

O'REILLY®



Feature Engineering for Machine Learning

PRINCIPLES AND TECHNIQUES FOR DATA SCIENTISTS

Alice Zheng & Amanda Casari

Chinese U of HK 2006

Feature Engineering for Machine Learning

Principles and Techniques for Data Scientists

Alice Zheng and Amanda Casari

O'REILLY®

Beijing • Boston • Farnham • Sebastopol • Tokyo

Feature Engineering for Machine Learning

by Alice Zheng and Amanda Casari

Copyright © 2018 Alice Zheng, Amanda Casari. 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.

Editors: Rachel Roumeliotis and Jeff Bleiel Indexer: Ellen Troutman

Production Editor: Kristen Brown

Interior Designer: David Futato

Copyeditor: Rachel Head

Cover Designer: Karen Montgomery

Proofreader: Sonia Saruba

Illustrator: Rebecca Demarest

April 2018: First Edition

Revision History for the First Edition

- 2018-03-23: First Release

See <http://oreilly.com/catalog/errata.csp?isbn=9781491953242> for release details.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Feature Engineering for Machine Learning*, 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.

978-1-491-95324-2

[LSI]

Preface

Introduction

Machine learning fits mathematical models to data in order to derive insights or make predictions. These models take features as input. A *feature* is a numeric representation of an aspect of raw data. Features sit between data and models in the machine learning pipeline. *Feature engineering* is the act of extracting features from raw data and transforming them into formats that are suitable for the machine learning model. It is a crucial step in the machine learning pipeline, because the right features can ease the difficulty of modeling, and therefore enable the pipeline to output results of higher quality. Practitioners agree that the vast majority of time in building a machine learning pipeline is spent on feature engineering and data cleaning. Yet, despite its importance, the topic is rarely discussed on its own. Perhaps this is because the right features can only be defined in the context of both the model and the data; since data and models are so diverse, it's difficult to generalize the practice of feature engineering across projects.

Nevertheless, feature engineering is not just an ad hoc practice. There are deeper principles at work, and they are best illustrated in situ. Each chapter of this book addresses one data problem: how to represent text data or image data, how to reduce the dimensionality of autogenerated features, when and how to normalize, etc. Think of this as a collection of interconnected short stories, as opposed to a single long novel. Each chapter provides a vignette into the vast array of existing feature engineering techniques. Together, they illustrate the overarching principles.

Mastering a subject is not just about knowing the definitions and being able to derive the formulas. It is not enough to know how the mechanism works and what it can do—one must also understand why it is designed that way, how it relates to other techniques, and what the pros and cons of each approach are. Mastery is about knowing precisely how something is done, having an intuition for the underlying principles, and integrating it into one's existing web of knowledge. One does not become a master of something by simply reading a

book, though a good book can open new doors. It has to involve practice—putting the ideas to use, which is an iterative process. With every iteration, we know the ideas better and become increasingly more adept and creative at applying them. The goal of this book is to facilitate the application of its ideas.

This book tries to teach the reason first, and the mathematics second. Instead of only discussing *how* something is done, we try to teach *why*. Our goal is to provide the *intuition* behind the ideas, so that the reader may understand how and when to apply them. There are tons of descriptions and pictures for folks who learn in different ways. Mathematical formulas are presented in order to make the intuition precise, and also to bridge this book with other existing offerings.

Code examples in this book are given in Python, using a variety of free and open source packages. The **NumPy** library provides numeric vector and matrix operations. **Pandas** provides the DataFrame that is the building block of data science in Python. **Scikit-learn** is a general-purpose machine learning package with extensive coverage of models and feature transformers. **Matplotlib** and the styling library **Seaborn** provide plotting and visualization support. You can find these examples as Jupyter notebooks in our **GitHub repo**.

The first few chapters start out slow in order to provide a bridge for folks who are just getting started with data science and machine learning. **Chapter 1** introduces the fundamental concepts in the machine learning pipeline (data, models, features, etc.). In **Chapter 2**, we explore basic feature engineering for numeric data: filtering, binning, scaling, log transforms and power transforms, and interaction features. **Chapter 3** dives into feature engineering for natural text, exploring techniques like bag-of-words, *n*-grams, and phrase detection. **Chapter 4** examines tf-idf (term frequency–inverse document frequency) as an example of feature scaling and discusses why it works. The pace starts to pick up around **Chapter 5**, where we talk about efficient encoding techniques for categorical variables, including feature hashing and bin counting. By the time we get to principal component analysis (PCA) in **Chapter 6**, we are deep in the land of machine learning. **Chapter 7** looks at *k*-means as a featurization technique, which illustrates the useful concept of model stacking. **Chapter 8** is all about images, which are much more challenging in terms of feature extraction than text data. We look at two manual feature extraction techniques, SIFT and HOG, before concluding with an explanation of deep learning as the latest feature

extraction technique for images. We finish up in **Chapter 9** by showing a few different techniques in an end-to-end example, creating a recommender for a dataset of academic papers.

IN LIVING COLOR

The illustrations in this book are best viewed in color. Really, you should print out the color versions of the Swiss roll in **Chapter 7** and paste them into your book. Your aesthetic sense will thank us.

Feature engineering is a vast topic, and more methods are being invented every day, particularly in the area of automatic feature learning. In order to limit the book to a manageable size, we've had to make some cuts. This book does not discuss Fourier analysis for audio data, though it is a beautiful subject that is closely related to eigen analysis in linear algebra (which we touch upon in **Chapters 4 and 6**). We also skip a discussion of random features, which are intimately related to Fourier analysis. We provide an introduction to feature learning via deep learning for image data, but do not go into depth on the numerous deep learning models under active development. Also out of scope are advanced research ideas like random projections, complex text featurization models such as word2vec and Brown clustering, and latent space models like Latent Dirichlet allocation and matrix factorization. If those words mean nothing to you, then you are in luck. If the frontiers of feature learning are where your interest lies, then this is probably not the book for you.

The book assumes knowledge of basic machine learning concepts, such as what a model is and what a vector is, though a refresher is provided so we're all on the same page. Experience with linear algebra, probability distributions, and optimization are helpful, but not necessary.

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.

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.

The book also contains numerous linear algebra equations. We use the following conventions with regard to notation: scalars are shown in lowercase italic (e.g., *a*), vectors in lowercase bold (e.g., **v**), and matrices in uppercase bold and italic (e.g., ***U***).

TIP

This element signifies a tip or suggestion.

NOTE

This element signifies a general note.

WARNING

This element indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at <https://github.com/alicezheng/feature-engineering-book>.

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: “*Feature Engineering for Machine Learning* by Alice Zheng and Amanda Casari (O'Reilly). Copyright 2018 Alice Zheng and Amanda Casari, 978-1-491-95324-2.”

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>.

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/featureEngineering_for_ML.

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

First and foremost, we want to thank our editors, Shannon Cutt and Jeff Bleiel, for shepherding two first-time authors through the (unknown to us) long marathon of book publishing. Without your many check-ins, this book would not have seen the light of day. Thank you also to Ben Lorica, O'Reilly Mastermind, whose encouragement and affirmation turned this from a crazy idea into an actual product. Thank you to Kristen Brown and the O'Reilly production team for their superb attention to detail and extreme patience in waiting for our responses.

If it takes a village to raise a child, it takes a parliament of data scientists to publish a book. We greatly appreciate every hashtag suggestion, notes on room for improvement and calls for clarification. Andreas Müller, Sethu Raman, and Antoine Atallah took precious time out of their busy days to provide technical reviews. Antoine not only did so at lightning speed, but also made available his

beefy machines for use on experiments. Ted Dunning's statistical fluency and mastery of applied machine learning are legendary. He is also incredibly generous with his time and his ideas, and he literally gave us the method and the example described in the *k*-means chapter. Owen Zhang revealed his cache of Kaggle nuggets on using response rate features, which were added to machine learning folklore on bin-counting collected by Misha Bilenko. Thank you also to Alex Ott, Francisco Martin, and David Garrison for additional feedback.

Special Thanks from Alice

I would like to thank the GraphLab/Dato/Turi family for their generous support in the first phase of this project. The idea germinated from interactions with our users. In the process of building a brand new machine learning platform for data scientists, we discovered that the world needs a more systematic understanding of feature engineering. Thank you to Carlos Guestrin for granting me leave from busy startup life to focus on writing.

Thank you to Amanda, who started out as technical reviewer and later pitched in to help bring this book to life. You are the best finisher! Now that this book is done, we'll have to find another project, if only to keep doing our editing sessions over tea and coffee and sandwiches and takeout food.

Special thanks to my friend and healer, Daisy Thompson, for her unwavering support throughout all phases of this project. Without your help, I would have taken much longer to take the plunge, and would have resented the marathon. You brought light and relief to this project, as you do with all your work.

Special Thanks from Amanda

As this is a book and not a lifetime achievement award, I will attempt to scope my thanks to the project at hand.

Many thanks to Alice for bringing me in as a technical editor and then coauthor. I continue to learn so much from you, including how to write better math jokes and explain complex concepts clearly.

Last in order only, special thanks to my husband, Matthew, for mastering the nearly impossible role of grounding me, encouraging me towards my next goal, and never allowing a concept to be hand-waved away. You are the best partner

and my favorite partner in crime. To the biggest and littlest sunshines, you inspire me to make you proud.

Chapter 1. The Machine Learning Pipeline

Before diving into feature engineering, let's take a moment to take a look at the overall machine learning pipeline. This will help us get situated in the larger picture of the application. To that end, we'll begin with a little musing on the basic concepts like *data* and *models*.

Data

What we call *data* are observations of real-world phenomena. For instance, stock market data might involve observations of daily stock prices, announcements of earnings by individual companies, and even opinion articles from pundits. Personal biometric data can include measurements of our minute-by-minute heart rate, blood sugar level, blood pressure, etc. Customer intelligence data includes observations such as “Alice bought two books on Sunday,” “Bob browsed these pages on the website,” and “Charlie clicked on the special offer link from last week.” We can come up with endless examples of data across different domains.

Each piece of data provides a small window into a limited aspect of reality. The collection of all of these observations gives us a picture of the whole. But the picture is messy because it is composed of a thousand little pieces, and there's always measurement noise and missing pieces.

Tasks

Why do we collect data? There are questions that data can help us answer—questions like “Which stocks I should invest in?” or “How can I live a healthier lifestyle?” or “How can I understand my customers' changing tastes, so that my business can serve them better?”

The path from data to answers is full of false starts and dead ends (see [Figure 1-1](#)). What starts out as a promising approach may not pan out. What was

originally just a hunch may end up leading to the best solution. Workflows with data are frequently multistage, iterative processes. For instance, stock prices are observed at the exchange, aggregated by an intermediary like Thomson Reuters, stored in a database, bought by a company, converted into a Hive store on a Hadoop cluster, pulled out of the store by a script, subsampled, massaged, and cleaned by another script, dumped to a file, and converted to a format that you can try out in your favorite modeling library in R, Python, or Scala. The predictions are then dumped back out to a CSV file and parsed by an evaluator, and the model is iterated multiple times, rewritten in C++ or Java by your production team, and run on all of the data before the final predictions are pumped out to another database.



Figure 1-1. The garden of bifurcating paths between data and answers

However, if we disregard the mess of tools and systems for a moment, we might see that the process involves two mathematical entities that are the bread and butter of machine learning: *models* and *features*.

Models

Trying to understand the world through data is like trying to piece together reality using a noisy, incomplete jigsaw puzzle with a bunch of extra pieces. This is where mathematical modeling—in particular statistical modeling—comes in.

The language of statistics contains concepts for many frequent characteristics of data, such as *wrong*, *redundant*, or *missing*. Wrong data is the result of a mistake in measurement. Redundant data contains multiple aspects that convey exactly the same information. For instance, the day of week may be present as a categorical variable with values of “Monday,” “Tuesday,” ... “Sunday,” and again included as an integer value between 0 and 6. If this day-of-week information is not present for some data points, then you’ve got missing data on your hands.

A *mathematical model* of data describes the relationships between different aspects of the data. For instance, a model that predicts stock prices might be a formula that maps a company’s earning history, past stock prices, and industry to the predicted stock price. A model that recommends music might measure the similarity between users (based on their listening habits), and recommend the same artists to users who have listened to a lot of the same songs.

Mathematical formulas relate numeric quantities to each other. But raw data is often not numeric. (The action “Alice bought *The Lord of the Rings* trilogy on Wednesday” is not numeric, and neither is the review that she subsequently writes about the book.) There must be a piece that connects the two together. This is where features come in.

Features

A *feature* is a numeric representation of raw data. There are many ways to turn raw data into numeric measurements, which is why features can end up looking like a lot of things. Naturally, features must derive from the type of data that is available. Perhaps less obvious is the fact that they are also tied to the model; some models are more appropriate for some types of features, and vice versa. The right features are relevant to the task at hand and should be easy for the model to ingest. *Feature engineering* is the process of formulating the most appropriate features given the data, the model, and the task.

The number of features is also important. If there are not enough informative features, then the model will be unable to perform the ultimate task. If there are too many features, or if most of them are irrelevant, then the model will be more expensive and tricky to train. Something might go awry in the training process that impacts the model’s performance.

Model Evaluation

Features and models sit between raw data and the desired insights (see [Figure 1-2](#)). In a machine learning workflow, we pick not only the model, but also the features. This is a double-jointed lever, and the choice of one affects the other. Good features make the subsequent modeling step easy and the resulting model more capable of completing the desired task. Bad features may require a much more complicated model to achieve the same level of performance. In the rest of this book, we will cover different kinds of features and discuss their pros and cons for different types of data and models. Without further ado, let's get started!

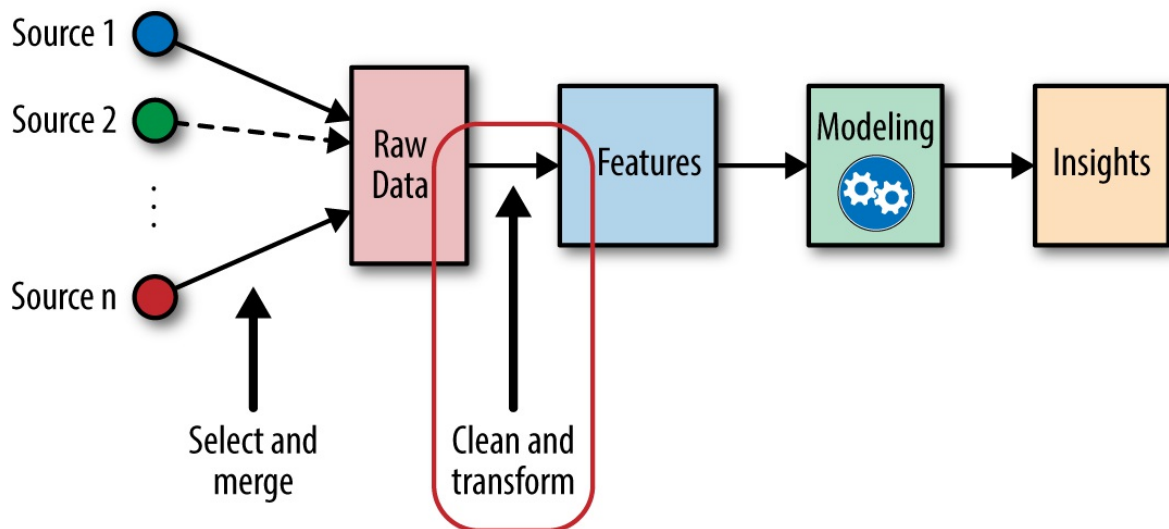


Figure 1-2. The place of feature engineering in the machine learning workflow