CHAPMAN & HALL/CRC
MACHINE LEARNING & PATTERN RECOGNITION

The Pragmatic Programmer for Machine Learning

Engineering Analytics and Data Science Solutions

Marco Scutari
Mauro Malvestio

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

The Pragmatic Programmer for Machine Learning

Machine learning has redefined the way we work with data and is increasingly becoming an indispensable part of everyday life. This book discusses how modern software engineering practices are part of this revolution both conceptually and in practical applications.

Comprising a broad overview of how to design machine learning pipelines as well as the state-of-the-art tools we use to make them, this book provides a multi-disciplinary view of how traditional software engineering can be adapted to and integrated with the workflows of domain experts and probabilistic models.

From choosing the right hardware to designing effective pipeline architectures and adopting software development best practices, this guide will appeal to machine learning and data science specialists, whilst also laying out key high-level principles in a way that is approachable for students of computer science and aspiring programmers.

Chapman & Hall/CRC Machine Learning & Pattern Recognition

A First Course in Machine Learning
Simon Rogers, Mark Girolami

Statistical Reinforcement Learning: Modern Machine Learning Approaches
Masashi Sugiyama

Sparse Modeling: Theory, Algorithms, and Applications
Irina Rish, Genady Grabarnik

Computational Trust Models and Machine Learning
Xin Liu, Anwitaman Datta, Ee-Peng Lim

Regularization, Optimization, Kernels, and Support Vector Machines
Johan A.K. Suykens, Marco Signoretto, Andreas Argyriou

Machine Learning: An Algorithmic Perspective, Second Edition
Stephen Marsland

Bayesian Programming
Pierre Bessiere, Emmanuel Mazer, Juan Manuel Ahuactzin, Kamel Mekhnacha

Multilinear Subspace Learning: Dimensionality Reduction of Multidimensional Data
Haiping Lu, Konstantinos N. Plataniotis, Anastasios Venetsanopoulos

Data Science and Machine Learning: Mathematical and Statistical Methods
Dirk P. Kroese, Zdravko Botev, Thomas Taimre, Radislav Vaisman

Deep Learning and Linguistic Representation
Shalom Lappin

Artificial Intelligence and Causal Inference
Momiao Xiong

Transformers for Machine Learning: A Deep Dive
Uday Kamath, Kenneth L. Graham, Wael Emara

Entropy Randomization in Machine Learning
Yuri S. Popkov, Alexey Yu. Popkov, Yuri A. Dubno

Introduction to Machine Learning with Applications in Information Security, Second Edition
Mark Stamp

The Pragmatic Programmer for Machine Learning: Engineering Analytics and Data Science Solutions
Marco Scutari and Mauro Malvestio

For more information on this series please visit: https://www.routledge.com/Chapman--HallCRC-Machine-Learning--Pattern-Recognition/book-series/CRCMACLEAPAT

The Pragmatic Programmer for Machine Learning

Engineering Analytics and Data Science Solutions

Marco Scutari
Mauro Malvestio

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
A CHAPMAN & HALL BOOK

Cover design by Carlo Sandonà

First edition published 2023
by CRC Press
6000 Broken Sound Parkway NW, Suite 300, Boca Raton, FL 33487-2742

and by CRC Press
4 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

CRC Press is an imprint of Taylor & Francis Group, LLC

Library of Congress Cataloging-in-Publication Data

Names: Scutari, Marco, author. | Malvestio, Mauro, author.
Title: The pragmatic programmer for machine learning : engineering analytics and data science solutions / Marco Scutari, Mauro Malvestio.
Description: Boca Raton : CRC Press, [2023] | Series: Machine learning & pattern recognition | Includes bibliographical references and index. | Summary: "The book provides a guide for data scientists and data analysts on modern software engineering to plan, implement, and improve the use of machine learning models. The book will also be useful to academics working in quantitative subjects where reliable software is required to obtain reproducible results and to increase the impact of research. It includes best practices on how best to implement models into code, how to document and troubleshoot code to ensure correctness, and how to deploy code into production. The book also explores how best practices are put to use in computing environments, using real-world examples and two case studies"-- Provided by publisher.
Identifiers: LCCN 2022043088 (print) | LCCN 2022043089 (ebook) | ISBN 9780367263508 (hardback) | ISBN 9780367255060 (paperback) | ISBN 9780429292835 (ebook)
Subjects: LCSH: Machine learning. | Software engineering. | Statistics--Data processing.
Classification: LCC Q325.5 .S38 2023 (print) | LCC Q325.5 (ebook) | DDC 006.3/1--dc23/eng20230111
LC record available at https://lccn.loc.gov/2022043088
LC ebook record available at https://lccn.loc.gov/2022043089

ISBN: 978-0-367-26350-8 (hbk)
ISBN: 978-0-367-25506-0 (pbk)
ISBN: 978-0-429-29283-5 (ebk)

DOI: 10.1201/9780429292835

Typeset in Merriweather
by KnowledgeWorks Global Ltd.

Publisher's note: This book has been prepared from camera-ready copy provided by the authors.

To my Oxford M.Sc. students: this is the book I would have liked to have to teach you Statistical Programming. Sorry that it took me a while to write it!

To my childhood best friend, the unbeatable C64.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Contents

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Preface

Pitching new ideas by prefacing them with quotes like "Data scientist: the sexiest job of the 21st century" [142] or "Data is the new oil" [357] has become such a cliché that any audience (in business and academia alike) will collectively roll their eyes in exasperation. And for good reason. Likewise, we do not believe radiologists or lorry drivers will be replaced by artificial intelligence and out of a job for the foreseeable future, and we are not alone in realising the limits of machine learning [358].

Even so, it is difficult to understate the impact that machine learning is having on many aspects of our lives. It has taken the pre-existing trends of using data and analytics (under the banner of "data mining", "big data" and similar buzzwords) to inform business decisions and drive scientific discovery, and made them ubiquitous. Machine learning has combined the mathematical rigour of information theory and statistics, the computational aspects of computer science and the goal-driven flexibility of optimisation theory, redefining how we work with data.

The flip side of trying to distil parts of so many different disciplines has been the clash between their respective cultures, which has been well summarised by Leo Breiman in "Statistical Modeling: The Two Cultures" [54]. On top of that, there is a tension between machine learning practice in the industry and academia: the latter strongly values producing novel models and theoretical results, while the former is driven by the need to produce practical results that have business value. With

so many different perspectives, it is a wonder that a rough consensus on what machine learning is has actually evolved! (Personally, our red line is conflating deep learning with machine learning. There is life beyond deep neural networks!)

In this melting pot of ideas, we feel that software engineering has played a remarkably small role compared to other disciplines. Machine learning, after all, is "a technique that allows computer systems to improve with experience and data" [122]. Therefore, there is a presumption that one will interact with a computer system, which in turn happens by engineering a piece of software that communicates to the computer system what it is supposed to do. The quality of this engineering is crucial in both academia and industry. In academia, software quality issues are one of the underlying causes of the "reproducibility crisis" [234, 340]. In industry, poor engineering leads to lower practical and computational performance [177], to a quick accumulation of technical debt [313] and sometimes to catastrophic failures with costs in the millions [317, 319, 368, 394]. There is, of course, a sizeable body of accumulated wisdom on how to architect and write software in foundational books like *The Pragmatic Programmer* [369] and *A Philosophy of Software Design* [252]. However, these books are written with business software in mind, and we find that they do not capture or touch only tangentially on key practices that go a long way towards successfully implementing and deploying machine learning models. Analysis of algorithms; matching data and algorithms with appropriate hardware; embracing data as part of the software; testing and documenting algorithms and their implementations; modularising and building pipelines; and, last but not least, naming variables. From our experience in academia and in the industry, engineering software and teaching software engineering to students and new staff alike, these topics are often not given

the importance they deserve. We hope to convince the readers of this book that the viability of any software that analyses data, whether you call it machine learning, data science or business analytics, depends crucially on putting careful thought into these engineering practices. We do not aim to be prescriptive: the individual practices that we discuss will be more or less relevant in different settings, and can be implemented with a variety of software tools. On the contrary, we want our readers to think about what we wrote in the context of their own experience and to figure out which parts apply and which do not!

The book starts with a brief introduction to machine learning and software engineering, to set out how we view them and how we think that they should interact in practical applications. The remainder is structured in four parts, from foundational to practical:

1. **Foundations of Scientific Computing:** covering key topics that are foundational for the *planning*, *analysis* and *design* of machine learning software, such as: the trade-offs of using different hardware configurations; the characteristics of different data types and of suitable data structures; and the analysis of algorithms to determine their computational complexity.
2. **Best Practices for Machine Learning and Data Science:** revisiting best practices in software engineering from the point of view of a machine learning engineer, from *writing*, *troubleshooting* and *deploying code* to production (that is, serving models) to *writing technical documentation*.
3. **Tools and Technologies:** discussing *broad classes of tools* that shape how we think about what is feasible to do with machine learning pipelines, with examples from the state of the art and the trade-offs they make.

4. **A Case Study:** putting the recommendations in the previous chapters into practice by *discussing and prototyping a machine learning pipeline* for natural language understanding from the work of Lipizzi et al. [195].

All the material in this book, including the book itself, is available online at

https://ppml.dev

and will be updated to fix assorted typos and code problems as they become known to us.

Finally, we would like to thank all the people who supported us and made this book possible. First of all, our families who put up with our long working hours. The colleagues who gave us feedback on early drafts of the book: Vincenzo Manzoni, Fabio Stella and Ron Kenett. And, last but not least, our editor Randi Cohen who bore with us through the many delays this book suffered during the Covid pandemic.

Lugano, Switzerland — *Marco Scutari*
Milano, Italy — *Mauro Malvestio*
August 2022

1

What Is This Book About?

The modern practice of data analysis is shaped by the convergence of many disciplines, each with its own history: information theory, computer science, optimisation, probability and statistics among them. Machine learning and data science can be considered their latest incarnations, inheriting the mantle of what used to be called "data analytics". Software engineering should be considered as a crucial addition to this list. Why do we need it to implement modern data analysis efficiently and effectively?

1.1 Machine Learning

There are many definitions of machine learning. Broadly speaking, it is a discipline that aims to create computer systems and algorithms that can learn a structured representation of reality without (or with less) human supervision in order to interact with it [305]. At one end of the spectrum, we can take this to be a narrow version of artificial general intelligence in which we want our computer systems to learn intellectual tasks independently and to generalise them to new problems, much like a human being would. At the other end, we can view machine learning as the ability to learn probabilistic models that provide a simplified representation of a specific phenomenon to perform a specific task [110] such as predicting an outcome of interest (supervised learning)

or finding meaningful patterns in the data (unsupervised learning). Somewhere in between these two extremes lie expert systems [60], which "capture the ability to think and reason about as an expert would in a particular domain" and can provide "a meaningful answer to a less than fully specified question."

Broadly speaking, in order to do this:

1. We need a working model of the world that describes the task and its context in a way that a computer can understand.
2. We need a goal: how do we measure the performance of the model? Because that is what we optimise for! Usually, it is the ability to predict new events.
3. We encode our knowledge of the world, drawing information from training data, experts or both.
4. The computer system uses the model as a proxy of reality and, as new inputs come in, to perform inference and decide if and how to perform the assigned task.

The exact form these elements take will depend on the domain we are trying to represent and on the model we will use to represent it. Machine learning is, at its core, a collection of models and algorithms from optimisation, statistics, probability and information theory that deal with abstract problems: from simple linear regression models [398], to Bayesian networks [314], to more complex models such as deep neural networks [122] and Gaussian processes [288]. These algorithms can be applied to a variety of domains from healthcare [381] to natural language processing [2] and computer vision [393], with some combinations of algorithms and domains working out better than others.

In classical statistics (Figure 1.1, bottom right), analysing data required the modeller to specify the probabilistic

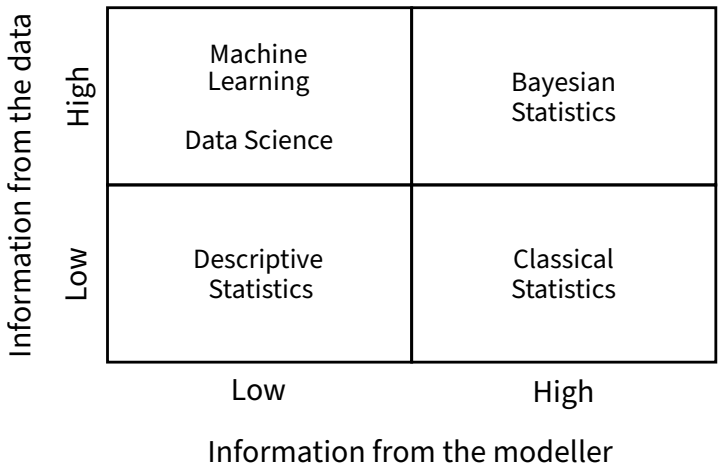

FIGURE 1.1 Different approaches to data analysis grouped by their sources of information: the data or the assumptions made by the modeller.

model generating them in order to draw inferences from a limited number of data points. Such models would necessarily have a simple structure for two reasons: because the modeller had to manually interpret their properties and their output, and because of the lack of any substantial computing power to estimate their parameters. This approach would put all the burden on the modeller: most of the utility that could be had from the model would come from the ability of the modeller to distil whatever he was modelling into simple mathematics and to incorporate any available prior information into the model structure. The result is the emphasis on closed-form results, low-order approximations and asymptotics that characterises the earlier part of modern statistics.

There are, however, many phenomena that cannot be feasibly studied in this fashion. Firstly, there are limits to a human modeller's ability to encode complex behaviour when manually structuring models. These

limits can easily be exceeded by phenomena involving large numbers of variables or by non-linear patterns of interactions between variables that are not very regular or known in advance. Secondly, there may not be enough information available to even attempt to structure a probabilistic model. Thirdly, limiting our choice of models to those that can be written in closed form to allow the modeller to fit, interpret and use them manually, without a significant use of computing power, does not necessarily ensure that those models are easy to interpret. For instance, there are many documented pitfalls in interpreting logistic regression [226, 287], which is arguably the simplest way to implement classification.

Classical applications of Bayesian statistics (Figure 1.1, top right) address some of these limitations. The modeller still has to structure a model covering both the data and any prior beliefs on their behaviour, but the posterior may be estimated algorithmically using Markov Chain Monte Carlo (MCMC).

In contrast [54], algorithmic approaches shift the burden from the modeller to data collection and computer software (Figure 1.1, top left). The modeller's role in constructing the probabilistic model is limited, and is largely replaced by a computer system sifting through large amounts of data: hence the name "machine learning". The structure of the model is learned from the data, with few limitations in what it may look like. Neural networks and Gaussian processes are universal approximators, for instance. Almost all the information comes from the data, instead of being prior information that is mediated by the modeller, which is why machine learning approaches are data-hungry.

1.2 Data Science

Data science is similarly data-driven (Figure 1.1, top left), but focuses on extracting insights from raw data

and presenting them graphically to support principled decision making. Kenett and Redman [179] describe it as follows: "the real work of data scientists involves helping people make better decisions on the important issues in the near term and building stronger organizations in the long term". It requires strong involvement from the data scientist in all areas of business, shifting the focus from computer systems to people. Nevertheless, data scientists use statistical and machine learning models as the means to obtain those insights.

Compared to classical statistics, when data are abundant (Big Data! [178]) we do not really need to construct their generating process from prior knowledge. The data contain enough information for us to "let them speak for themselves" and obtain useful insights, which are what we are mainly interested in. Of course, prior information from experts is still useful: models that incorporate it tend to be better at producing insights that can be acted upon.

As a result, data science puts a strong focus on the quality of the data, which is often problematic when dealing with data aggregated from multiple sources (data fusion) or with non-tabular data (natural language processing and computer vision). Often, data are poorly defined, simply wrong or ultimately irrelevant for the purpose they were collected for. Expert knowledge is crucial to assess them, to integrate them and to fix them if possible. Machine learning is widely applied to both text and images as well, but focused mostly on modelling their hidden structure until recently, when explainability became a hot topic [see, for instance, 191, 325].

Computer systems are key to data science, albeit with a different role than in machine learning. Storing and accessing large amounts of data, exploring them interactively, building the software pipelines that analyse them, handling the resulting spiky workloads: these are

all tasks that require a sophisticated use of both hardware and software.

1.3 Software Engineering

Software engineering is the systematic application of sound engineering principles to all phases of the software life cycle: design, development, maintenance, testing and evaluation [385]. Its central tenet is mastering the complexity inherent to developing large pieces of software that are reliable and efficient; that are usable and can be evolved over time; and that can be developed and maintained in a viable way in terms of both cost and effort [252].

Early definitions of software engineering suggested that we should treat it as if it were a traditional engineering discipline like, say, civil engineering. The result is the *waterfall model* [300], which lays out software development as a sequence of steps starting from collecting requirements and finishing with the deployment of the finished product. Modern practices recognise, however, that this model is flawed in several ways. Firstly, civil engineering arises from and is bound by the laws of physics, whereas we make up our own world with its own rules when we develop software. These rules will change over time as our understanding of the problem space evolves; the laws of physics do not. Secondly, the task the software is meant to perform will change over time, and our working definition of that task will change as well. Civil engineering mostly deals with well-defined problems that stay well-defined for the duration of the project. Finally, modifying a large building after its construction is completed is very difficult, but we routinely do that with software. Most

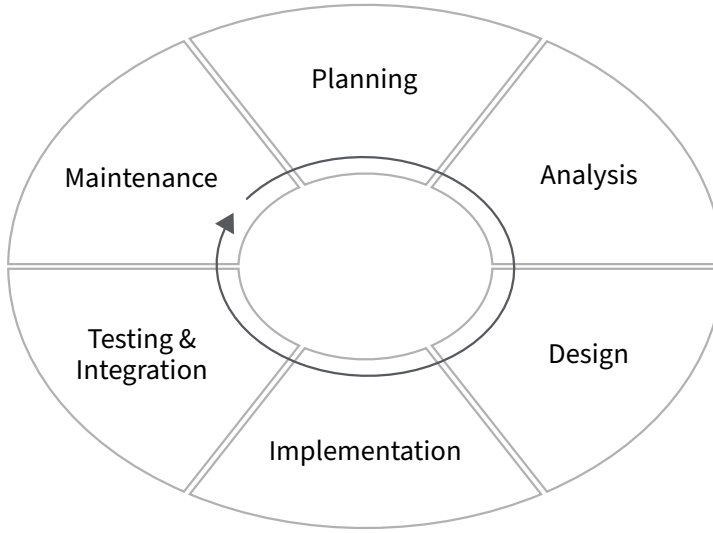

FIGURE 1.2 A schematic view of the phases of the software development life-cycle.

of the overall effort in the software lifetime is usually in maintaining and evolving it.

Current software engineering practices take the opposite view that software development is an open-ended ("software is never done"), iterative (the "software life-cycle") process: this is the core of the "Agile Manifesto" [36]. At a high level, it is organised as shown in Figure 1.2: a perpetual cycle of planning, analysis, design, implementation, testing and maintenance. The design of the software is heavily influenced by the domain it operates in [domain-driven development, 96]. It uses tests [test-driven development, 35], refactoring [105] and continuous integration [86] to incorporate new features, to fix bugs in a timely manner and to keep the code "in shape". Admittedly, all of these approaches have been touted as silver bullets to the point they have become buzzwords, and their practical implementation has often distorted them to the point of making software development worse. However, the key ideas of agile

have merit, and we will discuss and apply them in moderation in this book. They are well suited to structure the development of machine learning pipelines, which are built on a combination of mutable models and input data.

1.4 How Do They Go Together?

The centrality of computing in machine learning and data science makes software engineering practices essential in modern data analysis: most of the work is done by computer systems, which are powered by software.[1] Encoding the data, storing and retrieving them efficiently, implementing machine learning models, tying them together and with other systems: each of these tasks is complex enough that only sound engineering practices can ensure the overall correctness of what we are doing. This is true, in different ways, for both academic research and industry applications. As Kenett and Redman [179] put it, using a car analogy:

> "If data is the new oil, technology is the new engine. The engine powers the car and, without technological advancements, a data- and analytics-led transformation would not be possible. Technologies include databases, communications systems and protocols, applications that support the storage and processing of data, and the raw computing horsepower (much of it now in the cloud) to drive it all."

In academia, there is a widespread belief that the software implementations of novel methods can be treated as

[1] This is not to discount the role of hardware, just to set the focus of the book. Processing units tailored to machine learning use are an active research and engineering field as exemplified by Nvidia [244], Google [59] and other companies [295].

"one-off scripts". "We only need to run it once to write this paper, there is no point in refactoring and re-engineering it." is a depressingly common sentiment. As is not sharing code to "stay ahead of the competition". However, research and application papers using machine learning rely crucially on the quality of the software they use because:

1. The models themselves are often black boxes whose mathematical behaviour is not completely understood (Section 9.2).
2. The data are complex enough that even experts in the domains they come from struggle to completely explain them (Section 9.1).

If we do not understand both the data and the models completely, it becomes very difficult to spot problems in the software we use to work on them: unexpected behaviour arising from software bugs may be mistaken for a peculiarity in either of them. It is then crucial that we minimise the chances of this happening by applying all the best engineering practices we have at our disposal. Present and past failures to do so have led to a widespread "reproducibility crisis" in fields as diverse as drug research [272, 20–25% reproducible], comparative psychology [332, 36% reproducible], finance [61, 43% reproducible] and computational neuroscience [224, only 12% of papers provide both data and code]. Machine learning and artificial intelligence research is in a similarly sorry state: that "when the original authors provided assistance to the reproducers, 85% of results were successfully reproduced, compared to 4% when the authors didn't respond" [261] *does* suggest that there is margin for improvement. Fortunately, in recent years scientists have widely accepted this is a problem [234], and the machine learning community has reached some consensus on how to tackle it [340].

In industry, poor engineering leads to lower practical and computational performance and a quick accumulation of technical debt [313, and Section 5.2]. Badly engineered data may not contain the information we are looking for in a usable form; models that are not well packaged may be slow to deploy and difficult to roll back; data may contain biases or may change over time in ways that make models fail silently; or the machine learning software may become an inscrutable black box whose outputs are impossible to explain, making troubleshooting impossible.

To conclude, we believe that solid machine learning applications and research rest on three pillars:

1. The foundations of machine learning (mathematics, probability, computer science), which provide guarantees that the models work.
2. Software engineering, which provides guarantees that the implementations of the models work (effectively and efficiently).
3. The quality of the data in terms of features, size, fairness, and in how they were collected.

In this book, we will concentrate on the software engineering aspect, touching briefly on some aspects of the data. We will not discuss the theoretical or methodological aspects of machine learning, which are better covered in the huge amount of specialised literature published to date [such as 109, 122, 147, 288, 305, and many others].

References

Abid A , Abdalla A , Ali A , Khan D , Alfozan A , Zou J (2022). Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild. https://www.gradio.app/docs/.

Aggarwal CC (2018). Machine Learning for Text. Springer.

Alam S , Bălan L , Chan NL , Comym G , Dada Y , Danov I , Hoang L , Kanchwala R , Klein J , Milne A , Schwarzmann J , Theisen M , Wong S (2022). Kedro. https://github.com/kedro-org/kedro.

Alnæs MS , Project Jupyter (2022). nbdime – Diffing and Merging of Jupyter Notebooks. https://nbdime.readthedocs.io.

Alquraan A , Takruri H , Alfatafta M , Al-Kiswany S (2018). An Analysis of Network-Partitioning Failures in Cloud Systems. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)* , pp. 51–68.

Altair (2022). *Altair: Declarative Visualization in Python*. https://altair-viz.github.io/.

Amazon (2021). *Dynamic A/B Testing for Machine Learning Models with Amazon SageMaker MLOps Projects*. https://aws.amazon.com/blogs/machine-learning/dynamic-a-b-testing-for-machine-learning-models-with-amazon-sagemaker-mlops-projects/.

Amazon (2022a). *Amazon Redshift Documentation*. https://docs.aws.amazon.com/redshift/index.html.

Amazon (2022b). *Amazon SageMaker Examples*. https://sagemaker-examples.readthedocs.io/en/latest/sagemaker_model_monitor/index.html.

Amazon (2022a). *AWS Cloud9 Documentation*. https://docs.aws.amazon.com/cloud9.

Amazon (2022b). *Machine Learning: Amazon Sagemaker*. https://aws.amazon.com/sagemaker/.

Amazon Web Services (2022a). *Amazon Elastic Kubernetes Service Documentation*. https://docs.aws.amazon.com/eks.

Amazon Web Services (2022b). *Amazon Machine Images (AMI)*. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html.

Amazon Web Services (2022c). *AWS Trainium*. https://aws.amazon.com/machine-learning/trainium/.

Anaconda (2022a). *Conda for Data Scientists*. https://docs.conda.io/projects/conda/en/latest/user-guide/concepts/data-science.html.

Anaconda (2022b). *Package, Dependency and Environment Management for Any Language*. https://docs.conda.io.

Anderson E , Bai Z , Bishof C , Blackford S , Demmel J , Dongarra J , Du Croz J , Greenbaum A , Hammarling S , McKenney A , Sorensen D (1999). LAPACK Users' Guide. 3rd edition. SIAM.

Ansible Project (2022). *Ansible Documentation*. https://docs.ansible.com/ansible/latest/index.html.

Apache Software Foundation (2022a). *Celery Executor*. https://airflow.apache.org/docs/apache-airflow/stable/executor/celery.html.

Apache Software Foundation (2022b). *Impala Documentation*. https://impala.apache.org/impala-docs.html.

Apple (2022). *TensorFlow 2 Conversion*. https://coremltools.readme.io/docs/tensorflow-2.

Aquasecurity (2022). *Trivy Documentation*. https://aquasecurity.github.io/trivy/.

Argo Project (2022). *Argo Workflow Documentation*. https://argoproj.github.io/argo-workflows.

Arisholm E , Gallis H , Dybå T , Sjøberg DIK (2007). Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise. IEEE Transactions on Software Engineering, 33 (2), 5–86.

Arpteg A , Brinne B , Crnkovic-Friis L , Bosch J (2018). Software Engineering Challenges of Deep Learning. In *Euromicro Conference on Software Engineering and Advanced Applications* , pp. 50–59. IEEE.

ArXiv (2022). *arXiv API Access*. https://arxiv.org/help/api.

Atom (2022). *A hackable text editor for the 21st Century*. https://atom.io/.

Ayer A (2022). git-crypt: Transparent File Encryption in Git. https://github.com/AGWA/git-crypt.

Batchelder N , et al (2022). A Static Type Analyzer for Python Code. https://google.github.io/pytype.

Bates D , Maechler M (2021). Matrix: Sparse and Dense Matrix Classes and Methods. https://cran.r-project.org/web/packages/Matrix/.

BBC (2018). *Amazon Scrapped 'Sexist AI' Tool*. https://www.bbc.com/news/technology-45809919.

BBC (2021a). *Facebook Apology as AI Labels Black Men 'Primates'*. https://www.bbc.com/news/technology-58462511.

BBC (2021b). *Twitter Finds Racial Bias in Image-Cropping AI*. https://www.bbc.com/news/technology-57192898.

Beam AL , Manrai AK , Ghassemi M (2020). Challenges to the Reproducibility of Machine Learning Models in Health Care. Journal of the American Medical Association, 323 (4), 305–306.

Beck K (2002). Test-Driven Development by Example. Addison-Wesley.

Beck K , Beedle M , Van Bennekum A , Cockburn A , Cunningham W , Fowler M , Grenning J , Highsmith J , Hunt A , Jeffries R , Kern J (2001). The Agile Manifesto. https://www.agilealliance.org/wp-content/uploads/2019/09/agile-manifesto-download-2019.pdf.

BentoML (2022). *Unified Model Serving Framework*. https://docs.bentoml.org/en/latest/.

Bezanson J , Karpinski S , Shah VB , et al (2022). Style Guide: The Julia Language. https://docs.julialang.org/en/v1/manual/style-guide/index.html.

Bhupinder K , Dugré M , Hanna A , Glatard T (2021). An Analysis of Security Vulnerabilities in Container Images for Scientific Data Analysis. GigaScience, 10 (6), giab025.

Bishop CM (1995). Training with Noise is Equivalent to Tikhonov Regularization. Neural Computation, 7 (1), 108–116.

Blackford LS , Demmel J , Dongarra J , Duff I , Hammarling S , Henry G , Heroux, Kaufman L , Lumsdaine A , Petitet A , Pozo R , Remington K , Whaley RC (2002). An Updated Set of Basic Linear Algebra Subprograms (BLAS). ACM Transactions on Mathematical Software, 28 (2), 135–151.

Blagotic A , Valle-Jones D , Breen J , Lundborg J , White JM , Bode J , White K , Mueller K , Redaelli M , Lorang N , Schalk P , Schneider D , Hepp G , Jamile Z (2021). ProjectTemplate: Automates the Creation of New Statistical Analysis Projects. https://cran.r-project.org/web/packages/ProjectTemplate/.

Blei DM , Kucukelbir A , McAuliffe JD (2017). Variational Inference: A Review for Statisticians. Journal of American Statistical Association, 112 (518), 859–877.

Blischak JD , Carbonetto P , Stephens M (2022). workflowr: A Framework for Reproducible and Collaborative Data Science. https://cran.r-project.org/web/packages/workflowr.

Bogner J , Verdecchia R , Gerostathopoulos I (2021). Characterizing Technical Debt and Antipatterns in AI-Based Systems: A Systematic Mapping Study. In *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)* , pp. 64–73.

Bokeh (2022). *Bokeh Documentation*. https://docs.bokeh.org/en/latest/.

Bonawitz K , Eichner H , Grieskamp W , Huba D , Ingerman A , Ivanov V , Kiddon C , Koneččný J , Mazzocchi S , McMahan HB , Van Overveldt T , Petrou D , Ramage D , Roselander J (2019). Towards Federated Learning at Scale: System Design. In *Proceedings of Machine Learning and Systems* , pp. 374–388.
Braiek HB , Khomh F (2020). On Testing Machine Learning Programs. Journal of Systems and Software, 164 , 110542.
Brandl G , the Sphinx Team (2022). Sphinx: Python Documentation Generator. https://www.sphinx-doc.org/en/master/.
Brass P (2008). Advanced Data Structures. Cambridge University Press.
Breck E , Cai S , Nielsen E , Salib M , Sculley D (2017). The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction. In *IEEE International Conference on Big Data* , pp. 1123–1132.
Breiman L (1996). Out-of-Bag Estimation. https://www.stat.berkeley.edu/pub/users/breiman/OOBestimation.pdf.
Breiman L (2001a). Random Forests. Machine Learning, 45 (1), 5–32.
Breiman L (2001b). Statistical Modeling: The Two Cultures. Statistical Science, 16 (3), 199–231.
Callon R (1996). The Twelve Networking Truths. https://rfc-editor.org/rfc/rfc1925.txt.
Canonical (2022a). *Cloud-Init Documentation*. https://cloudinit.readthedocs.io/en/latest.
Canonical (2022b). *MicroK8s Documentation*. https://microk8s.io/docs.
Carpenter B , Gelman A , Hoffman MD , Lee D , Goodrich B , Betancourt M , Brubaker M , Guo J , Li P , Riddell A (2017). Stan: A Probabilistic Programming Language. Journal of Statistical Software, 76 (1), 1–32.
Cass S (2019). Taking AI to the Edge: Google's TPU Now Comes in a Maker-Friendly Package. IEEE Spectrum, 56 (5), 16–17.
Castillo E , Gutiérrez JM , Hadi AS (1997). Expert Systems and Probabilistic Network Models. Springer.
Chang AC , Li P (2015). Is Economics Research Replicable? Sixty Published Papers from Thirteen Journals Say ‘Usually Not’. In Federal Reserve Board Finance and Economics Discussion Paper, p. 083.
Chang W , Cheng J , Allaire JJ , Sievert C , Schloerke B , Xie Y , Allen J , McPherson J , Dipert A , Borges B (2022). shiny: Web Application Framework for R. https://cran.r-project.org/web/packages/shiny.
Cheney J , Chiticariu L , Tan WC (209). Provenance in Databases: Why, How and Where. Foundations and Trends in Databases, 1 (4), 379–474.
Clements P , Bachmann F , Bass L , Garlan D , Ivers J , Little R , Merson P , Nord R , Stafford J (2011). Documenting Software Architectures: Views and Beyond. 2nd edition. Addison-Wesley.
Cloudera (2022). *Cloudera: The Hybrid Data Company*. https://www.cloudera.com/.
Cohen AGV , Pavlick E , Tellex S (2019). OpenGPT-2: We Replicated GPT-2 Because You Can Too. https://blog.usejournal.com/opengpt-2-we-replicated-gpt-2-because-you-can-too-45e34e6d36dc.
Comet (2022). *Comet Documentation*. https://www.comet.com/docs/v2.
Cormen TH (2013). Algorithms Unlocked. The MIT Press.
Cortes-Ortuno D , Laslett O , Kluyver T , Fauske V , Albert M , MinRK, Hovorka O , Fangohr H (2022). IPython Notebook Validation for py.test: Documentation. https://nbval.readthedocs.io.
CRAN Team (2022). *The Comprehensive R Archive Network*. https://cran.r-project.org/.
Crook J , Banasik J (2004). Does Reject Inference Really Improve the Performance of Application Scoring Models? Journal of Banking and Finance, 28 , 857–874.
Crosley T (2022). A Python Utility and Library to Sort Imports. https://pycqa.github.io/isort/.
Cunningham W (1992). The WyCash Portfolio Management System. In *Addendum to the Proceedings of ACM Object-Oriented Programming, Systems, Languages & Applications Conference* , pp. 29–30.
Cunningham W (2011). Ward Explains the Debt Metaphor. https://wiki.c2.com/?WardExplainsDebtMetaphor.
DagsHub (2022). *Welcome to the DagsHub Docs*. https://dagshub.com/docs/.
Databricks (2022). *Databricks Documentation*. https://docs.databricks.com/applications/machine-learning/index.html.
de Lima Salge CA , Berente N (2016). Pair Programming vs. Solo Programming: What Do We Know After 15 Years of Research? In *Proceedings of the Annual Hawaii International Conference on System Sciences* , pp. 5398–5406.
Devlin J , Chang MW , Lee K , Toutanova K (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NNACL-HLT)* , pp. 4171–4186.
Dimakopoulou M , Zhou Z , Athey S , Imbens G (2018). Estimation Considerations in Contextual Bandits. https://arxiv.org/abs/1711.07077.
Dimakopoulou M , Zhou Z , Athey S , Imbens G (2019). Balanced Linear Contextual Bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence* , pp. 3445–3453.
DMTF (2022). *Open Virtualization Format*. https://www.dmtf.org/standards/ovf.
Docker (2022). *Docker*. https://www.docker.com/.
Docker (2022). *Docker Registry HTTP API V2 Documentation*. https://docs.docker.com/registry/spec/api/.
Docker (2022). *Overview of Docker Compose*. https://docs.docker.com/compose.
Dusserre E , Padró M (2017). Bigger Does Not Mean Better! We Prefer Specificity. In *Proceedings of the 12th International Conference on Computational Semantics* , pp. 1–6.
Duvall PM , Matyas S , Glover A (2007). Continuous Integration: Improving Software Quality and Reducing Risk. Addison-Wesley.
Eclipse Che (2022). Run your favorite IDE on Kubernetes. https://www.eclipse.org/che/technology/.
Eclipse Foundation (2022a). *Desktop IDEs*. https://www.eclipse.org/ide/.
Eclipse Foundation (2022b). *Theia: Cloud & Desktop IDE*. https://theia-ide.org/docs/.
Edmundson A (2021). The Rise (and Lessons Learned) of ML Models to Personalize Content on Home. URL https://engineering.atspotify.com/2021/11/the-rise-and-lessons-learned-of-ml-models-to-personalize-content-on-home-part-i/, https://engineering.atspotify.com/2021/11/the-rise-and-lessons-learned-of-ml-models-to-personalize-content-on-home-part-ii/.
Elasticsearch (2022). *Free and Open Search: The Creators of Elasticsearch, ELK & Kibana*. https://www.elastic.co/.
Elementl (2022). *Dagster Documentation*. https://docs.dagster.io.
Espe L , Jindal A , Podolskiy V , Gerndt M (2020). Performance Evaluation of Container Runtimes. In *Proceedings of the 10th International Conference on Cloud Computing and Services Science* , pp. 273–281.
Espeholt L , Soyer H , Munos R , Simonyan K , Mnih V , Ward T , Doron Y , Firoiu V , Harley T , Dunning I , Legg S , Kavukcuoglu K (2018). IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. In *Proceedings of the 35th*

International Conference on Machine Learning (ICML) , pp. 1407–1416.
ETF OAuth Working Group (2022). *OAuth 2.0*. https://oauth.net/2/.
Evans E (2003). Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley.
Explosion (2021). *Spacy: Industrial-Strength Natural Language Processing*. https://spacy.io/.
Feast Authors (2022). *Feast Documentation*. https://docs.feast.dev/.
Fenniak M (2022). PyPDF2 Documentation. https://pypdf2.readthedocs.io/en/latest/.
Fernandez A , Garcia S , Herrera F , Chawla NV (2018). SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-Year Anniversary. Journal of Artificial Intelligence Research, 61 , 863–905.
Firke S , Denney B , Haid C , Knight R , Grosser M , Zadra J (2022). janitor: Simple Tools for Examining and Cleaning Dirty Data. https://cran.r-project.org/web/packages/janitor.
Formagrid (2022). *Airtable Is a Modern Spreadsheet Platform with Database Functionalities*. https://airtable.com.
Fortin P , Fleury A , Lemaire F , Monagan M (2021). High-Performance SIMD Modular Arithmetic for Polynomial Evaluation. Concurrency and Computation: Practice and Experience, 33 (16), e6270.
Fowler M (2003). UML Distilled. 3rd edition. Addison-Wesley.
Fowler M (2018). Refactoring: Improving the Design of Existing Code. 2nd edition. Addison-Wesley.
Galassi M , Davies J , Theiler J , Gough B , Jungman G , Alken P , Booth M , Rossi F , Ulerich R (2021). GNU Scientific Library. https://www.gnu.org/software/gsl/doc/latex/gsl-ref.pdf.
Gama J , Žliobaitė I , Bifet A , Pechenizkiy M , Bouchachia A (2014). A Survey on Concept Drift Adaptation. ACM Computing Surveys, 46 (4), 44.
Ganiev A , Chapin C , Andrade A , Liu C (2021). An Architecture for Accelerated Large-Scale Inference of Transformer-Based Language Models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics* , pp. 163–169.
Gelman A , Carlin B , Stern HS , Dunson DB , Vehtari A (2013). Bayesian Data Analysis. 3rd edition. CRC Press.
Ghahramani Z (2015). Probabilistic Machine Learning and Artificial Intelligence. Nature, 521 , 452–459.
GitHub (2022a). *GitHub Codespaces*. https://github.com/features/codespaces.
GitHub (2022b). *Storing Workflow Data as Artifacts*. https://docs.github.com/en/actions/using-workflows/storing-workflow-data-as-artifacts.
GitHub (2022c). *Working with the Container Registry*. https://docs.github.com/en/packages/working-with-a-github-packages-registry/working-with-the-container-registry.
GitLab (2022a). *GitLab Artifacts*. URL https://docs.gitlab.com/ee/ci/pipelines/job_artifacts.html, https://docs.gitlab.com/ee/ci/pipelines/pipeline_artifacts.html.
GitLab (2022b). *GitLab Container Registry*. https://docs.gitlab.com/ee/user/packages/container_registry/.
Gitlab (2022). *GitLab Runner Documentation*. https://docs.gitlab.com/runner/.
GitLab (2022). *Group Direction: MLOps*. https://about.gitlab.com/direction/modelops/mlops/.
Gitlab (2022). *What Is GitOps?* https://about.gitlab.com/topics/gitops.
Gitpod (2022). *Gitpod: Always Ready to Code*. https://www.gitpod.io.
GNU Project (2022). *GNU EMacs*. https://www.gnu.org/software/emacs/.
Gong M , Xie Y , Pan K , Feng K (2020). A Survey on Differentially Private Machine Learning. IEEE Computational Intelligence Magazine, 15 (2), 49–64.
Goodfellow I , Bengio Y , Courville A (2016). Deep Learning. MIT Press.
Goodfellow I , Pouget-Abadie J , Mirza M , Xu B , Warde-Farley D , Ozair S , Courville A , Bengio Y (2014). Generative Adversarial Nets. In Advances in Neural Information Processing Systems (NIPS), pp. 2672–2680.
Goodger D , van Rossum G (2022). PEP 257: Docstring Conventions. https://peps.python.org/pep-0257/#what-is-a-docstring].
Google (2022). *BigQuery Documentation*. https://cloud.google.com/bigquery/docs.
Google (2022). *Deep Learning Containers*. https://cloud.google.com/deep-learning-containers.
Google (2022). *Google Kubernetes Engine*. https://cloud.google.com/kubernetes-engine/docs.
Google (2022a). *Google Python Style Guide*. https://google.github.io/styleguide/pyguide.html.
Google (2022b). *repo: The Multiple Git Repository Tool*. https://github.com/GerritCodeReview/git-repo.
Google (2022). *Vertex AI Documentation*. https://cloud.google.com/vertex-ai/docs.
Google (2022). *Welcome to Colab!* https://colab.research.google.com.
Grafana Labs (2022). *Grafana Loki Documentation*. https://grafana.com/docs/loki/latest/.
GrafanaLabs (2022). *Grafana: The Open Observability Platform*. https://grafana.com/.
Greenfeld AR (2022). Cookiecutter Data Science. https://drivendata.github.io/cookiecutter-data-science/.
Gregg B (2021). Systems Performance: Enterprise and the Cloud. 2nd edition. Addison-Wesley.
Grotov K , Titov S , Sotnikov V , Golubev Y , Bryksin T (2022). A Large-Scale Comparison of Python Code in Jupyter Notebooks and Scripts. In *Proceedings of the 19th Working Conference on Mining Software Repositories* , pp. 1–12.
Groves RM , Fowler FJ , Couper MP , Lepkowski JM , Singer E , Tourangeau R (2009). Survey Methodology. Wiley.
Hammant P (2020). Trunk Based Development. https://trunkbaseddevelopment.com/.
Hao J , Anang TJ , Kim K (2021). An Empirical Analysis of VM Startup Times in Public IaaS Clouds: An Extended Report. In *Proceedings of the 14th IEEE International Conference on Cloud Computing* , pp. 398–403.
Harbor (2022). *Harbor Documentation*. https://goharbor.io/docs/.
Harris CR , Millman KJ , van der Walt SJ , Gommers R , Virtanen P , Cournapeau D , Wieser E , Taylor J , Berg S , Smith NJ , Kern R , Picus M , Hoyer S , van Kerkwijk MH , Brett M , Haldane A , Fernández del Río J , Wiebe M , Peterson P , Gérard-Marchant P , Sheppard K , Reddy T , Weckesser W , Abbasi H , Gohlke C , Oliphant TE (2020). Array Programming with NumPy. Nature, 585 (7285), 357–362.
Harvard Business Review (2012). *Data Scientist: The Sexiest Job of the 21st Century*. https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century.
HashiCorp (2022). *Packer Documentation*. https://www.packer.io/docs.
HashiCorp (2022a). *Terraform Documentation*. https://www.terraform.io/docs.

HashiCorp (2022b). *Terraform Registry*. https://registry.terraform.io.
HashiCorp (2022c). *Vagrant Documentation*. https://www.vagrantup.com/docs.
Hastie T , Tibshirani R , Friedman J (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd edition. Springer.
Hazelwood K , Bird S , Brooks D , Chintala S , Diril U , Dzhulgakov D , Fawzy M , Jia B , Jia Y , Kalro A , Law J , Lee K , Lu J , Noordhuis P , Smelyanskiy M , Xiong L , Wang X (2018). Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)* , pp. 620–629.
He X , Zhao K , Chu X (2021). AutoML: A Survey of the State-of-the-Art. Knowledge-Based Systems, 212 , 106622.
Hester J , Angly F , Hyde R , Chirico M , Ren K , Rosenstock A (2022). A Linter for R Code. https://cran.r-project.org/web/packages/lintr.
Holoviz (2022). *Panel User Guide*. https://panel.holoviz.org/user_guide/index.html.
Hopsworks (2022). *Hopsworks Documentation*. https://docs.hopsworks.a.
Humble J , Farley D (2011). Continuous Delivery. Addison Wesley.
Hunt E (2016). Tay, Microsoft's AI Chatbot, Gets a Crash Course in Racism from Twitter.
https://www.theguardian.com/technology/2016/mar/24/tay-microsofts-ai-chatbot-gets-a-crash-course-in-racism-from-twitter.
Hunter JD (2022). Matplotlib API Reference. https://matplotlib.org/stable/api/index.
Intel (2021). *Intel oneAPI Math Kernel Library*.
https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/onemkl.html.
Iterative (2022). *CML Documentation*. https://cml.dev/doc.
Iterative (2022). *DVC: Data Version Control. Git for Data & Models*. https://github.com/iterative/dvc.
Iterative (2022a). *DVC Python API*. https://dvc.org/doc/api-reference.
Iterative (2022b). *MLEM Documentation*. https://mlem.ai/doc.
Jacek C , Greiler M , Bird C , Panjer L , Coatta T (2018). CodeFlow: Improving the Code Review Process at Microsoft. ACM Queue, 6 (5), 1–20.
Jain P , Mo X , Jain A , Subbaraj H , Durrani R , Tumanov A , Gonzalez J , Stoica I (2018). Dynamic Space-Time Scheduling for GPU Inference. In *Workshop on Systems for ML and Open Source Software, NeurIPS 2018* , pp. 1–9.
Jenkins (2022a). *A Command Line Tool to Run Jenkinsfile as a Function*. https://github.com/jenkinsci/jenkinsfile-runner.
Jenkins (2022b). *Jenkins User Documentation*. https://www.jenkins.io/doc/.
JetBrains (2022a). *IntelliJ IDEA*. https://www.jetbrains.com/idea/.
JetBrains (2022b). *PyCharm*. https://www.jetbrains.com/pycharm/.
Jouppi NP , Yoon DH , Kurian G , Li S , Patil N , Laudon J , Young C , Patterson D (2020). A Domain-Specific Supercomputer for Training Deep Neural Networks. Communications of the ACM, 63 (7), 67–78.
Jouppi NP , Young C , Patil N , Patterson D (2018). A Domain-Specific Architexture for Deep Neural Networks. Communications of the ACM, 61 (9), 50–59.
Julia VS Code (2022a). *An Implementation of the Microsoft Language Server Protocol for the Julia Language*.
https://juliapackages.com/p/languageserver.
Julia VS Code (2022b). *Julia for Visual Studio Code*. https://www.julia-vscode.org.
JuliaLang (2022). *Pkg: Package Manager for the Julia Programming Language*. https://pkgdocs.julialang.org/v1/.
Kaji N , Kobayashi H (2017). Incremental Skip-gram Model with Negative Sampling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* , pp. 363–371.
Kalnytskyi I (2022a). Poetry Documentation. https://python-poetry.org/docs.
Kalnytskyi I (2022b). sphinxcontrib-openapi Is a Sphinx Extension to Generate APIs Docs from OpenAPI. https://sphinxcontrib-openapi.readthedocs.io.
Kalnytskyi I (2022c). The Sphinx Extension that Renders OpenAPI Specs Using ReDoc. https://sphinxcontrib-redoc.readthedocs.io/en/stable.
Kanagawa M , Hennig P , Sejdinovic D , Sriperumbudur BK (2018). Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences. https://arxiv.org/abs/1807.02582.
Kang S , Jin R , Deng X , Kenett RS (2021). Challenges of Modeling and Analysis in Cybermanufacturing: A Review from a Machine Learning and Computation Perspective. *Journal of Intelligent Manufacturing*, Online first .
Katal A , Wazid M , Goudar RH (2013). Big Data: Issues, Challenges, Tools and Good Practices. In *Proceedings of the International Conference on Contemporary Computing* , pp. 404–409.
Kenett RS , Redman TC (2019). The Real Work of Data Science. Wiley.
Kernigham BW , Pike R (1999). The Practice of Programming. Addison-Wesley.
Khan WZ , Ahmed E , Hakak S , Yaqoob I , Ahmed A (2019). Edge Computing: A Survey. Future Generation Computer Systems, 97 , 219–235.
Kibirige H (2022). Plotnine API Reference. https://plotnine.readthedocs.io/en/stable/api.html.
Knuth DE (1976). Big Omicron and Big Omega and Big Theta. ACM Sigact News, 8 (2), 18–24.
Knuth DE (1997). The Art of Computer Programming, Volume 1: Fundamental Algorithms. 3rd edition. Addison-Wesley.
Krämer S (2022). Julia Autodoc. https://bastikr.github.io/sphinx-julia/juliaautodoc.html#julia-autodoc].
Kriasoft (2016). *Folder Structure Conventions*. https://github.com/kriasoft/Folder-Structure-Conventions.
Kuhn DR , Kacker RN , Lei Y (2013). Introduction to Combinatorial Testing. CRC Press.
Kuhn M , Johnson K (2013). Applied Predictive Modeling. Springer.
Lai R , Ren K (2022). An Implementation of the Language Server Protocol for R. https://cran.r-project.org/web/packages/languageserver.
Langa Ł , et al (2022). Black: The Uncompromising Code Formatter. https://black.readthedocs.io/en/stable/.
Li J , Chen X , Hovy E , Jurafsky D (2016). Visualizing and Understanding Neural Models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* , pp. 681–691. Association for Computational Linguistics.
Li Q , Wen Z , Wu Z , Hu S , Wang N , Li Y , Liu X , He B (2021). A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection. IEEE Transactions on Knowledge and Data Engineering, Advance publication.

Linardatos P , Papastefanopoulos V , Kotsiantis S (2021). Explainable AI: A Review of Machine Learning Interpretability Methods. Entropy, 23 (1), 18.
Linux Kernel Organization (2022). *The Linux Kernel Archives*. https://kernel.org/.
Lipizzi C , Behrooz H , Dressman M , Vishwakumar AG , Batra K (2022). Acquisition Research: Creating Synergy for Informed Change. In *Proceedings of the 19th Annual Acquisition Research Symposium* , pp. 242–255.
Lipizzi C , Borrelli D , de Oliveira Capela F (2021). A Computational Model Implementing Subjectivity with the 'Room Theory': The case of Detecting Emotion from Text. https://arxiv.org/abs/2005.06059.
Logilab and PyCQA and contributors (2022). *Pylint is a Static Code Analyser for Python 2 or 3*. https://pylint.pycqa.org/en/latest/.
Lohr SL (2021). Sampling: Design and Analysis. 3rd edition. CRC Press.
Lopes CV (2020). Exercises in Programming Style. CRC Press.
Loria S , et al (2022). A Pluggable API Specification Generator. https://apispec.readthedocs.io/en/latest.
Lundberg SM , Lee SI (2017). A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems (NIPS)* , pp. 4765–4774.
Lyttle I , Jeppson H , Altair Developers (2022). altair: Interface to Altair. https://cran.r-project.org/web/packages/altair.
Manohar A (2022). asdf Documentation. https://asdf-vm.com/guide/getting-started.html.
Marin JM , Robert CP (2014). Bayesian Essentials with R. 2nd edition. Springer.
Martin RC (2008). Clean Code. Prentice Hall.
McConnell S (2004). Code Complete. 2nd edition. Microsoft Press.
McKinney W (2017). Python for Data Analysis. 2nd edition. O'Reilly.
Mehrabi N , Morstatter F , Saxena N , Lerman K , Galstyan A (2021). A Survey on Bias and Fairness in Machine Learning. ACM Computing Surveys, 54 (6), 115.
Melançon G , Dutour I , Bousquet-Mélou M (2001). Random Generation of Directed Acyclic Graphs. Electronic Notes in Discrete Mathematics, 10 , 202–207.
Meta Platforms (2022a). *A Performant Type-Checker for Python 3*. https://pyre-check.org.
Meta Platforms (2022b). *React: A JavaScript Library for Building User Interfaces*. https://reactjs.org/.
Microsoft (2022). *A performant, Feature-Rich Language Server for Python in VS Code*. https://marketplace.visualstudio.com/items?itemName=ms-python.vscode-pylance.
Microsoft (2022). *Azure Kubernetes Service (AKS)*. https://docs.microsoft.com/en-gb/azure/aks.
Microsoft (2022a). *Azure Machine Learning*. https://azure.microsoft.com/en-us/services/machine-learning/.
Microsoft (2022b). *Code editing. Redefined.* https://code.visualstudio.com/.
Microsoft (2022c). *Data Visualization: Microsoft PowerBI*. https://powerbi.microsoft.com/.
Microsoft (2022d). *Language Server Protocol*. https://microsoft.github.io/language-server-protocol.
Microsoft (2022e). *Shadow Testing*. https://microsoft.github.io/code-with-engineering-playbook/automated-testing/shadow-testing/.
Microsoft (2022f). *Virtualization Documentation*. https://docs.microsoft.com/en-us/virtualization/.
Microsoft (2022g). *Visual Studio Code: Code Editing, Redefined*. https://code.visualstudio.com/.
Microsoft (2022h). *VS Code in the Web*. https://vscode.dev.
Microsoft Research Cambridge (2022). *Project InnerEye–Democratizing Medical Imaging AI*. URL https://www.microsoft.com/en-us/research/project/medical-image-analysis/, https://www.microsoft.com/en-us/research/video/five-minute-overview-innereye-research-project/.
MinIO (2022). *MinIO Documentation*. https://docs.min.io/docs.
Miłkowski M , Hensel WM , Hohol M (2018). Replicability or Reproducibility? On the Replication Crisis in Computational Neuroscience and Sharing Only Relevant Detail. Journal of Computational Neuroscience, 45 , 163–172.
Montgomery DC (20). Design and Analysis of Experiments. 10th edition. Wiley.
Mood C (2010). Logistic Regression: Why We Cannot Do What We Think We Can Do, and What We Can Do About It. European Sociological Review, 26 (1), 67–82.
Mujtaba H (2018). Samsung Powers NVIDIA Quadro RTX Graphics Cards With 16Gb GDDR6 Memory. https://wccftech.com/nvidia-quadro-rtx-turing-gpu-samsung-gddr6-memory/.
Müller K , Walthert L (2022a). Non-Invasive Pretty Printing of R Code. https://cran.r-project.org/web/packages/styler.
Müller K , Walthert L (2022b). Third-Party Integrations. https://styler.r-lib.org/articles/third-party-integrations.html.
Muth C , Oravecz Z , Gabry J (2018). User-Friendly Bayesian Regression Modeling: A Tutorial with rstanarm and shinystan. The Quantitative Methods for Psychology, 14 (2), 99–119.
Myers GJ , Badgett T , Sandler C (2012). The Art of Software Testing. 3rd edition. Wiley.
Narayanan A , Shmatikov V (2008). Robust De-Anonymization of Large Sparse Datasets. In *Proceedings of the IEEE Symposium on Security and Privacy* , pp. 111–125.
Natekin A , Knoll A (2013). Gradient Boosting Machines, a Tutorial. Frontiers in Neurorobotics, 7 (21), 1–21.
Nature (2016). Reality Check on Reproducibility. Nature, 533 (437).
nbQA Team (2022). *Run isort, pyupgrade, mypy, pylint, flake8, and More on Jupyter Notebooks*. https://github.com/nbQA-dev/nbQA.
Nektos (2022). *Run Your GitHub Actions Locally*. https://github.com/nektos/act.
Neovim (2022). *Hyperextensible Vim-Based Text Editor*. https://neovim.io/.
Neptune Labs (2022). *Neptune Documentation*. https://docs.neptune.ai/.
Newman S (2021). Building Microservices: Designing Fine-Grained Systems. O'Reilly.
NLTK Team (2021). *NLTK: A Natural Language Toolkit*. https://www.nltk.org/.
Nteract Team (2022a). *Papermill Is a Tool for Parameterizing and Executing Jupyter Notebooks*. https://papermill.readthedocs.io.
Nteract Team (2022b). *Testbook*. https://testbook.readthedocs.io/en/latest/.
Nvidia (2018). *Nvidia Turing GPU Architecture: Graphics Reinvented*. https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf.
Nvidia (2021). *CUDA Toolkit Documentation*. https://docs.nvidia.com/cuda/.

Oanda (2018). *A C++ Fixed Point Math Library Suitable for Financial Applications*. https://github.com/oanda/libfixed.
Odena A , Olsson C , Andersen D , Goodfellow I (2019). TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing. Proceedings of Machine Learning Research (ICML 2018), 97 , 4901–4911.
ONNX (2021). *Open Neural Network Exchange*. https://github.com/onnx/onnx.
Open Container Initiative (2022). *Open Container Initiative*. https://opencontainers.org/.
Open Virtualization Alliance (2022). *Documents*. https://www.linux-kvm.org/page/Documents.
Openrefine (2022). *A Free, Open Source, Powerful Tool for Working with Messy Data*. https://openrefine.org.
Oracle (2022). *Oracle VM Virtualbox*. https://www.virtualbox.org/.
Ousterhout J (2018). A Philosophy of Software Design. Yaknyam Press.
Overton ML (2001). Numerical Computing with IEEE Floating Point Arithmetic. SIAM.
Pachyderm (2022). *Data-Centric Pipelines and Data Versioning*. https://docs.pachyderm.com/latest.
PagerDuty (2022). *PagerDuty: Uptime Is Money*. https://www.pagerduty.com/.
Palantir (2022). *Python Language Server*. https://github.com/palantir/python-language-server.
Pallets Team (2022). *Flask Documentation*. https://flask.palletsprojects.com/en/latest.
Papernot N , McDaniel P , Sinha A , Wellman MP (2018). SoK: Security and Privacy in Machine Learning. In *Proceedings of the IEEE European Symposium on Security and Privacy* , pp. 399–414.
Paszke A , Gross S , Massa F , Lerer A , Bradbury J , Chanan G , Killeen T , Lin Z , Gimelshein N , Antiga L , Desmaison A , Kopf A , Yang E , DeVito Z , Raison M , Tejani A , Chilamkurthy S , Steiner B , Fang L , Bai J , Chintala S (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems (NIPS), volume 32, pp. 8026–8037.
Pennington J , Socher R , Manning C (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* , pp. 1532–1543.
Pineau J , Vincent-Lamarre P , Sinha K , Larivière V , Beygelzimer A , d'Alché-Buc F , Fox E , Larochelle H (2021). Improving Reproducibility in Machine Learning Research (A Report from the NeurIPS 2019 Reproducibility Program). Journal of Machine Learning Research, 22 , 1–20.
Plotly (2022a). *Analytical Web Apps for Python, R, Julia, and Jupyter. No JavaScript Required*. https://github.com/plotly/dash.
Plotly (2022b). *Dash Python User Guide*. https://dash.plotly.com/.
Plotly (2022c). *Plotly Open Source Graphing Library for Python*. https://plotly.com/python/.
Polyaxon (2022). *Polyaxon Documentation*. https://github.com/polyaxon/polyaxon.
Popejoy A , Fullerton SM (2016). Genomics Is Failing on Diversity. Nature, 538 , 161–164.
Popescu M (2019). Pair Programming Explained. https://shopify.engineering/pair-programming-explained.
Poseidon Laboratories (2022). *Typhoon Documentation*. https://typhoon.psdn.io/#documentation.
Potvin R , Levenberg J (2016). Why Google Stores Billions of Lines of Code in a Single Repository. Communications of the ACM, 59 (7), 78–87.
Prefect (2022). *Prefect 2.0 Documentation*. https://docs.prefect.io.
Preston-Werner T (2022). Semantic Versioning. https://semver.org/.
Prinz F , Schlange T , Asadullah K (2011). Believe It or Not: How Much Can We Rely on Published Data on Potential Drug Targets? Nature Reviews Drug Discovery, 10 , 712.
Progress Software (2022). *Chef Documentation*. https://docs.chef.io.
Project Jupyter (2022). *Jupyter*. https://jupyter.org/.
Prometheus Authors, The Linux Foundation (2022). *Prometheus: Monitoring System and Time Series Databases*. https://prometheus.io/.
Puppet (2022). *Puppet Documentation*. https://puppet.com/docs.
Pyright (2022). *Static Type Checker for Python*. https://github.com/microsoft/pyright.
Python Packaging Authority (2022). *Building and Distributing Packages with Setuptools*. https://setuptools.pypa.io/en/latest/userguide/index.html.
Python Packaging Authority (PyPA) (2022). *Virtualenv Documentation*. https://virtualenv.pypa.io/en/latest/.
Python Software Foundation (2022a). *PyPI: The Python Package Index*. https://pypi.org/.
Python Software Foundation (2022b). *Test Interactive Python Examples*. https://docs.python.org/3/library/doctest.html.
Python Software Foundation (2022c). *unittest: Unit Testing Framework*. https://docs.python.org/3/library/unittest.html.
QS Quacquarelli Symonds (2022). *QS World University Rankings*. https://www.topuniversities.com/qs-world-university-rankings.
Quest K (2022). Standard Go Project Layout. https://github.com/golang-standards/project-layout.
Radford A , Wu J , Child R , Luan D , Amodei D , Sutskever I (2019). Language Models Are Unsupervised Multitask Learners. https://openai.com/blog/better-language-models/.
Ramírez S (2022). FastAPI Framework, High Performance, Easy to Learn, Fast to Code, Ready for Production. https://fastapi.tiangolo.com.
Ranganathan P , Pramesh CS , Aggarwal R (2017). Common Pitfalls in Statistical Analysis: Logistic Regression. Perspectives in Clinical Research, 8 (3), 148–151.
Rasmussen CE , Williams CKI (2006). Gaussian Processes for Machine Learning. MIT Press.
Rathgeber F (2022). Strip Output from Jupyter and IPython Notebooks. https://github.com/kynan/nbstripout.
Read the Docs (2022). *Read the Docs: Documentation Simplified*. https://docs.readthedocs.io.
REditorSupport (2022). *R in Visual Studio Code*. https://marketplace.visualstudio.com/items?itemName=REditorSupport.r.
Řehůřek R , Sojka P (2022a). Gensim Documentation. https://radimrehurek.com/gensim/auto_examples/index.html.
Řehůřek R , Sojka P (2022b). Gensim Documentation. https://radimrehurek.com/gensim/models/word2vec.html.
Reitz K , Python Packaging Authority (PyPA) (2022). Pipenv: Python Dev Workflow for Humans. https://pipenv.pypa.io.
Reuther A , Michaleas P , Jones M , Gadepally V , Samsi S , Kepner J (2020). Survey of Machine Learning Accelerators. In *Proceedings of the 2020 IEEE High Performance Extreme Computing Conference (HPEC)* , pp. 1–12.
Ribeiro MT , Singh S , Guestrin C (2016). Why Should I Trust You? Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* , pp. 1135–1144. ACM.

Rice L (2020). Container Security: Fundamental Technology Concepts that Protect Containerized Applications. O'Reilly.
Rigby P , Bird C (2013). Convergent Contemporary Software Peer Review Practices. In *Proceedings of the 9th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering* , pp. 202–212.
Rong X (2014). Word2vec Parameter Learning Explained. arXiv preprint arXiv:1411.2738.
Royce WW (1987). Managing the Development of Large Software Systems: Concepts and Techniques. In *Proceedings of the 9th International Conference on Software Engineering* , pp. 328–338.
RStudio (2022a). *Open Source and Enterprise-Ready Professional Software for Data Science*. https://www.rstudio.com.
RStudio (2022b). *RStudio Server*. https://www.rstudio.com/products/rstudio/#rstudio-server.
Rump SM (2020a). Addendum to 'On Recurrences Converging to the Wrong Limit in Finite Precision'. Electronic Transactions on Numerical Analysis, 52 , 571–575.
Rump SM (2020b). On Recurrences Converging to the Wrong Limit in Finite Precision. Electronic Transactions on Numerical Analysis, 52 , 358–369.
Russell SJ , Norvig P (2009). Artificial Intelligence: A Modern Approach. 3rd edition. Prentice Hall.
Sadowski C , Söderberg E , Church L , Sipko M , Bacchelli A (2018). Modern Code Review: A Case Study at Google. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice* , pp. 181–190.
Saltz JS , Shamshurin I (2017). Does Pair Programming Work in a Data Science Context? An Initial Case Study. In *Proceedings of the IEEE International Conference on Big Data* , pp. 2348–2354.
Santner TJ , Williams BJ , Notz EI (2018). The Design and Analysis of Computer Experiments. 2nd edition. Springer.
Satyanarayan A , Moritz D , Wongsuphasawat K , Heer J (2022). A High-Level Grammar of Interactive Graphics. https://vega.github.io/vega-lite/docs/.
Schubert E , Sander J , Ester M , Kriegel HP , Xu X (2017). DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. ACM Transactions on Database Systems, 42 (3), 19.
Scikit-learn Developers (2022). *Scikit-learn: Machine Learning in Python*. https://scikit-learn.org/.
Sculley D , Holt G , Golovin D , Davydov E , Phillips T , Ebner D , Chaudhary V , Young M (2014). Machine Learning: The High Interest Credit Card of Technical Debt. In *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)* .
Sculley D , Holt G , Golovin D , Davydov E , Phillips T , Ebner D , Chaudhary V , Young M , Crespo JF , Dennison D (2015). Hidden Technical Debt in Machine Learning Systems. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)* , volume 2, pp. 2503–2511.
Scutari M , Denis JB (2021). Bayesian Networks with Examples in R. 2nd edition. Chapman & Hall.
Seldon Technologies (2022). Seldon Core. https://docs.seldon.io/projects/seldon-core/en/latest/.
Services AW (2022). AWS Deep Learning Containers. https://aws.amazon.com/en/machine-learning/containers/.
Seven D (2014). Knightmare: A DevOps Cautionary Tale. https://dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale/.
Shelton K (2017). The Value of Search Results Rankings. https://www.forbes.com/sites/forbesagencycouncil/2017/10/30/the-value-of-search-results-rankings/.
Sherman E (2022). What Zillow's Failed Algorithm Means for the Future of Data Science. https://fortune.com/education/business/articles/2022/02/01/what-zillows-failed-algorithm-means-for-the-future-of-data-science/.
Shinyama Y , Guglielmetti P , Marsman P (2022). Pdfminer.six's Documentation. https://pdfminersix.readthedocs.io/en/latest/.
Shiraishi M , Washizaki H , Fukazawa Y , Yoder J (2019). Mob Programming: A Systematic Literature Review. In *Proceedings of the IEEE 43rd Annual Computer Software and Applications Conference* , pp. 616–621.
SIGHUP (2022). *Kubernetes Fury Distribution*. https://docs.kubernetesfury.com/docs/distribution/.
Silverlake Software (2022). *Velocity: The Documentation and Docset Viewer for Windows*. https://velocity.silverlakesoftware.com/.
Simmons AJ , Barnett S , Rivera-Villicana J , Bajaj A , Vasa R (2020). A Large-Scale Comparative Analysis of Coding Standard Conformance in Open-Source Data Science Projects. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* , pp. 1–11.
Simonyan K , Vedaldi A , Zisserman A (2014). Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR), Workshop Track* .
SmartBear Software (2021). *OpenAPI Specification*. https://swagger.io/specification/.
Snowflake (2022). *Snowflake Documentation*. https://docs.snowflake.com.
Sonatype (2022). *Nexus Repository Manager*. https://www.sonatype.com/products/nexus-repository.
Spotify (2022a). *Luigi Documentation*. https://luigi.readthedocs.io/en/stable/.
Spotify (2022b). *Spotify Engineering Blog*. https://engineering.atspotify.com/.
Stapleton Cordasco I (2022). Flake8: Your Tool for Style Guide Enforcement. https://flake8.pycqa.org/en/latest/.
Stevens JR (2017). Replicability and Reproducibility in Comparative Psychology. Frontiers in Psychology, 8 , 862.
Streamlit (2022). *Streamlit Documentation*. https://docs.streamlit.io/.
Superconductive (2022). *Great Expectations*. https://docs.greatexpectations.io/docs.
Swoboda S (2021). Connecting with Mob Programming. https://shopify.engineering/mob-programming.
Tableau (2022). *Execute Python Code on The Fly and Display Results in Tableau Visualizations*. https://tableau.github.io/TabPy.
Tableau Software (2022). *Tableau*. https://www.tableau.com/.
Tabuchi A , Kasagi A , Yamazaki M , Honda T , Miwa M , Shiraishi T , Kosaki M , Fukumoto N , Tabaru T , Ike A , Nakashima K (2019). Extremely Accelerated Deep Learning: ResNet-50 Training in 70.4 Seconds. Poster at SC19., https://sc19.supercomputing.org/proceedings/tech_poster/poster_files/rpost203s2-file3.pdf.
Tang Y , Khatchadouriant R , Bagherzadeh M , Singh R , Stewart A , Raja A (2021). An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems. In *Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering* , pp. 238–250.
Tatman R , VanderPlas J , Dane S (2018). A Practical Taxonomy of Reproducibility for Machine Learning Research. In *Proceedings of 2nd the Reproducibility in Machine Learning Workshop at ICML 2018* .
TensorFlow (2021a). *TensorFlow*. https://www.tensorflow.org/overview/.

TensorFlow (2021b). *TensorFlow Extended (TFX)*. https://www.tensorflow.org/tfx/.
Tensorflow (2021). *XLA: Optimizing Compiler for Machine Learning*. https://www.tensorflow.org/xla.
TensorFlow (2022a). *ML Metadata*. https://www.tensorflow.org/tfx/guide/mlmd.
TensorFlow (2022b). *Serving Models*. https://www.tensorflow.org/tfx/guide/serving.
TensorFlow (2022c). *TensorBoard: TensorFlow's Visualization Toolkit*. https://www.tensorflow.org/tensorboard.
TensorFlow (2022d). *The TFX User Guide*. https://www.tensorflow.org/tfx/guide.
The Apache Software Foundation (2022a). *Airflow Documentation*. https://airflow.apache.org/docs/.
The Apache Software Foundation (2022b). *Apache Beam Documentation*. https://beam.apache.org/documentation/.
The Apache Software Foundation (2022c). *Apache Hadoop*. https://hadoop.apache.org/.
The Apache Software Foundation (2022d). *Apache Hive Documentation*. https://cwiki.apache.org/confluence/display/Hive.
The Apache Software Foundation (2022e). *Apache Pig Documentation*. https://pig.apache.org/docs/latest.
The Apache Software Foundation (2022f). *Apache Spark Documentation*. https://spark.apache.org/docs/latest.
The Containers Organization (2022). *podman*. https://podman.io.
The Delta Lake Project Authors (2022a). *Delta Lake Documentation*. https://docs.delta.io.
The Delta Lake Project Authors (2022b). *Zeal Is an Offline Documentation Browser for Software Developers*. https://zealdocs.org.
The Economist (2017). *The World's Most Valuable Resource Is No Longer Oil, but Data*. https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data.
The Economist (2020). *An Understanding of AI's Limitations Is Starting to Sink In*. https://www.economist.com/technology-quarterly/2020/06/11/an-understanding-of-ais-limitations-is-starting-to-sink-in.
The Fluentd Project (2022). *Fluentd: Open Source Data Collector*. https://www.fluentd.org/.
The Git Development Team (2022). *Git Source Code Mirror*. https://github.com/git/git.
The Hadolint Project (2022). *Hadolint: Haskell Dockerfile Linter Documentation*. https://github.com/hadolint/hadolint.
The KServe Authors (2022). *KServe Control Plane*. https://kserve.github.io/website/latest/modelserving/control_plane.
The Kubeflow Authors (2022). *All of Kubeflow documentation*. https://www.kubeflow.org/docs/.
The Kubernetes Authors (2022a). *Kubernetes*. https://kubernetes.io/.
The Kubernetes Authors (2022b). *Kubernetes Documentation: Schedule GPUs*. https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/.
The Kubernetes Authors (2022c). *minikube*. https://minikube.sigs.k8s.io/docs.
The mypy Project (2014). *mypy: Optional Static Typing for Python*. http://mypy-lang.org/.
The Register (2020). *Twilio: Someone Waltzed into Our Unsecured AWS S3 Silo, Added Dodgy Code to Our JavaScript SDK for Customers*. https://www.theregister.com/2020/07/21/twilio_javascript_sdk_code_injection.
Thomas D , Hunt A (2019). *The Pragmatic Programmer: Your Journey to Mastery*. Anniversary edition. Addison-Wesley.
Tian Y , Zhang Y , Stol KJ , Jiang L , Liu H (2022). What Makes a Good Commit Message? In *Proceedings of the 44th International Conference on Software Engineering* , pp. 1–13.
Tornhill A , Borg M (2022). Code Red: The Business Impact of Code Quality: A Quantitative Study of 39 Proprietary Production Codebases. In *Proceedings of International Conference on Technical Debt* , pp. 1–10.
Toro AL (2020). Great Code Reviews–The Superpower Your Team Needs. https://shopify.engineering/great-code-reviews.
Tremel E (2017). Deployment Strategies on Kubernetes. https://www.cncf.io/wp-content/uploads/2020/08/CNCF-Presentation-Template-K8s-Deployment.pdf.
Trifacta (2022). *Profile, Prepare, and Pipeline Data for Analytics and Machine Learning*. https://www.trifacta.com.
Tsay RS (2010). Analysis of Financial Time Series. 3rd edition. Wiley.
Uber (2022). *Piranha: A Tool for Refactoring Code Related to Feature Flag APIs*. https://github.com/uber/piranha.
Uber Technologies (2022). *Uber Engineering Blog*. https://eng.uber.com/.
Unicode (2021). *Unicode Technical Documentation*. https://www.unicode.org/main.html.
Ushey K , Allaire J , Tang Y (2022a). reticulate: Interface to Python. https://cran.r-project.org/web/packages/reticulate.
Ushey K , McPherson J , Cheng J , Atkins A , Allaire J , Allen T (2022b). Packrat: Reproducible Package Management for R. https://rstudio.github.io/packrat/.
van der Schaar M , Alaa AM , Floto A , Gimson A , Scholtes S , Wood A , McKinney E , Jarrett D , Liò P , Ercole A (2021). How Artificial Intelligence and Machine Learning Can Help Healthcare Systems Respond to COVID-19. Machine Learning, 110 , 1–14.
van Heesch D (2022). Doxygen. https://www.doxygen.nl/index.html.
van Oort B , Cruz L , Aniche M , van Deursen A (2021). The Prevalence of Code Smells in Machine Learning Projects. In *Proceedings of the 2021 IEEE/ACM 1st Workshop on AI Engineering: Software Engineering for AI* , pp. 35–42.
van Rossum G , Warsaw B , Coghlan N (2001). PEP 8: Style Guide for Python Code. https://peps.python.org/pep-0008/.
van Vliet H (2008). Software Engineering: Principles and Practice. Wiley.
Velero Authors (2022a). *Postman Documentation*. https://learning.postman.com/docs.
Velero Authors (2022b). *Velero Documentation*. https://velero.io/docs.
Virtanen P , Gommers R , Oliphant TE , Haberland M , Reddy T , Cournapeau D , Burovski E , Peterson P , Weckesser W , Bright J , van der Walt SJ , Brett M , Wilson J , Millman KJ , Mayorov N , Nelson ARJ , Jones E , Kern R , Larson E , Carey CJ , Polat I , Feng Y , Moore EW , VanderPlas J , Laxalde D , Perktold J , Cimrman R , Henriksen I , Quintero EA , Harris CR , Archibald AM , Ribeiro AH , Pedregosa F , van Mulbregt P , SciPy 10 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17 , 261–272.
VmWare (2022). *VMware vSphere Documentation*. https://docs.vmware.com/en/VMware-vSphere/index.html.
VMware (2022). *VMware Workstation Pro*. https://www.vmware.com/products/workstation-pro.html.
Voilà Dashboards (2022). *From Notebooks to Standalone Web Applications and Dashboards*. https://voila.readthedocs.io/en/stable.
Volkov V , Demmel JW (2008). Benchmarking GPUs to Tune Dense Linear Algebra. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing* , pp. 1–11.
Voulodimos A , Doulamis N , Doulamis A , Protopapadakis E (2018). Deep Learning for Computer Vision: A Brief Review. Computational Intelligence and Neuroscience, 2018 (7068349), 1–13.

VPNOverview (2022). *Fintech App Switch Leaks Users' Transactions, Personal IDs*. https://vpnoverview.com/news/fintech-app-switch-leaks-users-transactions-personal-ids.
Walters M , Lee Scott P (2021). meta-git: Manage Your Meta Repo and Child Git Repositories. https://www.npmjs.com/package/meta-git.
Waskom M (2022). Seaborn: Statistical Data Visualization. https://seaborn.pydata.org/.
Weights & Biases (2022). *Weights & Biases Documentation*. https://docs.wandb.ai/.
Weisberg S (2014). Applied Linear Regression. 4th edition. Wiley.
Wickham H (2022a). ggplot2: Elegant Graphics for Data Analysis. https://cran.r-project.org/web/packages/ggplot2.
Wickham H (2022b). The tidyverse Style Guide. https://style.tidyverse.org/.
Wickham H , Danenberg P , Csárdi G , Eugster M , RStudio (2022a). roxygen2: In-Line Documentation for R.
Wickham H , François R , LHenry, Müller K (2022b). A Fast, Consistent Tool for Working with Data Frame Like Objects, Both in Memory and Out of Memory. https://cloud.r-project.org/web/packages/dplyr.
Wickham H , Girlich M , RStudio (2022c). tidyr: Tidy Messy Data. https://cloud.r-project.org/web/packages/tidyr.
Wickham H , RStudio, R Core Team (2022d). Unit Testing for R. https://cloud.r-project.org/web/packages/testthat.
Widgren S , et al (2022). git2r: Provides Access to Git Repositories. https://cran.r-project.org/web/packages/git2r/index.html.
Wiggins A (2017). The Twelve Factor App. https://12factor.net.
Wikipedia (2021a). *Cholesky Decomposition*. https://en.wikipedia.org/wiki/Cholesky_decomposition.
Wikipedia (2021b). *Matrix Multiplication Algorithm*. https://en.wikipedia.org/wiki/Matrix_multiplication_algorithm.
Wikipedia (2021c). *QR Decomposition*. https://en.wikipedia.org/wiki/QR_decomposition.
Wilkinson L (2005). The Grammar of Graphics. 2nd edition. Springer.
Williams L , Kessler RR , Cunningham W (2000). Strengthening the Case for Pair Programming. IEEE Software, 17 (4), 19–25.
Williams-Young DB , Li X (2019). On the Efficacy and High-Performance Implementation of Quaternion Matrix Multiplication. https://arxiv.org/abs/1903.05575.
Wu X , Xiao L , Sun Y , Zhang J , Ma T , He L (2022). A Survey of Human-in-the-Loop for Machine Learning. Future Generation Computer Systems, 135 , 364–381.
Xie Y (2015). Dynamic Documents with R and knitr. 2nd edition. CRC Press.
Xie Y , Allaire JJ , Grolemund G (2022). R Markdown: The Definitive Guide. https://bookdown.org/yihui/rmarkdown/.
Xin D , Ma L , Liu J , Song S , Parameswaran A (2018). Accelerating Human-in-the-Loop Machine Learning: Challenges and Opportunities. In *Proceedings of the Second Workshop on Data Management for End-to-End Machine Learning* , pp. 1–4.
Yamashita Y , Stephenson S , et al (2022). pyenv: Simple Python Version Management. https://github.com/pyenv/pyenv.
Yang Z , Dai Z , Yang Y , Carbonell J , Salakhutdinov RR , Le QV (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems (NeurIPS)* , pp. 5753–5763.
You E (2022). Vue.js: The Progressive JavaScript Framework. https://vuejs.org/.
Zaharia M , The Linux Foundation (2022). MLflow Documentation. https://www.mlflow.org/docs/latest/index.html.
Zellers R , Holtzman A , Rashkin H , Bisk Y , Farhadi A , Roesner F , Choi Y (2019). Defending against Neural Fake News. In *Advances in Neural Information Processing Systems (NeurIPS)* , pp. 9054–9065.
Zelvenskiy S , Harisinghani G , Yu T , Ng E , Wei R (2022). Project Radar: Intelligent Early-Fraud Detection. https://eng.uber.com/project-radar-intelligent-early-fraud-detection/.
Zhang H , Cruz L , van Deursen A (2022). Code Smells for Machine Learning Applications. In *Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI* , pp. 1–12.
Zhang JM , Harman M , Ma L , Liu Y (2020). Machine Learning Testing: Survey, Landscapes and Horizons. IEEE Transactions on Software Engineering, 48 (1), 1–36.
Zheng A (2015). Evaluating Machine Learning Models. O'Reilly.