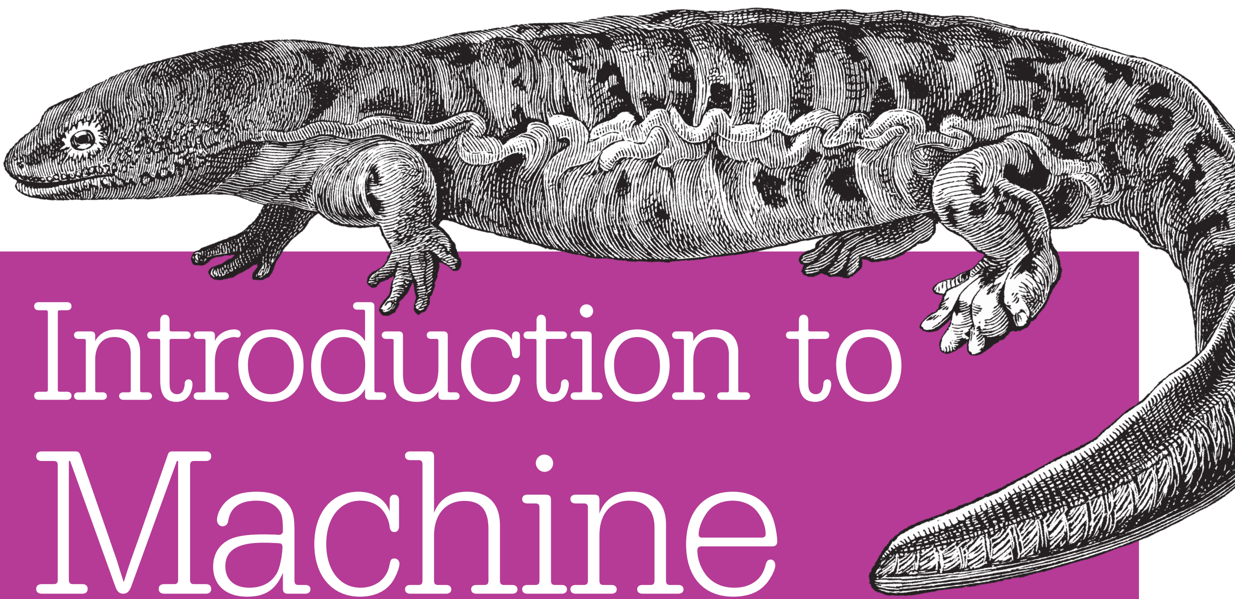
O'REILLY®



Introduction to Machine Learning with Python

A GUIDE FOR DATA SCIENTISTS

powered by jupyter

easy computing

Andreas C. Müller & Sarah Guido

O'REILLY®

Introduction to Machine Learning with Python

Machine learning has become an integral part of many commercial applications and research projects, but this field is not exclusive to large companies with extensive research teams. If you use Python, even as a beginner, this book will teach you practical ways to build your own machine learning solutions. With all the data available today, machine learning applications are limited only by your imagination.

You'll learn the steps necessary to create a successful machine learning application with Python and the scikit-learn library. Authors Andreas Müller and Sarah Guido focus on the practical aspects of using machine learning algorithms, rather than the math behind them. Familiarity with the NumPy and matplotlib libraries will help you get even more from this book.

With this book, you'll learn:

- Fundamental concepts and applications of machine learning
- Advantages and shortcomings of widely used machine learning algorithms
- How to represent data processed by machine learning, including which data aspects to focus on
- Advanced methods for model evaluation and parameter tuning
- The concept of pipelines for chaining models and encapsulating your workflow
- Methods for working with text data, including text-specific processing techniques
- Suggestions for improving your machine learning and data science skills

"This book is a fantastic, super-practical resource for anyone who wants to start using machine learning in Python—I just wish it had existed when I started using scikit-learn!"

—**Hanna Wallach**
Senior Researcher, Microsoft Research

Andreas Müller earned a PhD in machine learning from the University of Bonn. After working as a machine learning researcher on computer vision applications at Amazon, he joined the Center for Data Science at New York University. He's also a maintainer and core contributor to scikit-learn.

Sarah Guido is a data scientist who has spent a lot of time working with startups, and most recently served as Lead Data Scientist at Bitly. An accomplished conference speaker, Sarah earned a Master of Science in Information from the University of Michigan.

PYTHON/MACHINE LEARNING

US $49.99 CAN $57.99
ISBN: 978-1-449-36941-5

easy computing

Twitter: @oreillymedia
facebook.com/oreilly

Introduction to Machine Learning with Python

A Guide for Data Scientists

Andreas C. Müller and Sarah Guido

easy computing

Beijing · Boston · Farnham · Sebastopol · Tokyo

O'REILLY®

Introduction to Machine Learning with Python
by Andreas C. Müller and Sarah Guido

Copyright © 2017 Sarah Guido, Andreas Müller. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com/safari*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editor: Dawn Schanafelt
Production Editor: Kristen Brown
Copyeditor: Rachel Head
Proofreader: Jasmine Kwityn
Indexer: Judy McConville
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Rebecca Demarest

October 2016: First Edition

Revision History for the First Edition

2016-09-22: First Release
2017-01-13: Second Release
2017-06-09: Third Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781449369415* for release details.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Introduction to Machine Learning with Python*, the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

While the publisher and the authors have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the authors disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

978-1-449-36941-5

[LSI]

easy computing

Table of Contents

easy computing

easy computing

easy computing

easy computing

Preface

Machine learning is an integral part of many commercial applications and research projects today, in areas ranging from medical diagnosis and treatment to finding your friends on social networks. Many people think that machine learning can only be applied by large companies with extensive research teams. In this book, we want to show you how easy it can be to build machine learning solutions yourself, and how to best go about it. With the knowledge in this book, you can build your own system for finding out how people feel on Twitter, or making predictions about global warming. The applications of machine learning are endless and, with the amount of data available today, mostly limited by your imagination.

Who Should Read This Book

This book is for current and aspiring machine learning practitioners looking to implement solutions to real-world machine learning problems. This is an introductory book requiring no previous knowledge of machine learning or artificial intelligence (AI). We focus on using Python and the `scikit-learn` library, and work through all the steps to create a successful machine learning application. The methods we introduce will be helpful for scientists and researchers, as well as data scientists working on commercial applications. You will get the most out of the book if you are somewhat familiar with Python and the NumPy and `matplotlib` libraries.

We made a conscious effort not to focus too much on the math, but rather on the practical aspects of using machine learning algorithms. As mathematics (probability theory, in particular) is the foundation upon which machine learning is built, we won't go into the analysis of the algorithms in great detail. If you are interested in the mathematics of machine learning algorithms, we recommend the book *The Elements of Statistical Learning* (Springer) by Trevor Hastie, Robert Tibshirani, and Jerome Friedman, which is available for free at the authors' website. We will also not describe how to write machine learning algorithms from scratch, and will instead focus on

easy computing

how to use the large array of models already implemented in `scikit-learn` and other libraries.

Why We Wrote This Book

There are many books on machine learning and AI. However, all of them are meant for graduate students or PhD students in computer science, and they're full of advanced mathematics. This is in stark contrast with how machine learning is being used, as a commodity tool in research and commercial applications. Today, applying machine learning does not require a PhD. However, there are few resources out there that fully cover all the important aspects of implementing machine learning in practice, without requiring you to take advanced math courses. We hope this book will help people who want to apply machine learning without reading up on years' worth of calculus, linear algebra, and probability theory.

Navigating This Book

This book is organized roughly as follows:

- Chapter 1 introduces the fundamental concepts of machine learning and its applications, and describes the setup we will be using throughout the book.
- Chapters 2 and 3 describe the actual machine learning algorithms that are most widely used in practice, and discuss their advantages and shortcomings.
- Chapter 4 discusses the importance of how we represent data that is processed by machine learning, and what aspects of the data to pay attention to.
- Chapter 5 covers advanced methods for model evaluation and parameter tuning, with a particular focus on cross-validation and grid search.
- Chapter 6 explains the concept of pipelines for chaining models and encapsulating your workflow.
- Chapter 7 shows how to apply the methods described in earlier chapters to text data, and introduces some text-specific processing techniques.
- Chapter 8 offers a high-level overview, and includes references to more advanced topics.

While Chapters 2 and 3 provide the actual algorithms, understanding all of these algorithms might not be necessary for a beginner. If you need to build a machine learning system ASAP, we suggest starting with Chapter 1 and the opening sections of Chapter 2, which introduce all the core concepts. You can then skip to "Summary and Outlook" on page 129 in Chapter 2, which includes a list of all the supervised models that we cover. Choose the model that best suits your needs and flip back to read the

easy computing

section devoted to it for details. Then you can use the techniques in Chapter 5 to evaluate and tune your model.

Online Resources

While studying this book, definitely refer to the `scikit-learn` website for more in-depth documentation of the classes and functions, and many examples. There is also a video course created by Andreas Müller, "Advanced Machine Learning with scikit-learn," that supplements this book. You can find it at *http://bit.ly/advanced_machine_learning_scikit-learn*.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords. Also used for commands and module and package names.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This element signifies a tip or suggestion.

This element signifies a general note.

This icon indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, IPython notebooks, etc.) is available for download at *https://github.com/amueller/introduction_to_ml_with_python*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*An Introduction to Machine Learning with Python* by Andreas C. Müller and Sarah Guido (O'Reilly). Copyright 2017 Sarah Guido and Andreas Müller, 978-1-449-36941-5."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

O'Reilly Safari

Safari (formerly Safari Books Online) is a membership-based training and reference platform for enterprise, government, educators, and individuals.

Members have access to thousands of books, training videos, Learning Paths, interactive tutorials, and curated playlists from over 250 publishers, including O'Reilly Media, Harvard Business Review, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Adobe, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, and Course Technology, among others.

For more information, please visit *http://oreilly.com/safari*.

easy computing

How to Contact Us

Please address comments and questions concerning this book to the publisher:

> O'Reilly Media, Inc.
> 1005 Gravenstein Highway North
> Sebastopol, CA 95472
> 800-998-9938 (in the United States or Canada)
> 707-829-0515 (international or local)
> 707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/intro-machine-learning-python*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

From Andreas

Without the help and support of a large group of people, this book would never have existed.

I would like to thank the editors, Meghan Blanchette, Brian MacDonald, and in particular Dawn Schanafelt, for helping Sarah and me make this book a reality.

I want to thank my reviewers, Thomas Caswell, Olivier Grisel, Stefan van der Walt, and John Myles White, who took the time to read the early versions of this book and provided me with invaluable feedback—in addition to being some of the cornerstones of the scientific open source ecosystem.

I am forever thankful for the welcoming open source scientific Python community, especially the contributors to `scikit-learn`. Without the support and help from this community, in particular from Gael Varoquaux, Alex Gramfort, and Olivier Grisel, I would never have become a core contributor to `scikit-learn` or learned to understand this package as well as I do now. My thanks also go out to all the other contributors who donate their time to improve and maintain this package.

easy computing

I'm also thankful for the discussions with many of my colleagues and peers that helped me understand the challenges of machine learning and gave me ideas for structuring a textbook. Among the people I talk to about machine learning, I specifically want to thank Brian McFee, Daniela Huttenkoppen, Joel Nothman, Gilles Louppe, Hugo Bowne-Anderson, Sven Kreis, Alice Zheng, Kyunghyun Cho, Pablo Baberas, and Dan Cervone.

My thanks also go out to Rachel Rakov, who was an eager beta tester and proofreader of an early version of this book, and helped me shape it in many ways.

On the personal side, I want to thank my parents, Harald and Margot, and my sister, Miriam, for their continuing support and encouragement. I also want to thank the many people in my life whose love and friendship gave me the energy and support to undertake such a challenging task.

From Sarah

I would like to thank Meg Blanchette, without whose help and guidance this project would not have even existed. Thanks to Celia La and Brian Carlson for reading in the early days. Thanks to the O'Reilly folks for their endless patience. And finally, thanks to DTS, for your everlasting and endless support.

easy computing

CHAPTER 1
Introduction

Machine learning is about extracting knowledge from data. It is a research field at the intersection of statistics, artificial intelligence, and computer science and is also known as predictive analytics or statistical learning. The application of machine learning methods has in recent years become ubiquitous in everyday life. From automatic recommendations of which movies to watch, to what food to order or which products to buy, to personalized online radio and recognizing your friends in your photos, many modern websites and devices have machine learning algorithms at their core. When you look at a complex website like Facebook, Amazon, or Netflix, it is very likely that every part of the site contains multiple machine learning models.

Outside of commercial applications, machine learning has had a tremendous influence on the way data-driven research is done today. The tools introduced in this book have been applied to diverse scientific problems such as understanding stars, finding distant planets, discovering new particles, analyzing DNA sequences, and providing personalized cancer treatments.

Your application doesn't need to be as large-scale or world-changing as these examples in order to benefit from machine learning, though. In this chapter, we will explain why machine learning has become so popular and discuss what kinds of problems can be solved using machine learning. Then, we will show you how to build your first machine learning model, introducing important concepts along the way.

Why Machine Learning?

In the early days of "intelligent" applications, many systems used handcoded rules of "if" and "else" decisions to process data or adjust to user input. Think of a spam filter whose job is to move the appropriate incoming email messages to a spam folder. You could make up a blacklist of words that would result in an email being marked as

easy computing

spam. This would be an example of using an expert-designed rule system to design an "intelligent" application. Manually crafting decision rules is feasible for some applications, particularly those in which humans have a good understanding of the process to model. However, using handcoded rules to make decisions has two major disadvantages:

- The logic required to make a decision is specific to a single domain and task. Changing the task even slightly might require a rewrite of the whole system.
- Designing rules requires a deep understanding of how a decision should be made by a human expert.

One example of where this handcoded approach will fail is in detecting faces in images. Today, every smartphone can detect a face in an image. However, face detection was an unsolved problem until as recently as 2001. The main problem is that the way in which pixels (which make up an image in a computer) are "perceived" by the computer is very different from how humans perceive a face. This difference in representation makes it basically impossible for a human to come up with a good set of rules to describe what constitutes a face in a digital image.

Using machine learning, however, simply presenting a program with a large collection of images of faces is enough for an algorithm to determine what characteristics are needed to identify a face.

Problems Machine Learning Can Solve

The most successful kinds of machine learning algorithms are those that automate decision-making processes by generalizing from known examples. In this setting, which is known as *supervised learning*, the user provides the algorithm with pairs of inputs and desired outputs, and the algorithm finds a way to produce the desired output given an input. In particular, the algorithm is able to create an output for an input it has never seen before without any help from a human. Going back to our example of spam classification, using machine learning, the user provides the algorithm with a large number of emails (which are the input), together with information about whether any of these emails are spam (which is the desired output). Given a new email, the algorithm will then produce a prediction as to whether the new email is spam.

Machine learning algorithms that learn from input/output pairs are called supervised learning algorithms because a "teacher" provides supervision to the algorithms in the form of the desired outputs for each example that they learn from. While creating a dataset of inputs and outputs is often a laborious manual process, supervised learning algorithms are well understood and their performance is easy to measure. If your application can be formulated as a supervised learning problem, and you are able to

easy computing

create a dataset that includes the desired outcome, machine learning will likely be able to solve your problem.

Examples of supervised machine learning tasks include:

Identifying the zip code from handwritten digits on an envelope
: Here the input is a scan of the handwriting, and the desired output is the actual digits in the zip code. To create a dataset for building a machine learning model, you need to collect many envelopes. Then you can read the zip codes yourself and store the digits as your desired outcomes.

Determining whether a tumor is benign based on a medical image
: Here the input is the image, and the output is whether the tumor is benign. To create a dataset for building a model, you need a database of medical images. You also need an expert opinion, so a doctor needs to look at all of the images and decide which tumors are benign and which are not. It might even be necessary to do additional diagnosis beyond the content of the image to determine whether the tumor in the image is cancerous or not.

Detecting fraudulent activity in credit card transactions
: Here the input is a record of the credit card transaction, and the output is whether it is likely to be fraudulent or not. Assuming that you are the entity distributing the credit cards, collecting a dataset means storing all transactions and recording if a user reports any transaction as fraudulent.

An interesting thing to note about these examples is that although the inputs and outputs look fairly straightforward, the data collection process for these three tasks is vastly different. While reading envelopes is laborious, it is easy and cheap. Obtaining medical imaging and diagnoses, on the other hand, requires not only expensive machinery but also rare and expensive expert knowledge, not to mention the ethical concerns and privacy issues. In the example of detecting credit card fraud, data collection is much simpler. Your customers will provide you with the desired output, as they will report fraud. All you have to do to obtain the input/output pairs of fraudulent and nonfraudulent activity is wait.

Unsupervised algorithms are the other type of algorithm that we will cover in this book. In unsupervised learning, only the input data is known, and no known output data is given to the algorithm. While there are many successful applications of these methods, they are usually harder to understand and evaluate.

Examples of unsupervised learning include:

Identifying topics in a set of blog posts
: If you have a large collection of text data, you might want to summarize it and find prevalent themes in it. You might not know beforehand what these topics are, or how many topics there might be. Therefore, there are no known outputs.

easy computing

Segmenting customers into groups with similar preferences
Given a set of customer records, you might want to identify which customers are similar, and whether there are groups of customers with similar preferences. For a shopping site, these might be "parents," "bookworms," or "gamers." Because you don't know in advance what these groups might be, or even how many there are, you have no known outputs.

Detecting abnormal access patterns to a website
To identify abuse or bugs, it is often helpful to find access patterns that are different from the norm. Each abnormal pattern might be very different, and you might not have any recorded instances of abnormal behavior. Because in this example you only observe traffic, and you don't know what constitutes normal and abnormal behavior, this is an unsupervised problem.

For both supervised and unsupervised learning tasks, it is important to have a representation of your input data that a computer can understand. Often it is helpful to think of your data as a table. Each data point that you want to reason about (each email, each customer, each transaction) is a row, and each property that describes that data point (say, the age of a customer or the amount or location of a transaction) is a column. You might describe users by their age, their gender, when they created an account, and how often they have bought from your online shop. You might describe the image of a tumor by the grayscale values of each pixel, or maybe by using the size, shape, and color of the tumor.

Each entity or row here is known as a *sample* (or data point) in machine learning, while the columns—the properties that describe these entities—are called *features*.

Later in this book we will go into more detail on the topic of building a good representation of your data, which is called *feature extraction* or *feature engineering*. You should keep in mind, however, that no machine learning algorithm will be able to make a prediction on data for which it has no information. For example, if the only feature that you have for a patient is their last name, no algorithm will be able to predict their gender. This information is simply not contained in your data. If you add another feature that contains the patient's first name, you will have much better luck, as it is often possible to tell the gender by a person's first name.

Knowing Your Task and Knowing Your Data

Quite possibly the most important part in the machine learning process is understanding the data you are working with and how it relates to the task you want to solve. It will not be effective to randomly choose an algorithm and throw your data at it. It is necessary to understand what is going on in your dataset before you begin building a model. Each algorithm is different in terms of what kind of data and what problem setting it works best for. While you are building a machine learning solution, you should answer, or at least keep in mind, the following questions:

easy computing

- What question(s) am I trying to answer? Do I think the data collected can answer that question?
- What is the best way to phrase my question(s) as a machine learning problem?
- Have I collected enough data to represent the problem I want to solve?
- What features of the data did I extract, and will these enable the right predictions?
- How will I measure success in my application?
- How will the machine learning solution interact with other parts of my research or business product?

In a larger context, the algorithms and methods in machine learning are only one part of a greater process to solve a particular problem, and it is good to keep the big picture in mind at all times. Many people spend a lot of time building complex machine learning solutions, only to find out they don't solve the right problem.

When going deep into the technical aspects of machine learning (as we will in this book), it is easy to lose sight of the ultimate goals. While we will not discuss the questions listed here in detail, we still encourage you to keep in mind all the assumptions that you might be making, explicitly or implicitly, when you start building machine learning models.

Why Python?

Python has become the lingua franca for many data science applications. It combines the power of general-purpose programming languages with the ease of use of domain-specific scripting languages like MATLAB or R. Python has libraries for data loading, visualization, statistics, natural language processing, image processing, and more. This vast toolbox provides data scientists with a large array of general- and special-purpose functionality. One of the main advantages of using Python is the ability to interact directly with the code, using a terminal or other tools like the Jupyter Notebook, which we'll look at shortly. Machine learning and data analysis are fundamentally iterative processes, in which the data drives the analysis. It is essential for these processes to have tools that allow quick iteration and easy interaction.

As a general-purpose programming language, Python also allows for the creation of complex graphical user interfaces (GUIs) and web services, and for integration into existing systems.

scikit-learn

`scikit-learn` is an open source project, meaning that it is free to use and distribute, and anyone can easily obtain the source code to see what is going on behind the

easy computing

scenes. The `scikit-learn` project is constantly being developed and improved, and it has a very active user community. It contains a number of state-of-the-art machine learning algorithms, as well as comprehensive documentation about each algorithm. `scikit-learn` is a very popular tool, and the most prominent Python library for machine learning. It is widely used in industry and academia, and a wealth of tutorials and code snippets are available online. `scikit-learn` works well with a number of other scientific Python tools, which we will discuss later in this chapter.

While reading this, we recommend that you also browse the `scikit-learn` user guide and API documentation for additional details on and many more options for each algorithm. The online documentation is very thorough, and this book will provide you with all the prerequisites in machine learning to understand it in detail.

Installing scikit-learn

`scikit-learn` depends on two other Python packages, *NumPy* and *SciPy*. For plotting and interactive development, you should also install `matplotlib`, IPython, and the Jupyter Notebook. We recommend using one of the following prepackaged Python distributions, which will provide the necessary packages:

Anaconda
: A Python distribution made for large-scale data processing, predictive analytics, and scientific computing. Anaconda comes with NumPy, SciPy, `matplotlib`, `pandas`, IPython, Jupyter Notebook, and `scikit-learn`. Available on Mac OS, Windows, and Linux, it is a very convenient solution and is the one we suggest for people without an existing installation of the scientific Python packages. Anaconda now also includes the commercial Intel MKL library for free. Using MKL (which is done automatically when Anaconda is installed) can give significant speed improvements for many algorithms in `scikit-learn`.

Enthought Canopy
: Another Python distribution for scientific computing. This comes with NumPy, SciPy, `matplotlib`, `pandas`, and IPython, but the free version does not come with `scikit-learn`. If you are part of an academic, degree-granting institution, you can request an academic license and get free access to the paid subscription version of Enthought Canopy. Enthought Canopy is available for Python 2.7.x, and works on Mac OS, Windows, and Linux.

Python(x,y)
: A free Python distribution for scientific computing, specifically for Windows. Python(x,y) comes with NumPy, SciPy, `matplotlib`, `pandas`, IPython, and `scikit-learn`.

easy computing

About the Authors

Andreas Müller received his PhD in machine learning from the University of Bonn. After working as a machine learning researcher on computer vision applications at Amazon for a year, he joined the Center for Data Science at New York University. For the last four years, he has been a maintainer of and one of the core contributors to `scikit-learn`, a machine learning toolkit widely used in industry and academia, and has authored and contributed to several other widely used machine learning packages. His mission is to create open tools to lower the barrier of entry for machine learning applications, promote reproducible science, and democratize the access to high-quality machine learning algorithms.

Sarah Guido is a data scientist who has spent a lot of time working in start-ups. She loves Python, machine learning, large quantities of data, and the tech world. An accomplished conference speaker, Sarah attended the University of Michigan for grad school and currently resides in New York City.

Colophon

The animal on the cover of *Introduction to Machine Learning with Python* is a hellbender salamander (*Cryptobranchus alleganiensis*), an amphibian native to the eastern United States (ranging from New York to Georgia). It has many colorful nicknames, including "Allegheny alligator," "snot otter," and "mud-devil." The origin of the name "hellbender" is unclear: one theory is that early settlers found the salamander's appearance unsettling and supposed it to be a demonic creature trying to return to hell.

The hellbender salamander is a member of the giant salamander family, and can grow as large as 29 inches long. This is the third-largest aquatic salamander species in the world. Their bodies are rather flat, with thick folds of skin along their sides. While they do have a single gill on each side of the neck, hellbenders largely rely on their skin folds to breathe: gas flows in and out through capillaries near the surface of the skin.

Because of this, their ideal habitat is in clear, fast-moving, shallow streams, which provide plenty of oxygen. The hellbender shelters under rocks and hunts primarily by sense of smell, though it is also able to detect vibrations in the water. Its diet is made up of crayfish, small fish, and occasionally the eggs of its own species. The hellbender is also a key member of its ecosystem as prey: predators include various fish, snakes, and turtles.

Hellbender salamander populations have decreased significantly in the last few decades. Water quality is the largest issue, [illegible] their respiratory system makes them very sensitive to polluted or murky water. An increase in agriculture and other human

easy computing

activity near their habitat means greater amounts of sediment and chemicals in the water. In an effort to save this endangered species, biologists have begun to raise the amphibians in captivity and release them when they reach a less vulnerable age.

Many of the animals on O'Reilly covers are endangered; all of them are important to the world. To learn more about how you can help, go to *animals.oreilly.com*.

The cover image is from *Wood's Animate Creation*. The cover fonts are URW Typewriter and Guardian Sans. The text font is Adobe Minion Pro; the heading font is Adobe Myriad Condensed; and the code font is Dalton Maag's Ubuntu Mono.

easy computing