z dwóch katalogów w zależności od pochodzenia. Pochodzenie oznacza, że te biblioteki, które są uważane za istotne lub rdzenne dla jądra systemu Linux, są przechowywane w katalogu System-packages. Wszystkie pozostałe, chociaż mogą być niezwykle ważne do prawidłowego działania Pythona, są przechowywane w katalogu Site-packages. Za każdym razem, gdy jest wydawana nowa wersja języka Python, katalog System-packages jest aktualizowany i modyfikowany zgodnie z potrzebami najnowszej wersji. Dlatego jest tylko jedna wersja każdej z niezbędnych bibliotek systemowych, zapisanych w tym katalogu. Inaczej jest w przypadku katalogu Site-packages. Jest to spowodowane tym, że użytkownik zwykle instaluje żądane oprogramowanie i wszelkie biblioteki lub zależności wymagane dla tego oprogramowania. Jest całkiem możliwe posiadanie większej liczby wersji tej samej biblioteki (zależności) w katalogu Site-packages, po prostu z powodu wielu instalacji oprogramowania. Problem szybko narasta, ponieważ Linux instaluje zależności, bazując wyłącznie na ich nazwach, i lekceważy jakiekolwiek sprawdzanie wersji. Jest całkiem możliwe, że Projekt A wymaga biblioteki oprogramowania X w wersji 1, a Projekt B wymaga biblioteki oprogramowania X w wersji 2. Linux nie potrafi ujednoznacznić niespójności wersji, więc jeden lub oba projekty nie będą działać prawidłowo. Środowiska wirtualne Pythona zostały zaprojektowane w celu wyeliminowania tego problemu.

Głównym celem środowisk wirtualnych Pythona jest utworzenie izolowanego środowiska dla każdego projektu Pythona. To oznacza, że każdy projekt będzie miał własne zależności, bez względu na zależności wymagane przez inne projekty.

Utworzenie oddzielnych środowisk wirtualnych dla obu projektów A i B wyeliminowałoby problem z niespójnością wersji. Każde środowisko mogłoby zależeć od dowolnej wymaganej wersji oprogramowania X, niezależnie od jakichkolwiek innych projektów.

Jedną z zalet środowisk wirtualnych jest to, że nie ma ograniczenia liczby tworzonych środowisk, poza ograniczeniami wynikającymi z ilości pamięci fizycznej. Odpowiedź na pierwsze postawione wcześniej pytanie jest więc prosta. Środowiska wirtualne Pythona to po prostu hierarchiczny zbiór katalogów, zawierających trochę skryptów i łącz symbolicznych, nic więcej. Ich utworzenie nie jest związane z żadną czarną magią ani sztuką. Wierzę, że gdy zaczniesz z niech korzystać, już nie zrezygnujesz. Wielu deweloperów rutynowo ich używa, oszczędzając sobie wielu godzin frustracji i złości w trakcie prób rozwiązywania nieznanych błędów powodowanych nieumyślnymi problemami dotyczącymi zależności.

Instalacja środowiska wirtualnego Pythona

Przed wykonaniem tych instrukcji upewnij się, że Python 3 jest zainstalowany i działa prawidłowo. Ponadto upewnij się, że masz zaktualizowaną i uaktualnioną dystrybucję Raspbian Stretch systemu Linux, zgodnie z wcześniejszym opisem w tym rozdziale.

Ta procedura zawiera sześć kroków. Wykonaj je kolejno, aby pomyślnie utworzyć środowisko wirtualne Pythona, którego będziesz używać w pracy nad modelami danych:

1. Zainstaluj pip, czyli narządzie menedżera pakietów Pythona. To narzędzie jest bardzo podobne do zaawansowanego narzędzia pakietów (apt), ale używa oddzielnego repozytorium dystrybucji. Wprowadź następujące polecenia:

```
wget https://bootstrap.pypa.io/get-pip.py
sudo python3 get-pip.py
```

> **Uwaga** Podczas pisana tej książki najnowszą wersją narzędzia pip była 19.0.3.

2. Zainstaluj narzędzia `virtualenv` i `virtualenvwrapper`. Narzędzie `virtualenv` służy do tworzenia wirtualnego środowiska w Pythonie 3. Narzędzie `virtualenvwrapper`

tworzy łącza miedzy językiem Python a kodem Pythona do wykonania w środowisku. Wprowadź następujące polecenie:

```
sudo pip install virtualenv virtualenvwrapper
sudo rm -rf ~/get-pip.py ~/.cache/pip
```

3. Trzeba przeedytować ukryty plik o nazwie `.profile`, znajdujący się w katalogu domowym, aby zawierał pewne dane inicjowania. Zalecam użycie edytora `nano` i dopisanie danych, jak poniżej:

```
cd ~
sudo nano .profile
```

Dane do dopisania po ostatnim wierszu w istniejącym pliku:

```
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh
```

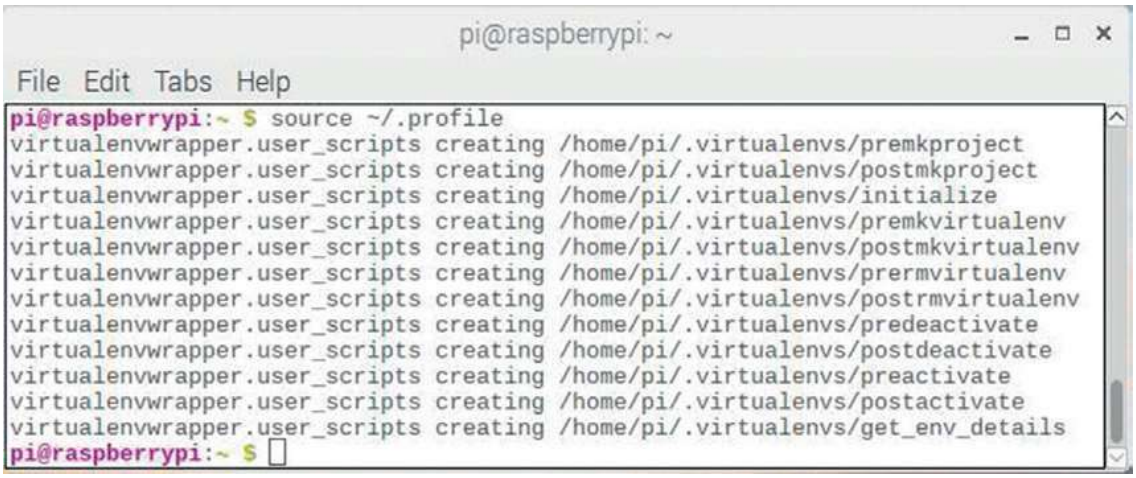
Zamiast tego możesz bezpośrednio wprowadzić dane inicjujące w wierszu polecenia przy użyciu następujących poleceń:

```
echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile
echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.profile
echo "export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3" >> ~/.profile
echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile
```

4. Teraz trzeba wykonać polecenie `source` na pliku `~./profile`. Polecenie `source` służy do załadowania funkcji zawartych w nazwanym pliku do bieżącej powłoki w celu wykonania.

```
source ~/.profile
```

Uwaga Po uruchomieniu powyższego polecenia po raz pierwszy powinien pojawić się tekst pokazany na rysunku 1-14.

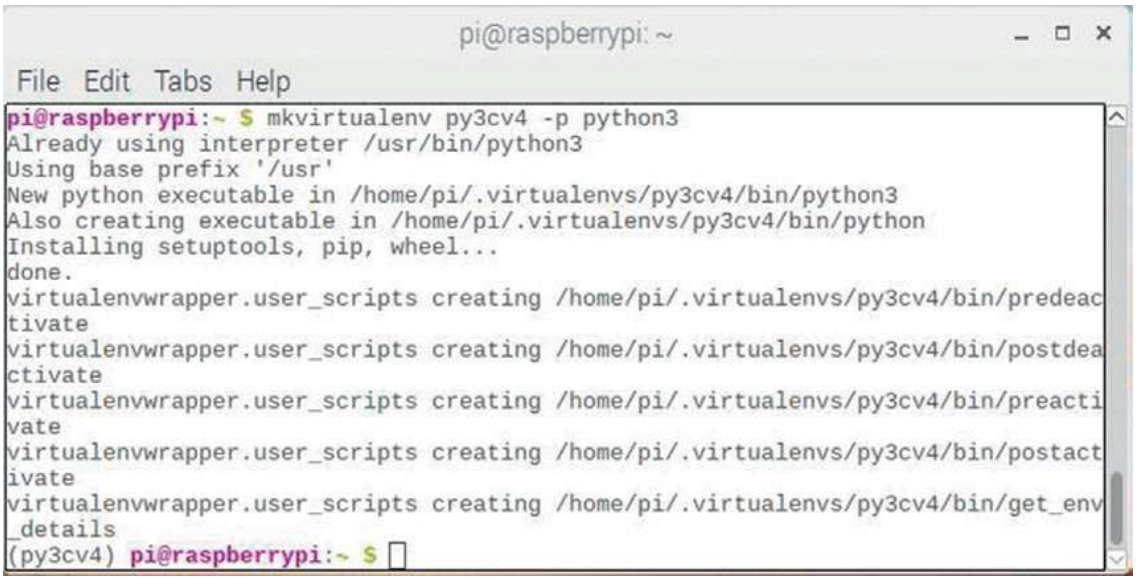


Rysunek 1-14. *Początkowe wyniki polecenia source*

5. Ten krok w istocie tworzy środowisko wirtualne przy użyciu narzędzi `virtualenv` i `virtualenvwrapper`, zainstalowanych wcześniej w kroku 2. Będzie trzeba podać unikalną nazwę środowiska. W tym przykładzie użyłem py3cv4_1. Jeśli planujesz generowanie wielu środowisk, wtedy możesz użyć schematu nazw, takich jak py-3cv4_1, py3cv4_2, py3cv4_3 itd. Nazwa py3cv4_1 odwołuje się do faktu, że środowisko wirtualne używa wersji Python 3, a także zawiera pakiet oprogramowania OpenCV 4. Ponadto warto dokumentować przyczynę tworzenia każdego ze środowisk, aby się w tym szybko nie pogubić. Wprowadź następujące polecenie, aby utworzyć środowisko wirtualne Pythona py3cv4_1:

```
mkvirtualenv py3cv4_1 -p python3
```

Utworzenie środowiska wirtualnego zajmuje około 40 sekund. Rysunek 1-15 przedstawia wyniki działania tego polecenia. Zwróć uwagę na napis (`py3cv4_1`) poprzedzający zwykły monit wiersza polecenia. To wskazuje, że obecnie działa środowisko wirtualne.



Rysunek 1-15. *Wyniki polecenia* `mkvirtualenv`

Środowisko wirtualne py3cv4_1 można łatwo zamknąć, po prostu zamykając okno terminala. Zalecam, aby to zrobić.

6. Otwórz nowe środowisko terminala, aby zweryfikować, że możesz uruchomić środowisko wirtualne py3cv4_1. Wprowadź następujące polecenie:

```
source ~/.profile
workon py3cv4_1
```

Polecenie `workon` jest zawarte w pakiecie oprogramowania `virtualenvwrapper`. To polecenie pozwala na łatwe i szybkie uruchomienie dowolnego środowiska wirtualnego Pythona. Rysunek 1-16 przedstawia wyniki powyższych poleceń.

Rysunek 1-16. *Wyniki polecenia* `workon`

Rysunek ten potwierdza, że masz działające środowisko wirtualne przygotowane do przeprowadzenia kolejnych kroków w celu utworzenia środowiska modelu danych.

Instalacja zależności

Następna demonstracja wymaga instalacji kilku pakietów oprogramowania. Niektóre pakiety są już preinstalowane w oryginalnym pobranym obrazie, a inne trzeba jawnie zainstalować. Następujące polecenia służą do instalacji wszystkich pakietów. Otrzymasz informację, jeśli pakiet jest już zainstalowany, w przeciwnym razie nastąpi pełna instalacja. Te polecenia zajmą dłuższy czas, ponieważ pakiety do instalacji mogą być duże i złożone:

```
pip install numpy
pip install scipy
pip install matplotlib
pip install pandas
sudo apt-get install libatlas-base-dev
pip install -U scikit-learn
```

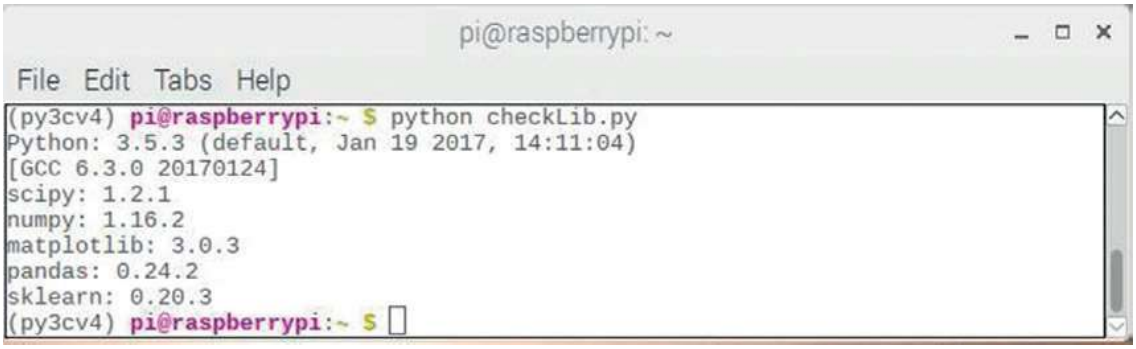
Następujący skrypt Pythona o nazwie `checkLib.py` zwraca numery wersji wszystkich załadowanych pakietów oprogramowania. Zalecam uruchomienie go, aby potwierdzić, że wszystkie zależności są zainstalowane. Ten skrypt jest dostępny w witrynie internetowej towarzyszącej tej książce[1]:

```
# Sprawdzanie wersji bibliotek
# Wersja Pythona
import sys
print('Python: {}'.format(sys.version))
# scipy
import scipy
print('scipy: {}'.format(scipy.__version__))
# numpy
import numpy
print('numpy: {}'.format(numpy.__version__))
```

1 Pliki skryptów dostępne są w witrynie wydawcy pod adresem `https://ksiazki.promise.pl/produkt/uczenie-maszynowe-na-raspberry-pi`
Strona ta zawiera również plik PDF zawierający wszystkie ilustracje w wersji kolorowej.

```
# matplotlib
import matplotlib
print('matplotlib: {}'.format(matplotlib.__version__))
# pandas
import pandas
print('pandas: {}'.format(pandas.__version__))
# scikit-learn
import sklearn
print('sklearn: {}'.format(sklearn.__version__))
```

Rysunek 1-17 przedstawia wyniki uruchomienia przeze mnie tego skryptu.



Rysunek 1-17. *Wyniki skryptu* `checkLib.py`

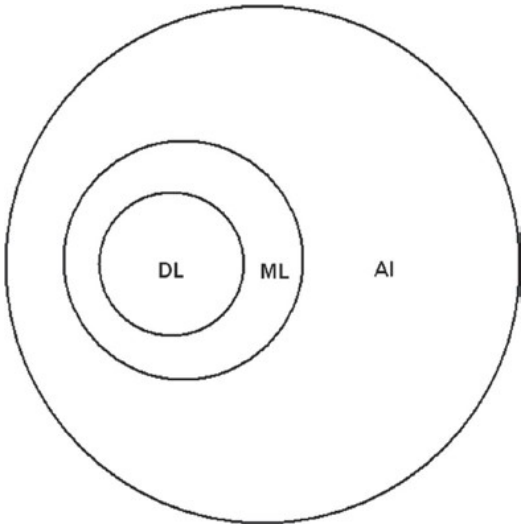
Widoczne wersje prawdopodobnie będą w pewnym stopniu różnić się od tych na rysunkach, ponieważ oprogramowanie open source jest stale modyfikowane. Jednak działanie pakietów powinno być zgodne z wcześniejszymi wersjami, o ile nie zostaną wprowadzone radykalne i nieprzewidziane zmiany. Zwykle ze względów zachowania spójności tak się nie dzieje.

Teraz, gdy wszystkie zależności są zainstalowane i działające, nadszedł czas na zmierzenie się z demonstracją uczenia maszynowego.

Fakty dotyczące uczenia maszynowego

Uczenie maszynowe jest znacznym działem nadrzędnej dziedziny sztucznej inteligencji. Rysunek 1-18 to diagram Venna pokazujący relacje między sztuczną inteligencją (*artificial intelligence* – AI), uczeniem maszynowym (*machine learning* – ML) i uczeniem głębokim (*deep learning* – DL).

Z tego rysunku jasno wynika, że uczenie maszynowe jest ważną częścią sztucznej inteligencji, a uczenie głębokie jest ważną częścią uczenia maszynowego. Jeśli chodzi o bieżące zainteresowania i rozwój, rysunek powinien być odwrócony, ponieważ największa uwaga przywiązywana jest do uczenia głębokiego, uczenie maszynowe ma mniejszą ważność, a jeszcze mniejszą sztuczna inteligencja w ogólności. Hierarchia, w której przeważająca uwaga kierowana jest na uczenie głębokie, jest także zauważalna w tej książce. Jest tak po prostu dlatego, że uczenie głębokie jest istotnym składnikiem implementacji rozpoznawania obrazów, co jest obecnie najważniejszym tematem sztucznej inteligencji i uczenia maszynowego. Bardziej złożonymi kwestiami uczenia głębokiego zajmę się dokładnie w dalszych rozdziałach, jednak najpierw muszę opisać pewne podstawowe tematy uczenia maszynowego.



Rysunek 1-18. *Diagram Venna dotyczący AI, ML i DL*

Podstawy uczenia maszynowego

Jeśli zapytasz tuzina badaczy AI/ML, czym jest uczenie maszynowe, prawdopodobnie otrzymasz tuzin różnych, chociaż umiarkowanie podobnych odpowiedzi. Studiowałem wiele definicji i uważam, że poniższa, utworzona przeze mnie, jest równie prawidłowa, jak pozostałe.

> *Uczenie maszynowe to nauka i sztuka tworzenia algorytmów umożliwiających komputerom uczenie się na podstawie danych bez jawnego programowania.*

Interesujące jest, że znalazłem wiele definicji, w których użyto dokładnie tej samej frazy „bez jawnego programowania", co potwierdziło moje przekonanie, że dowolna czysta aplikacja uczenia maszynowego musi wykluczać wszystkie algorytmy lub systemy, które obejmują wiedzę ekspercką. Zauważ jednak, że systemy eksperckie są ważną częścią sztucznej inteligencji, ale już nie uczenia maszynowego. Jednak zdarzają się systemy hybrydowe, które uwzględniają zarówno uczenie maszynowe, jak i systemy eksperckie, wykorzystując najlepsze możliwości zapewniane przez każdą z tych technologii.

Uczenie maszynowe zostało po raz pierwszy zdefiniowane w roku 1959 przez profesora MIT Arthura Samuela, znanego pioniera zarówno informatyki, jak i sztucznej inteligencji. Profesor Samuel sformułował część „...daje komputerom możliwość uczenia się bez jawnego programowania". Jego dążeniem były komputery programowane przez algorytmy uczące się z danych wejściowych, a następnie dokonujące kolejnych predykcji na podstawie tych samych danych. Oznacza to możliwość całkowitego oddzielenia algorytmów uczących się od jakichkolwiek algorytmów wstępnie zaprogramowanych lub statycznych, a także swobodę podejmowania decyzji lub predykcji sterowanych danymi poprzez budowanie modeli opartych na danych wejściowych.

Modele uczenia maszynowego są głównie używane do predykcji lub klasyfikacji. Warto zatem wprowadzić pewnie fundamentalne koncepcje dotyczące tych operacji przed omówieniem bardziej złożonych zastosowań uczenia maszynowego. To wprowadzenie będzie miało formę małego, ale kompletnego projektu ML.

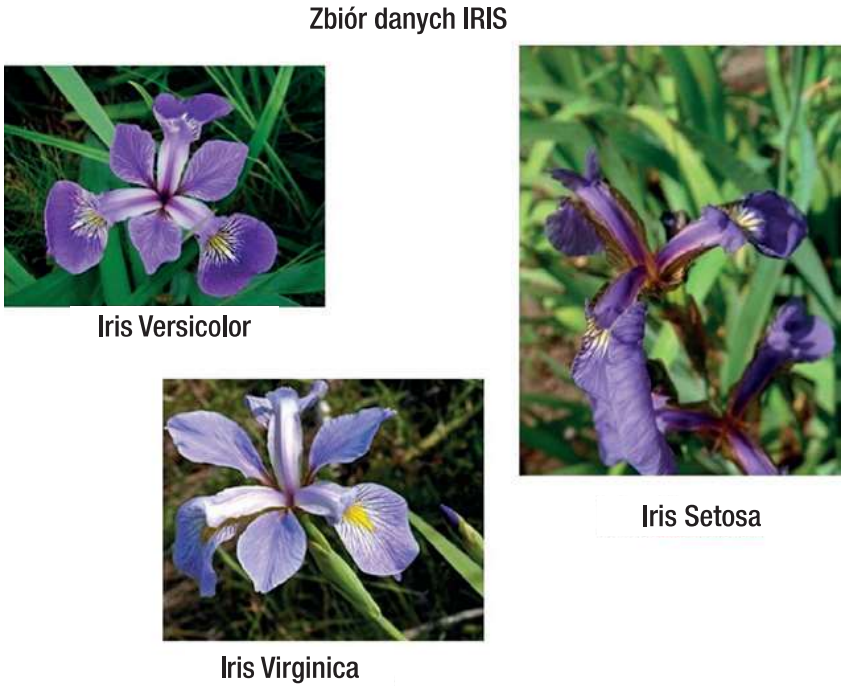
Liniowa predykcja i klasyfikacja

Ten projekt jest oparty głównie na wpisie bloga z czerwca 2016 roku, zatytułowanego *Your First Machine Learning Project in Python Step by Step* (Pierwszy projekt uczenia maszynowego w Pythonie krok po kroku), napisanego przez dr. Jasona Brownlee, który jest obecnie aktywnym badaczem ML mieszkającym w Australii. Polecam zaglądanie do jego bloga pod adresem `MachineLearningMastery.com`, zawierającego bogactwo informacji i zasobów związanych z uczeniem maszynowym. Jason sugeruje i całym sercem się z nim zgadzam, że projekty ML należy rozpoczynać od podejścia strukturalnego składającego się z następujących kroków, które parafrazuję na podstawie bloga:

1. Zdefiniuj problem.

2. Przygotuj i oczyść odpowiednie dane.
3. Oszacuj wszelkie odpowiednie algorytmy.
4. Stale poprawiaj wyniki, aż do otrzymania zanikających przyrostów.
5. Zaprezentuj wyniki w możliwie czytelnej i jednoznacznej formie.

Ten początkowy projekt ML jest sławnym projektem dotyczącym klasyfikacji kwiatów irysów. Zbiór danych Iris dotyczący tych kwiatów jest zbiorem danych wielu zmiennych zaprezentowanym przez brytyjskiego statystyka i biologa Ronalda Fishera w dokumencie z roku 1936 „The Use of Multiple Measurements in Taxonomic Problems" (Zastosowanie wielu miar w problemach taksonomicznych) jako przykład liniowej analizy dyskryminacyjnej (LDA, ang. *linear discriminant analysis*). Ten zbiór danych jest czasem nazywany zbiorem Iris Andersona, ponieważ Edgar Anderson zebrał te dane, aby uzyskać oszacowanie morfologicznych odmian kwiatów trzech powiązanych gatunków irysów. Dwa z trzech gatunków zostały zebrane na Półwyspie Gaspésie w prowincji Quebec w Kanadzie, „wszystkie z tego samego pastwiska i zebrane tego samego dnia i zmierzone w tym samym czasie przez tę samą osobę za pomocą tego samego przyrządu" zgodnie z cytatem z dokumentu Andersona. Zdjęcia trzech gatunków irysów są pokazane na rysunku 1-19.

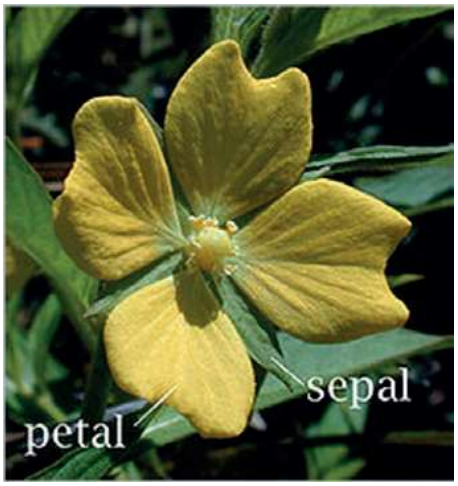


Rysunek 1-19. *Trzy gatunki irysów*

Zbiór danych składa się z 50 próbek z każdego z trzech gatunków irysów (iris setosa, iris virginica i iris versicolor). W każdej próbce zmierzono cztery cechy: długość i szerokość działek kielicha (*sepal*) i płatków (*petal*), w centymetrach. Fisher opracował liniowy model dyskryminacyjny oparty ma kombinacji tych czterech cech w celu rozróżniania gatunków irysów. Działka kielicha jest częścią kwiatów okrytonasiennych (roślin kwitnących) i jest zwykle zielona. Działki kielicha typowo służą jako ochrona dla kwiatu w pąku, a często jako wsparcie dla płatków podczas kwitnienia. Płatki są zmodyfikowanymi liśćmi wokół rozrodczych części kwiatów. Często mają jasne kolory i niezwykłe kształty, aby przyciągać owady zapylające, czyli pszczoły. Rysunek 1-20 pokazuje kwiat (nie irys) z wyróżnioną działką kielicha i płatkiem.

Krok 1 podejścia do rozwiązania problemu jest rozsądnie prosty. Rozpoznać gatunki irysa na podstawie czterech wymiarów opisujący wysokość i szerokość działki kielicha oraz wysokość i szerokość płatka. Wszystkie wymiary powinny być w centymetrach, aby pasować do jednostek w podstawowym zbiorze danych.

Następnym krokiem w procesie rozwiązania jest odniesienie do zbioru danych. Istnieje wiele zasobów online dostępnych do pobrania oryginalnego zbioru danych Iris w formacie CSV. Użyłem zbioru danych Iris Jasona w formacie CSV, który jest udostępniony na github.com. W pierwszej części demonstracji w tym rozdziale skupimy się na zapoznaniu ze zbiorem danych.



Rysunek 1-20. *Działka kielicha (sepal) i płatek (petal)*

Demonstracja Iris – część 1

Przedstawiony poniżej skrypt Pythona nazywa się `irisDemo.py` i został utworzony do pracy z wieloma kolejnymi krokami, aby zapoznać Czytelników z właściwościami i charakterystykami danych. Zapoznanie z tymi danymi pomoże wybrać właściwe algorytmy, które będą najlepiej pasować do wymagań. Te kroki to:

- Ładowanie zależności.
- Ładowanie zbioru danych.
- Wyświetlanie wymiarów zbioru danych.
- Wyświetlenie pierwszych 20 rekordów zbioru danych.
- Wyświetlanie statystyk zbioru danych.
- Wyświetlanie klas zbioru danych i skojarzonych rozmiarów.
- Wykresy jednej zmiennej i wielu zmiennych danych.

Poniższy skrypt jest w całości dostępny w witrynie internetowej dotyczącej tej książki. Opiszę wyniki całego skryptu po listingu kodu.

```
# Użycie
# python irisDemo.py

# Ładowanie bibliotek
import pandas
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt

# Ładowanie zbioru danych
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petalwidth',
'class']
dataset = pandas.read_csv(url, names=names)

# Wyświetlanie kształtu
print('Dataset dimensions')
print(dataset.shape)
```

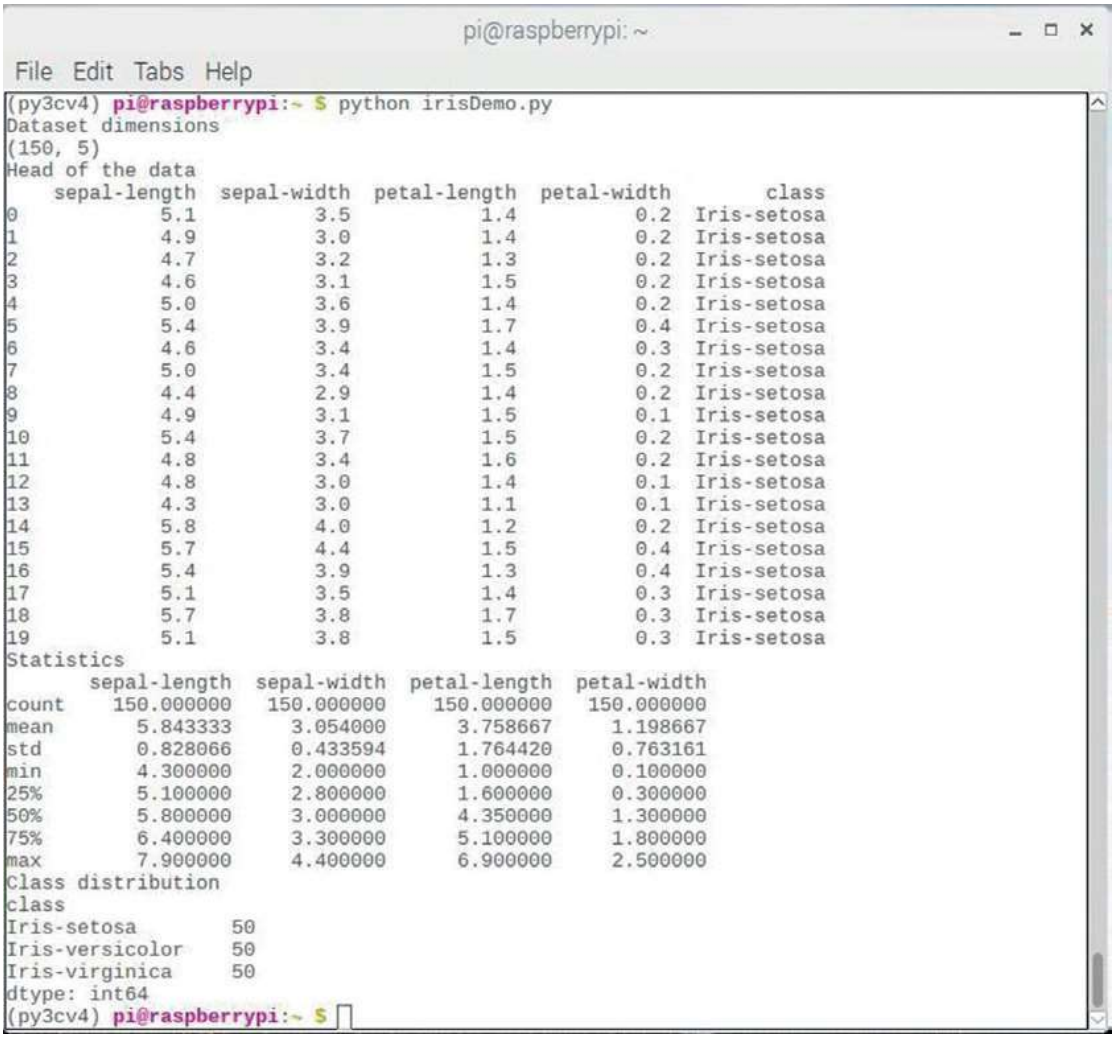
```
# Wyświetlanie pierwszej części zbioru danych
print('Head of the data')
print(dataset.head(20))

# Wyświetlanie statystyk danych
print('Statistics')
print(dataset.describe())

# Wyświetlanie rozkładu klas
print('Class distribution')
print(dataset.groupby('class').size())

# Wyświetlanie danych na wykresach pudełkowych z wąsami
dataset.plot(kind='box', subplot=True, layout=(2,2), sharex=False,
sharey=False)
plt.show()

# Wizualizacja danych na histogramach
dataset.hist()
plt.show()

# Wizualizacja danych na wykresach punktowych
scatter_matrix(dataset)
plt.show()
```

Rysunek 1-21 przedstawia pierwszą część wyników skryptu irisDemo.

Pierwsze dwa wiersze przedstawiają wymiary zbioru danych, czyli 150 wierszy o 5 kolumnach. Następnie widać pierwsze 20 wierszy zbioru danych. Nagłówki kolumn są wyraźnie pokazane z wszystkimi wierszami danych wyświetlonymi w formie tabelarycznej. Ten listing powinien zapewnić dobry wgląd w dane, które mają być przetwarzane.

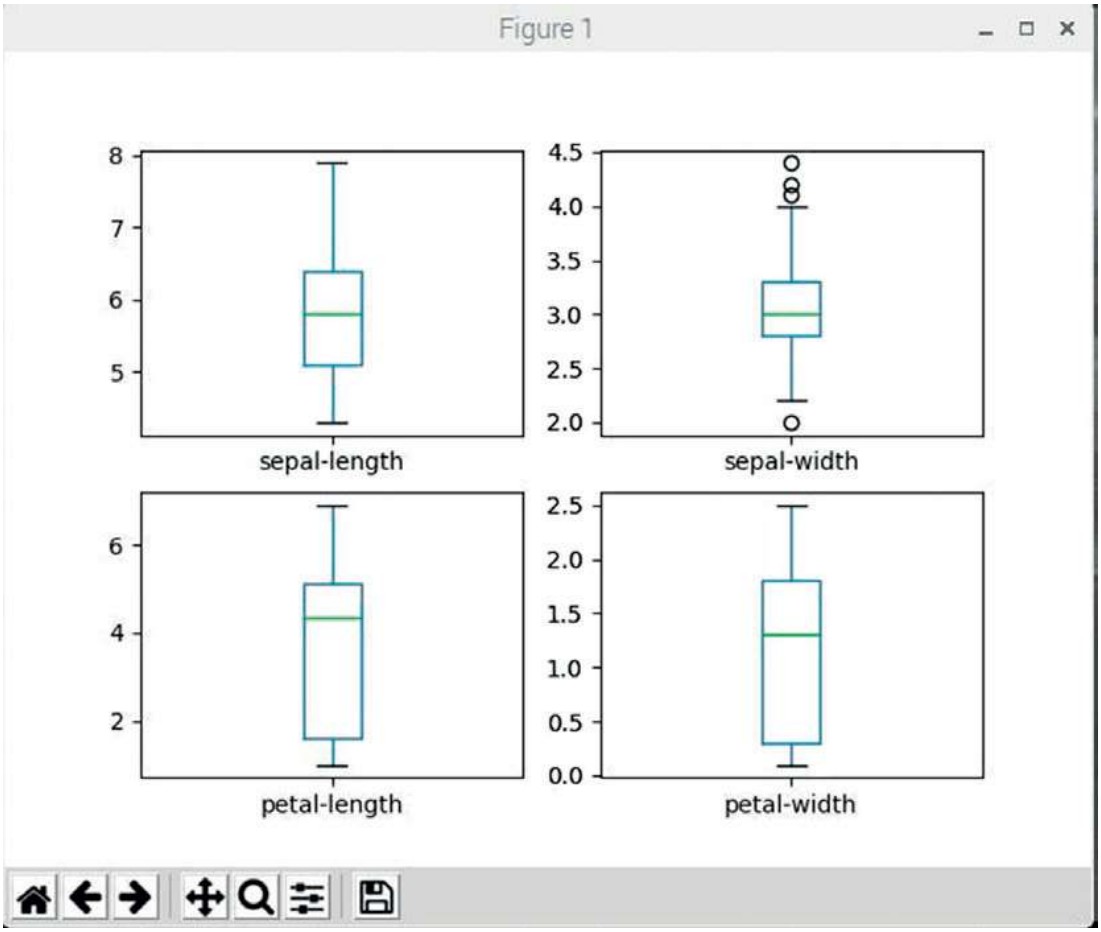
Następnie pokazano mały ekran statystyk pokazujący klasyczne miary statystyczne dla kolumn zbioru danych obejmujące średnie, odchylenia standardowe, wartości minimalne/maksymalne, a także poziomy 25, 50 i 75 percentyla.

Na końcu widać, ile elementów zostało zaliczonych do każdej z klas irysów w zbiorze danych. Zgodnie z oczekiwaniami w każdej klasie zgłoszono 50 elementów, co dokładnie odpowiada oczekiwanym wartościom.



Rysunek 1-21. *Początkowa część wyników irisDemo*

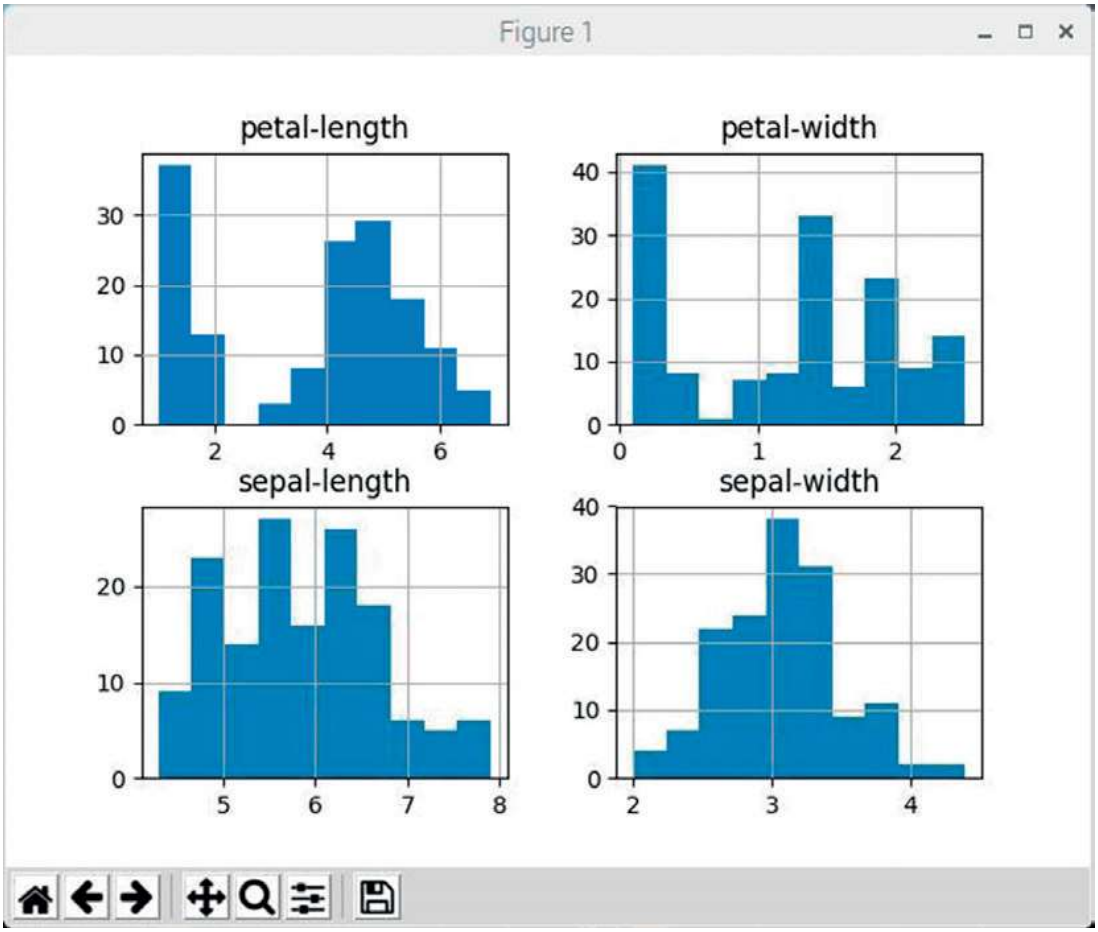
Rysunek 1-22 przedstawia wykresy „pudełkowe z wąsami” dla każdego z czterech atrybutów klas.



Rysunek 1-22. *Wykres pudełkowy z wąsami atrybutów klas*

Te wykresy jednej zmiennej są przydatnym dodatkiem pomocnym w zrozumieniu liczbowych rozkładów skojarzonych z każdym atrybutem. Może okazać się, że pewne modele mogą operować na szerokich rozkładach liczbowych, a inne są bardziej czułe, co może prowadzić do niepożądanych wyników. Przegląd wykresów wskazuje lekko szerszy rozkład liczbowy atrybutów długości płatka i szerokości płatka w porównaniu z tymi samymi atrybutami działki kielicha. Ponadto okazuje się, że jest niewiele danych o wartościach odstających z atrybutem szerokości działki kielicha, co może powodować problemy w pewnych modelach. Te wykresy zostały po prostu zaprojektowane, aby zapewnić dalszy wgląd w dane w celu ułatwienia wyjaśnienia ewentualnych dziwnych wyników modelu.

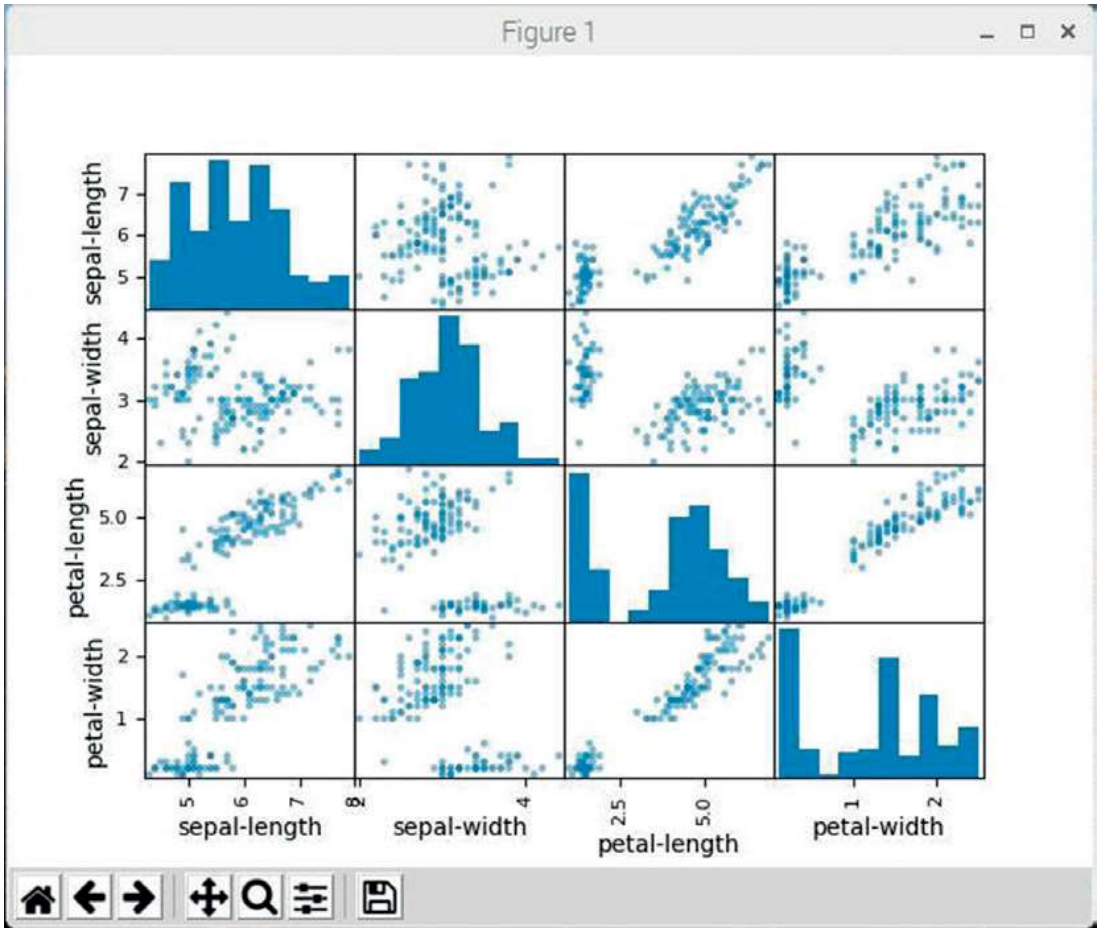
Innym podejściem do wizualizacji danych jest tworzenie histogramów każdej zmiennej wejściowej. Rysunek 1-23 przedstawia histogramy dotyczące wszystkich czterech atrybutów irysów.



Rysunek 1-23. *Histogramy dotyczące atrybutów klas*

Okazuje się, że dwa atrybuty działki kielicha mają rozkład Gaussa lub zbliżony do rozkładu Gaussa. Ta informacja może być przydatna podczas wybierania odpowiedniego modelu lub algorytmu do celów predykcji. Atrybuty płatka sprawiają wrażenie histogramów bimodalnych, co jest interesującym faktem, który może pomóc w doborze algorytmu, takiego jak metoda binaryzacji Otsu.

Innym podejściem do zbadania zbioru danych jest sprawdzenie strukturalnych związków między atrybutami. To podejście jest znane jako analiza wielu zmiennych. Rysunek 1-24 pokazuje wykresy punktowe wszystkich par atrybutów klas.



Rysunek 1-24. *Wykresy punktowe atrybutów klas*

Te wykresy są pomocne w wizualizacji związków, których nie można łatwo wykryć podczas samego przeglądania listingów liczbowych. Zauważ diagonalne grupowanie niektórych par atrybutów na rysunku. To stanowczo sugeruje wysoką korelację między atrybutami oraz to, że prawdopodobnie istnieje mierzalny związek.

W następnej części demonstracji dane zostaną podane na wejście do szeregu modeli i nastąpi przeprowadzenie predykcji. Muszę ostrzec, że popełnię kardynalny błąd pisarski w książce technicznej, używając modeli po prostu bez wcześniejszego wprowadzenia. Wiedz jednak, że nadgonię zaległości i szczegółowo opiszę algorytmy modeli w dalszej części tej książki o podstawach uczenia maszynowego (w tym rozdziale lub w kolejnych).

Demonstracja Iris – część 2

Przedstawiony dalej listing to skrypt Pythona o nazwie `irisDemoTest.py`, który został utworzony do testowania wielu modeli przy użyciu zbioru danych Iris i wyznaczenia ich dokładności w opisie gatunków irysów na postawie danego zbioru atrybutów irysów. Testy będą przeprowadzone w szeregu kroków w sposób podobny do dokonanego w części 1. Te kroki to

- Importowanie modeli.
- Utworzenie zbiorów danych uczącego i walidacyjnego.
- Konfigurowanie środowiska testowego z zastosowaniem 10-krotnej walidacji krzyżowej.
- Użycie sześciu różnych modeli do opisania gatunków irysów na podstawie pomiarów atrybutów.
- Wybranie dokładnego modelu.

Poniższy skrypt jest dostępny w witrynie internetowej towarzyszącej tej książce. Wszystkie części skryptu i ich powiązania z powyższymi krokami zostaną opisane poniżej listingu.

```
# Użycie
# python irisDemoTest.py

# Ładowanie bibliotek
import pandas
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

# Ładowanie zbioru danych
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.
csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',
'class']
dataset = pandas.read_csv(url, names=names)

# Tworzenie zbiorów danych uczącego i walidacyjnego
array = dataset.values
X = array[:,0:4]
Y = array[:,4]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_
split(X, Y, test_size=validation_size, random_state=seed)

# Określanie kryterium oceny
scoring = 'accuracy'

# Budowanie wszystkich modeli
models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_
class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))

# Szacowanie każdego modelu
results = []
names = []
```

```
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train,
    cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f(%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

Lista importowanych bibliotek zmieniła się znacznie w porównaniu z pierwszym skryptem. Teraz zawiera sześć modeli z pakietu sklearn. Oto lista modeli w kolejności alfabetycznej

- DecisionTreeClassifier (klasyfikator drzewa decyzyjnego)
- GaussianNB (naiwny klasyfikator Bayesa z funkcją gęstości Gaussa)
- KNeighborsClassifier (klasyfikator k najbliższych sąsiadów)
- LinearDiscriminantAnalysis (liniowa analiza dyskryminacyjna)
- LogisticRegression (regresja logistyczna)
- SVC (klasyfikator wektorów nośnych)

Jak wspomniałem wcześniej, nie będę tu omawiać żadnych szczegółów działania modeli.

Zbiór danych jest następnie ładowany w dokładnie ten sam sposób, jak zostało to przeprowadzone w pierwszym skrypcie. Ważne jest, aby użyć dokładnie tego zbioru danych, co zauważyłem, próbując zbiorów danych Iris z innego źródła. Podejrzewam, że małe zmiany formatowania powodowały problemy podczas próby użycia nowego zbioru danych.

Następna część skryptu realizuje krok dotyczący tworzenia zbiorów uczącego i walidacyjnego. Osiemdziesiąt procent oryginalnego zbioru danych zostanie użyte do uczenia modelu, a 20% zostanie przydzielone do walidacji. Podczas procesu walidacji mały podzbiór danych zostanie wprowadzony do modeli, które nie były uczone tymi danymi. Wyniki wyjściowe z modeli zostaną następnie porównane z etykietą gatunku dla każdego rekordu w zbiorze walidacyjnym. Następnie zostanie obliczona procentowa dokładność (*accuracy*) jako prosta proporcja między liczbą poprawie rozpoznanych gatunków a całkowitą liczbą rekordów w zbiorze danych.

Następnym krokiem procesu testowania jest skonfigurowanie pętli, która implementuje procedurę 10-krotnej walidacji krzyżowej dla każdego modelu. To oznacza, że wejściowy zbiór danych jest początkowo dzielony na dziesięć części. Uczenie jest przeprowadzane na dziewięciu z dziesięciu części, a walidacja jest wykonywana przy użyciu tylko dziesiątej części.

Wyniki są zapisywane, a następnie zbiór danych jest losowo dzielony ponownie na dziesięć części. Ten proces jest powtarzany dziesięć razy, stąd nazwa 10-krotna. Chcę zwrócić uwagę na coś, co może być mylące dla czytelników. Zbiór danych używany do procedury 10-krotnej walidacji krzyżowej jest tylko zbiorem uczącym, a nie walidacyjnym. Walidacyjny zbiór danych nie będzie używany do czasu następnego skryptu, którego opis zawiera szczegółowe informacje o dokładności predykcji modelu.

Pętla w skrypcie przeprowadza ocenę walidacji krzyżowej dla każdego z modeli. Wyniki są przedstawione na rysunku 1-25.

Rysunek 1-25. *Wyniki walidacji krzyżowej dla sześciu modeli*

Możesz zobaczyć, że wszystkie wyniki są znacznie powyżej 90%, co oznacza, że prawdopodobnie wszystkie dobrze opisują gatunki irysów dla podanego zbioru atrybutów klas. Liczby w nawiasach to odchylenia standardowe dotyczące wyniku każdego modelu. Powinno być widoczne, że odchylenia są względnie małe, co ponownie oznacza, że modele są dobrymi sposobami opisu. Kiedy spojrzymy na rysunek, widać, że maszyna wektorów nośnych (SVM, ang. *support vector machine*) ma najlepszy wynik z oceną 0.991667 i odchylenie standardowe wynoszące tylko 0.025. Jednak, czy jest to rzeczywiście najlepszy model do naszych celów? To zostanie opisane w części 3 tej demonstracji.

Demonstracja Iris – część 3

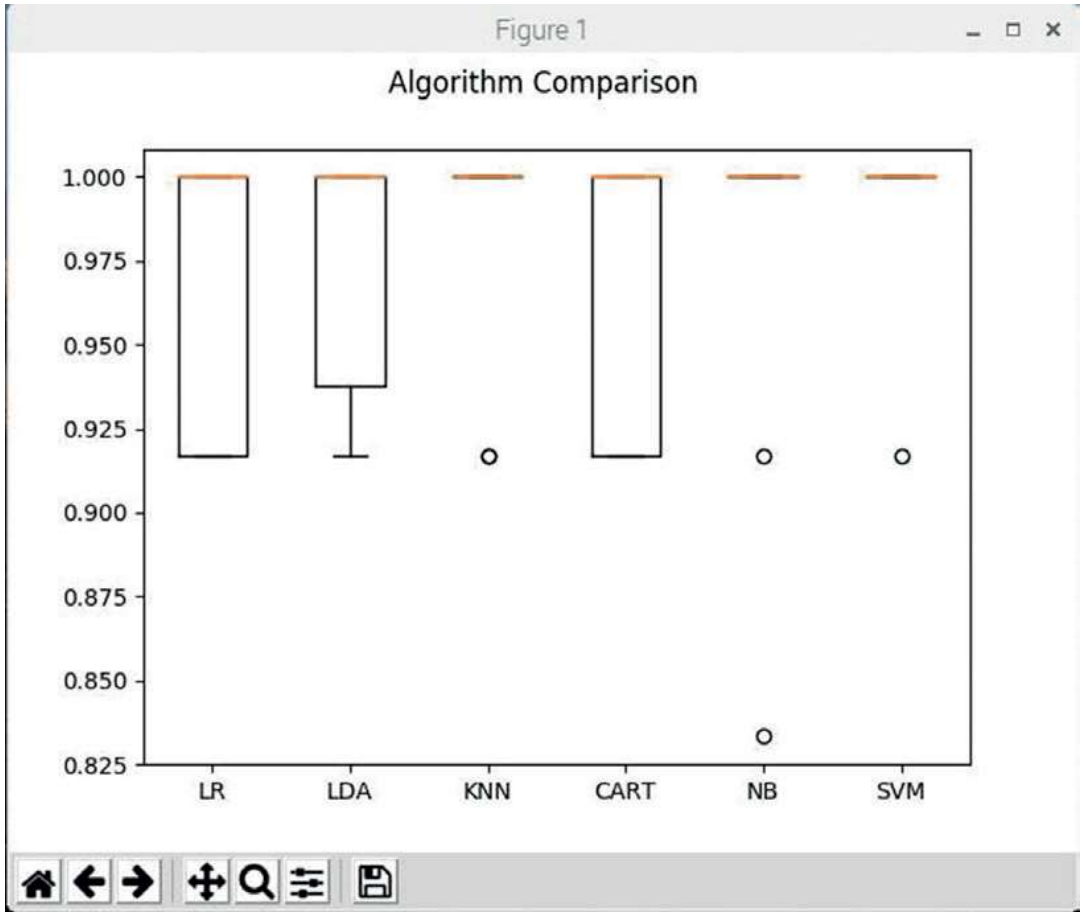
Następny opis dotyczy porównania dokładności sześciu modeli przy użyciu wykresów pudełkowych z wąsami. Użyty skrypt ma nazwę `irisDemoSelection.py` i jest zasadniczo taki sam jak poprzedni skrypt, ale różni się dodaniem funkcji wykresów, a także kilkoma algorytmami użytymi do uszczegółowienia wydajności wybranego modelu. Ekran dokładności sześciu modeli został także wyeliminowany z tego skryptu:

```
# Ładowanie bibliotek
import pandas
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

# Ładowanie zbioru danych
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.
csv"
names = ['sepal-length', 'sepal-width', 'petal-length',
'petal-width', 'class']
dataset = pandas.read_csv(url, names=names)

# Tworzenie zbiorów danych uczącego i walidacyjnego
array = dataset.values
X = array[:,0:4]
Y = array[:,4]
validation_size = 0.20
```

```
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_
split(X, Y, test_size=validation_size, random_state=seed)

# Określanie kryteriów oceny
scoring = 'accuracy'

# Budowanie wszystkich modeli
models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_
class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))

# Szacowanie każdego modelu
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train,
    cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)

# Kreślenie wyników modeli
figure = plt.figure()
figure.suptitle('Algorithm Comparison')
algPlot = figure.add_subplot(1, 1, 1)
plt.boxplot(results)
algPlot.set_xticklabels(names)
plt.show()
```

```
# Predykcja kNN
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
predictions = knn.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions)
```

Rysunek 1-26 przedstawia wyniki wszystkich modeli na wykresie pudełkowym z wąsami.



Rysunek 1-26. *Wykres pudełkowy z wąsami dla wyników modeli*

Możemy zauważyć, że wykresy pudełkowe z wąsami są stłoczone u góry zakresu osi y, ilustrując, że połowa modeli osiąga dokładność zbliżoną do 100% lub całkiem stuprocentową. Wybranie najlepiej działającego modelu w tej sytuacji jest prawie niemożliwe. Jednak wybrałem model kNN do dokładniejszego zbadania jego wydajności. Model

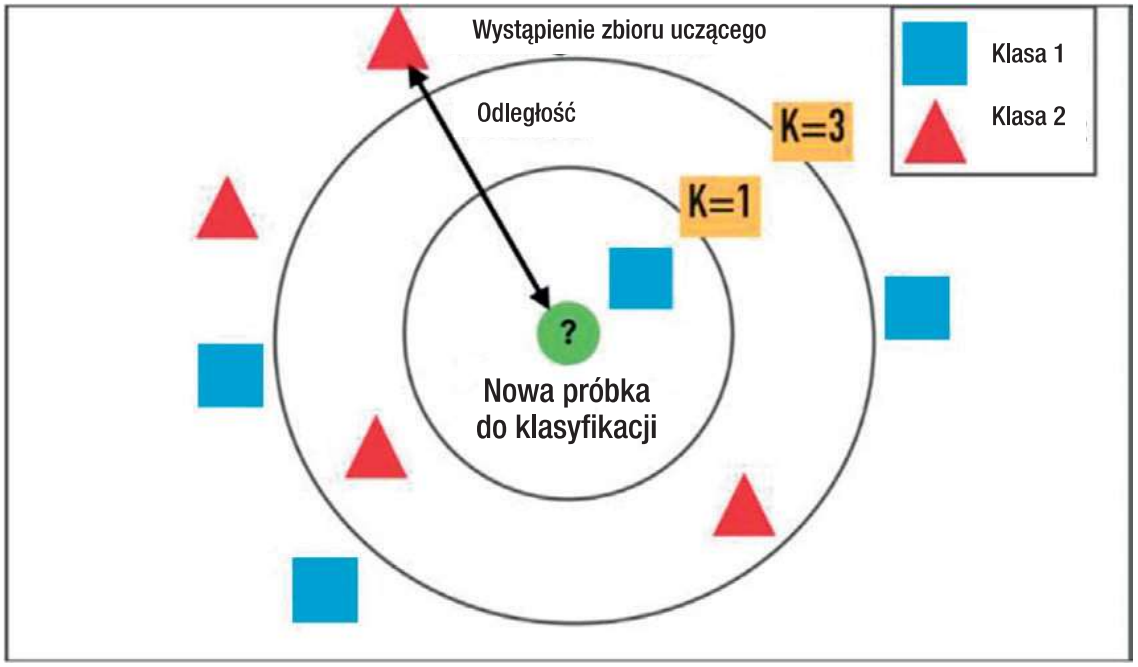
kNN jest prosty i dokładny. Formalna nazwa modelu kNN to algorytm k-najbliższych sąsiadów i jest jednym z najprostszych istniejących algorytmów klasyfikacji, a całkiem możliwe, że również jednym z najszerzej stosowanych. Model kNN jest nieparametrycznym, leniwym algorytmem uczenia. Jest głównie używany ze zbiorem danych, w którym punkty danych są podzielone na kilka klas w celu predykcji klasyfikacji nowego punktu danych lub próbki. Mówiąc prościej, algorytm kNN został opisany jako algorytm implementujący powiedzenie „powiedz mi, kim są twoi sąsiedzi, a powiem ci kim jesteś".

Opisałem model kNN jako nieparametryczny, co oznacza, że model nie przyjmuje żadnych założeń dotyczących rozkładu badanych danych. Innymi słowy, struktura modelu jest wyznaczana na podstawie danych. Biorąc ten fakt pod uwagę, model kNN powinien być prawdopodobnie wybierany jako jeden z pierwszych sposobów badania klasyfikacji, gdy całkowicie lub częściowo brakuje nam wcześniejszej wiedzy o rozkładzie danych.

Model kNN jest też algorytmem leniwym (ang. *lazy*) w przeciwieństwie do algorytmów gorliwych (ang. *eager*). Oznacza to, że algorytm nie używa punktów danych szkoleniowych do dokonywania jakichkolwiek uogólnień. Innymi słowy, nie ma jawnej fazy szkolenia lub jest ona bardzo minimalna. Wynika z tego, że jakakolwiek faza uczenia ma możliwie najkrótszy czas trwania, co jest ważną kwestią dla dużych zbiorów danych. Brak generalizacji oznacza także, że kNN utrzymuje wszystkie dane uczące. Dokładniej, brak uogólnień oznacza, że większość danych uczących (jeśli nie wszystkie) ma zastosowanie podczas fazy walidacji/testowania.

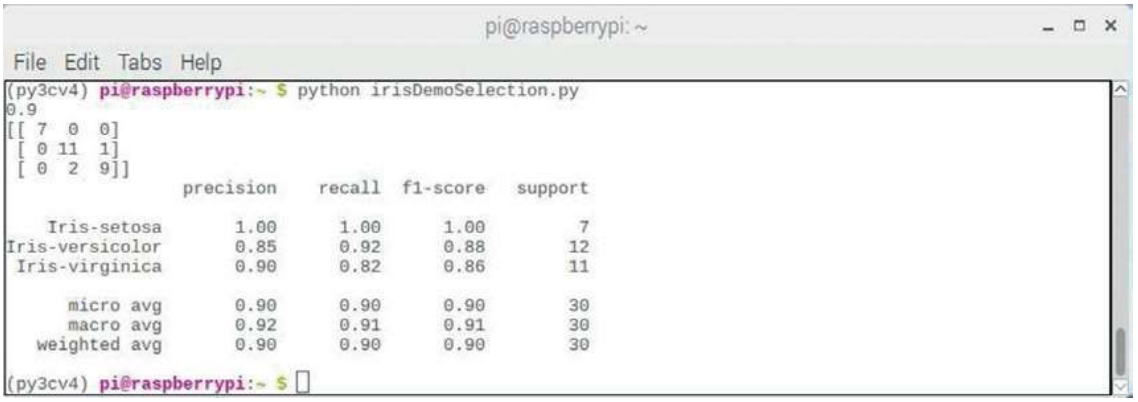
Algorytm kNN jest oparty na podobieństwie cech. Oznacza to, że informacja, jak blisko cecha z poza próby przypomina zbiór uczący, wyznacza klasyfikację danego punktu danych. Ten proces powinien stać się jaśniejszy po zbadaniu rysunku 1-27, graficznego przykładu klasyfikacji kNN.

Próbka testowa (w wewnętrznym okręgu) może zostać sklasyfikowana albo do pierwszej klasy niebieskich kwadratów, albo do drugiej klasy czerwonych trójkątów. Jeśli $k = 3$ (w zewnętrznym okręgu), próbka jest przypisywana do drugiej klasy, ponieważ wewnątrz wewnętrznego okręgu są 2 trójkąty i tylko 1 kwadrat. Jeśli na przykład $k = 5$, nowa próbka byłaby przypisana do pierwszej klasy (3 kwadraty i 2 trójkąty poza zewnętrznym okręgiem).



Rysunek 1-27. *Klasyfikacja kNN*

Ostatnia część skryptu uruchamia model kNN bezpośrednio na zbiorze walidacyjnym. Podsumowanie wyników dla ostatecznego wyniku dokładności, macierz pomyłek i raport klasyfikacji są pokazane na rysunku 1-28.



Rysunek 1-28. *Wyniki walidacji modelu kNN*

Pierwszy wiersz pokazuje wynik dokładności 0.9, czyli 90%. Ten wynik dokładności to sumaryczna ocena pokazująca proporcję poprawnych predykcji do rozmiaru całego zbioru danych. W tym przypadku zbiór danych miał 30 rekordów, a liczba poprawnych predykcji wyniosła 27, co dało wynik 0.9. Pamiętaj, że ten wynik dokładności jest oparty

na użyciu zbioru walidacyjnego, który zawierał 20% oryginalnego zbioru danych Iris. Wynik dokładności pokazany na rysunku 1-28 jest sztucznie zawyżony z powodu natury testowania metodą 10-krotnej walidacji krzyżowej. Niższy wynik dokładności odzwierciedla warunki świata rzeczywistego i powinien być uznany za bardziej wiarygodny.

Macierz pomyłek (macierz błędów) wskazuje, gdzie wystąpiły trzy błędy. Tabela 1-1 zawiera szczegóły dotyczące rzeczywistych klas względem klas przewidywanych.

Tabela 1-1. *Rzeczywiste a przewidywane klasy*

		Rzeczywista klasa		
		Iris setosa	**Iris versicolor**	**Iris virginica**
Przewidywana klasa	Iris setosa	7	0	0
	Iris versicolor	0	11	1
	Iris virginica	0	2	9

W końcu raport klasyfikacji dostarczony przez metryki sklearn zawiera rozbicie wyników klas według precyzji (ang. *precision*), czułości (ang. *recall*), oceny F1 (ang. *F1-score*) i nośności (ang. *support*). Ten raport pokazuje wyniki od bardzo dobrego do wyśmienitego, chociaż zbiór walidacyjny był mały. Każda z tych metryk jest wyjaśniona w tabeli 1-2.

Tabela 1-2. *Analiza metryki uczenia sklearn*

Metryka	**Iris setosa**	**Iris versicolor**	**Iris virginica**	**Spostrzeżenia (patrz uwagi)**
precyzja	1.00	0.85	0.90	precyzja = tp / (tp + fp)
czułość	1.00	0.92	0.82	czułość = tp / (tp + fn)
ocena f1	1.00	0.88	0.86	średnia (precyzja + czułość)
nośność	7	12	11	rozmiar klasy

Uwagi: tp = prawdziwe wyniki pozytywne (true-positive)
fp = fałszywe wyniki pozytywne (false-positive)
fn = fałszywe wyniki negatywne (false-negative)

Zgodnie z intuicją, *precyzja* jest zdolnością klasyfikatora nie etykietowania jako pozytywnej próbki, która jest negatywna.

Czułość to zgodnie z intuicją zdolność klasyfikatora do znalezienia wszystkich próbek pozytywnych.

Ocena f1 (F-beta) może być interpretowana jako ważona średnia harmoniczna precyzji i czułości, gdzie ocena F-beta osiąga najlepszą wartość przy 1 i najgorszą ocenę przy 0.

Warto zapoznać się z dokumentacją biblioteki sklearn, aby dowiedzieć się więcej na temat wartości `micro`, `macro` i `weighted` parametru `average`. Jest ona dostępna na stronie

```
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.
precision_recall_fscore_support.html
```

Rozdział 2

Badanie modelu danych uczenia maszynowego: część 1

Główną treścią tego rozdziału jest omówienie i demonstracja podstawowych modeli danych stosowanych w uczeniu maszynowym. Zanim jednak dojdę do sedna działania modeli danych, przedstawię sposób instalacji pakietów oprogramowania OpenCV 4 i Seaborn. Oba te pakiety są niezbędne do poprawnej obsługi działania i wizualizacji podstawowych modeli danych. Będą one również wymagane do innych demonstracji w dalszych rozdziałach książki.

Instalacja OpenCV 4

Ten podrozdział dotyczy instalacji pakietu oprogramowania OpenCV na licencji open source. Będę używać OpenCV do różnych demonstracji uczenia maszynowego z zastosowaniem wielu różnorodnych narzędzi wizualizacji zawartych w tym pakiecie. Najnowsza wersja to OpenCV 4, która jeszcze nie jest dostępna do bezpośredniego pobrania i instalacji z żadnego z popularnych repozytoriów. Trzeba ją załadować w postaci kodu źródłowego i skompilować na miejscu. Do wykonania tego zadania służą poniższe instrukcje. Ważne jest, aby dokładnie ich przestrzegać, w przeciwnym razie instalacja OpenCV może się nie udać.

Pierwszym krokiem jest instalacja narzędzia CMake wraz z trzema innymi kluczowymi narzędziami. Wprowadź poniższe polecenie:

```
sudo apt-get install build-essential cmake unzip pkg-config
```

Następnie zainstaluj trzy biblioteki graficzne i wideo, które obsługują trzy najbardziej popularne formaty obrazów, jpeg, png i tiff. Wprowadź następujące polecenie:

```
sudo apt-get install libjpeg-dev libpng-dev libtiff-dev
```

Aby zapewnić pomyślne wykonanie powyższego polecenia, upewnij się, że narzędzie apt-get jest zaktualizowane zgodnie z poniższym poleceniem:

```
sudo apt-get update
```

Teraz zainstaluj trzy narzędzia graficzne udostępniające popularne funkcje przetwarzania wideo. Wprowadź poniższe polecenie. Podobnie upewnij się, że narzędzie apt-get jest zaktualizowane.

```
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
```

Następnie zainstaluj dwie uzupełniające biblioteki do przetwarzania wideo. Wprowadź poniższe polecenie:

```
sudo apt-get install libxvidcore-dev libx264-dev
```

Następne polecenie instaluje bibliotekę GTK. Biblioteka GTK będzie używana do implementacji zaplecza GUI biblioteki OpenCV. Wprowadź poniższe polecenie:

```
sudo apt-get install libgtk2.0-dev
```

Następne polecenie redukuje lub eliminuje niepożądane ostrzeżenia GTK. Znak „*” w poleceniu zapewnia załadowanie właściwych modułów obsługujących procesor ARM. Wprowadź poniższe polecenie:

```
sudo apt-get install libcanberra-gtk*
```

Następne dwa pakiety oprogramowania służą do optymalizacji numerycznych OpenCV. Wprowadź poniższe polecenie:

```
sudo apt-get install libatlas-base-dev gfortran
```

Teraz, kiedy wszystkie poprzednie zależności zostały załadowane, możesz przystąpić do pobrania kodu źródłowego OpenCV 4.

Pobieranie kodu źródłowego OpenCV 4

Przed rozpoczęciem pobierania upewnij się, że jesteś w środowisku wirtualnym i w katalogu domowym. Wprowadź następujące polecenie, aby przejść do swojego katalogu domowego, jeśli w nim nie jesteś:

```
cd ~
```

Następnie użyj polecenia `wget`, aby pobrać zarówno najnowszą bibliotekę OpenCV, jak i moduły opencv_contrib. W czasie pisania tej książki najnowsza wersja miała numer 4.0.1. Kiedy przystąpisz do pobierania, prawdopodobnie będzie on inny. Po prostu podstaw najnowszą wersję we wszystkich miejscach, gdzie widać numer wersji wprowadzony w tym opisie. Moduł `opencv_contrib` zawiera uzupełniające funkcje wprowadzone przez społeczność open source, które będą używane w projektach i demonstracjach w tej książce. Wprowadź poniższe polecenie, aby pobrać spakowany plik OpenCV z witryny internetowej GitHub:

```
wget -O opencv.zip https://github.com/opencv/opencv/archive/4.0.1.zip
```

Wprowadź poniższe polecenie, aby pobrać spakowany plik `opencv_contrib` z witryny internetowej GitHub:

```
wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/
archive/4.0.1.zip
```

Pobrane pliki będzie trzeba teraz wyodrębnić i rozpakować przy użyciu poniższych poleceń:

```
unzip opencv.zip
unzip opencv_contrib.zip
```

Następnie zmień nazwę nowo wygenerowanych katalogów na podane poniżej, aby ułatwić dostęp do pakietów i funkcji OpenCV i zapewnić, że katalogi są nazwane zgodnie z plikiem konfiguracyjnym CMake.