Probability for Machine Learning

Discover How To Harness Uncertainty With Python

Jason Brownlee

MACHINE LEARNING MASTERY

Acknowledgements

Special thanks to my copy editor Sarah Martin and my technical editors Michael Sanderson and Arun Koshy.

Edition: v1.1

Contents

Preface

Probability is foundational to machine learning and required background for machine learning practitioners.

- Probability is a prerequisite in most courses and books on applied machine learning.
- Probability methods are used at each step in an applied machine learning project.
- Probabilistic frameworks underlie the training of many machine learning algorithms.

Data distributions and statistics don't make sense without probability. Fitting models on a training dataset does not make sense without probability. Interpreting the performance of a stochastic learning algorithm does not make sense without probability. A machine learning practitioner cannot be effective without an understanding and appreciation of the basic concepts and methods from the field of probability.

Practitioners Don't Know Probability

Developers don't know probability, and this is a huge problem. Programmers don't need to know or use probability in order to develop software. Software engineering and computer science courses focus on deterministic programs, with inputs, outputs, and no randomness or noise. As such, it is common for machine learning practitioners coming from the computer science or developer tradition not to know and not value probabilistic thinking. This is a problem given that the bedrock of a predictive modeling project is probability.

Practitioners Study the Wrong Probability

Eventually, machine learning practitioners realize the need for skills in probability. This might start with a need to better interpret descriptive statistics and may progress to the need to understand the probabilistic frameworks behind many popular machine learning algorithms. The problem is, they don't seek out the probability information they need. Instead, they try to read through a textbook on probability or work through the material for an undergraduate course on probabilistic methods. This approach is slow, it's boring, and it covers a breadth and depth of material on probability that is beyond the needs of the machine learning practitioner.

Practitioners Study Probability Wrong

It's worse than this. Regardless of the medium used to learn probability, be it books, videos, or course material, machine learning practitioners study probability the wrong way. Because the material is intended for undergraduate students that need to pass a test, the material is focused on the math, theory, proofs, and derivations. This is great for testing students but terrible for practitioners that need results. Practitioners need methods that clearly state when they are appropriate and instruction on how to interpret the results. They need intuitions behind the complex equations. They need code examples that they can use immediately on their project.

A Better Way

I set out to write a playbook for machine learning practitioners that gives them only those parts of probability that they need to be more effective at working through a predictive modeling project. I set out to present probability methods in the way that practitioners learn: that is with simple language and working code examples. Probability is important to machine learning, and I believe that if it is taught at the right level for practitioners, it can be a fascinating, fun, directly applicable, and immeasurably useful area of study. I hope that you agree.

Jason Brownlee
2019

Part I

Introduction

Welcome

Welcome to *Probability for Machine Learning*. The field of probability is hundreds of years old and probabilistic methods are central to working through predictive modeling problems with machine learning. The simplest concepts when working with data come from the field of probability, such as the most likely value and the idea of a probability distribution over observations. Building on these simple ideas, probabilistic frameworks like maximum likelihood estimation underly a range of machine learning algorithms, from logistic regression to neural networks. I designed this book to teach you step-by-step the basics of probability with concrete and executable examples in Python.

Who Is This Book For?

Before we get started, let's make sure you are in the right place. This book is for developers that may know some applied machine learning. Maybe you know how to work through a predictive modeling problem end-to-end, or at least most of the main steps, with popular tools. The lessons in this book do assume a few things about you, such as:

- You know your way around basic Python for programming.
- You may know some basic NumPy for array manipulation.
- You want to learn probability to deepen your understanding and application of machine learning.

This guide was written in the top-down and results-first machine learning style that you're used to from Machine Learning Mastery.

About Your Outcomes

This book will teach you the basics of probability that you need to know as a machine learning practitioner. After reading and working through this book, you will know:

- About the field of probability, how it relates to machine learning, and how to harness probabilistic thinking on a machine learning project.
- How to calculate different types of probability, such as joint, marginal, and conditional probability.

- How to consider data in terms of random variables and how to recognize and sample from common discrete and continuous probability distribution functions.
- How to frame learning as maximum likelihood estimation and how this important probabilistic framework is used for regression, classification, and clustering machine learning algorithms.
- How to use probabilistic methods to evaluate machine learning models directly without evaluating their performance on a test dataset.
- How to calculate and consider probability from the Bayesian perspective and to calculate conditional probability with Bayes theorem for common scenarios.
- How to use Bayes theorem for classification with Naive Bayes, optimization with Bayesian Optimization, and graphical models with Bayesian Networks.
- How to quantify uncertainty using measures of information and entropy from the field of information theory and calculate quantities such as cross-entropy and mutual information.
- How to develop and evaluate naive classifiers using a probabilistic framework.
- How to evaluate classification models that predict probabilities and calibrate probability predictions.

This new basic understanding of probability will impact your practice of machine learning in the following ways:

- Confidently calculate and wield both frequentist probability (counts) and Bayesian probability (beliefs) generally and within the context of machine learning datasets.
- Confidently select and use loss functions and performance measures when training machine learning algorithms, backed by a knowledge of the underlying probabilistic framework (e.g. maximum likelihood estimation) and the relationships between metrics (e.g. cross-entropy and negative log-likelihood).
- Confidently evaluate classification predictive models, including establishing a robust baseline in performance, probabilistic performance measures, and calibrated predicted probabilities.

This book is not a substitute for an undergraduate course in probability or a textbook for such a course, although it could complement such materials. For a good list of top courses, textbooks, and other resources on probability, see the *Further Reading* section at the end of each tutorial.

How to Read This Book

This book was written to be read linearly, from start to finish. That being said, if you know the basics and need help with a specific technique or type of problem, then you can flip straight to that section and get started. This book was designed for you to read on your workstation, on the screen, not on a tablet or eReader. My hope is that you have the book open right next to your editor and run the examples as you read about them. This book is not intended to be read passively or be placed in a folder as a reference text. It is a playbook, a workbook, and a guidebook intended for you to learn by doing and then applying your new understanding with working Python examples. To get the most out of the book, I would recommend playing with the examples in each tutorial. Extend them, break them, then fix them. Try some of the extensions presented at the end of each lesson and let me know how you do.

About the Book Structure

This book was designed around major probabilistic techniques that are directly relevant to applied machine learning. There are a lot of things you could learn about probability, from theory to abstract concepts to APIs. My goal is to take you straight to developing an intuition for the elements you must understand with laser-focused tutorials. I designed the tutorials to focus on how to get things done with probability. They give you the tools to both rapidly understand and apply each technique or operation. Each tutorial is designed to take you about one hour to read through and complete, excluding the extensions and further reading. You can choose to work through the lessons one per day, one per week, or at your own pace. I think momentum is critically important, and this book is intended to be read and used, not to sit idle. I recommend picking a schedule and sticking to it. The tutorials are divided into seven parts; they are:

- **Part 1: Foundations**. Discover a gentle introduction to the field of probability, the relationship to machine learning, and the importance that probability has when working through predictive modeling problems.
- **Part 2: Basics**. Discover the different types of probability, such as marginal, joint, and conditional, and worked examples that develop an intuition for how to calculate each.
- **Part 3: Distributions**. Discover that probability distributions summarize the likelihood of events and common distribution functions for discrete and continuous random variables.
- **Part 4: Maximum Likelihood**. Discover the maximum likelihood estimation probabilistic framework that underlies how the parameters of many machine learning algorithms are fit on training data.
- **Part 5: Bayesian Probability**. Discover Bayes theorem and some of the most important uses in applied machine learning, such as the naive Bayes algorithm and Bayesian optimization.
- **Part 6: Information Theory**. Discover the relationship between probability and information theory and some of the most important concepts to applied machine learning, such as cross-entropy and information gain.
- **Part 7: Classification**. Discover the relationship between classification and probability, including models that predict probabilities for class labels, evaluation metrics, and probability calibration.

Each part targets a specific learning outcome, and so does each tutorial within each part. This acts as a filter to ensure you are only focused on the things you need to know to get to a specific result and do not get bogged down in the math or near-infinite number of digressions. The tutorials were not designed to teach you everything there is to know about each of the theories or techniques of probability. They were designed to give you an understanding of how they work, how to use them, and how to interpret the results the fastest way I know how: to learn by doing.

About Python Code Examples

The code examples were carefully designed to demonstrate the purpose of a given lesson. Code examples are complete and standalone. The code for each lesson will run as-is with no code from prior lessons or third-parties needed beyond the installation of the required packages. A complete working example is presented with each tutorial for you to inspect and copy-and-paste. All source code is also provided with the book, and I would recommend running the provided files whenever possible to avoid any copy-paste issues. The provided code was developed in a text editor and is intended to be run on the command line. No special IDE or notebooks are required. If you are using a more advanced development environment and are having trouble, try running the example from the command line instead. All code examples were tested on a POSIX-compatible machine with Python 3.

About Further Reading

Each lesson includes a list of further reading resources. This may include:

- Research papers.
- Books and book chapters.
- Web Pages and Articles.
- API documentation.

Wherever possible, I tried to list and link to the relevant API documentation for key objects and functions used in each lesson so you can learn more about them. I have tried to link to books on Amazon. I don't know everything, and if you discover a good resource related to a given lesson, please let me know so I can update the book.

About Getting Help

You might need help along the way. Don't worry; you are not alone.

- **Help with a Technique?** If you need help with the technical aspects of a specific operation or technique, see the *Further Reading* section at the end of each tutorial.

- **Help with APIs?** If you need help with using a specific Python function, see the list of resources in the *Further Reading* section at the end of each lesson, and also see *Appendix A*.
- **Help with your workstation?** If you need help setting up your environment, I would recommend using Anaconda and following my tutorial in *Appendix B*.
- **Help in general?** You can shoot me an email. My details are in *Appendix A*.

Next

Are you ready? Let's dive in! Next up you will discover that probability is an important foundational field of mathematics.

Part II

Distributions

Chapter 1

Continuous Probability Distributions

The probability for a continuous random variable can be summarized with a continuous probability distribution. Continuous probability distributions are encountered in machine learning, most notably in the distribution of numerical input and output variables for models and in the distribution of errors made by models. Knowledge of the normal continuous probability distribution is also required more generally in the density and parameter estimation performed by many machine learning models. As such, continuous probability distributions play an important role in applied machine learning and there are a few distributions that a practitioner must know about. In this tutorial, you will discover continuous probability distributions used in machine learning. After completing this tutorial, you will know:

- The probability of outcomes for continuous random variables can be summarized using continuous probability distributions.
- How to parameterize, define, and randomly sample from common continuous probability distributions.
- How to create probability density and cumulative density plots for common continuous probability distributions.

Let's get started.

1.1 Tutorial Overview

This tutorial is divided into four parts; they are:

1. Continuous Probability Distributions
2. Normal Distribution
3. Exponential Distribution
4. Pareto Distribution

1.2 Continuous Probability Distributions

A random variable is a quantity produced by a random process. A continuous random variable is a random variable that has a real numerical value. Each numerical outcome of a continuous random variable can be assigned a probability. The relationship between the events for a continuous random variable and their probabilities is called the continuous probability distribution and is summarized by a probability density function, or PDF for short.

Unlike a discrete random variable, the probability for a given continuous random variable cannot be specified directly; instead, it is calculated as an integral (area under the curve) for a tiny interval around the specific outcome. The probability of an event equal to or less than a given value is defined by the cumulative distribution function, or CDF for short. The inverse of the CDF is called the percentage-point function and will give the discrete outcome that is less than or equal to a probability.

- **PDF: Probability Density Function**, returns the probability of a given continuous outcome.
- **CDF: Cumulative Distribution Function**, returns the probability of a value less than or equal to a given outcome.
- **PPF: Percent-Point Function**, returns a discrete value that is less than or equal to the given probability.

There are many common continuous probability distributions. The most common is the normal probability distribution. Practically all continuous probability distributions of interest belong to the so-called exponential family of distributions, which are just a collection of parameterized probability distributions (e.g. distributions that change based on the values of parameters).

Continuous probability distributions play an important role in machine learning from the distribution of input variables to the models, the distribution of errors made by models, and in the models themselves when estimating the mapping between inputs and outputs. In the following sections, will take a closer look at some of the more common continuous probability distributions.

1.3 Normal Distribution

The normal distribution is also called the Gaussian distribution (named for Carl Friedrich Gauss) or the bell curve distribution. The distribution covers the probability of real-valued events from many different problem domains, making it a common and well-known distribution, hence the name *normal*. A continuous random variable that has a normal distribution is said to be *normal* or *normally distributed*. Some examples of domains that have normally distributed events include:

- The heights of people.
- The weights of babies.
- The scores on a test.

The distribution can be defined using two parameters:

- **Mean** (mu or μ): The expected value.
- **Variance** (sigma^2 or σ^2): The spread from the mean.

Often, the standard deviation is used instead of the variance, which is calculated as the square root of the variance, e.g. normalized.

- **Standard Deviation** (sigma or σ): The average spread from the mean.

A normal distribution with a mean of zero and a standard deviation of 1 is called a standard normal distribution, and often data is reduced or *standardized* to this for analysis for ease of interpretation and comparison. We can define a distribution with a mean of 50 and a standard deviation of 5 and sample random numbers from this distribution. We can achieve this using the `normal()` NumPy function. The example below samples and prints 10 numbers from this distribution.

```
# sample a normal distribution
from numpy.random import normal
# define the distribution
mu = 50
sigma = 5
n = 10
# generate the sample
sample = normal(mu, sigma, n)
print(sample)
```

Listing 1.1: Example of sampling from a normal distribution.

Running the example prints 10 numbers randomly sampled from the defined normal distribution.

```
[48.71009029 49.36970461 45.58247748 51.96846616 46.05793544 40.3903483
 48.39189421 50.08693721 46.85896352 44.83757824]
```

Listing 1.2: Example output from sampling from a normal distribution.

A sample of data can be checked to see if it is normal by plotting it and checking for the familiar normal shape, or by using statistical tests. If the samples of observations of a random variable are normally distributed, then they can be summarized by just the mean and variance, calculated directly on the samples. We can calculate the probability of each observation using the probability density function. A plot of these values would give us the tell-tale bell shape. We can define a normal distribution using the `norm()` SciPy function and then calculate properties such as the moments, PDF, CDF, and more. The example below calculates the probability for integer values between 30 and 70 in our distribution and plots the result, then does the same for the cumulative probability.

```
# pdf and cdf for a normal distribution
from scipy.stats import norm
from matplotlib import pyplot
# define distribution parameters
mu = 50
sigma = 5
```

```
# create distribution
dist = norm(mu, sigma)
# plot pdf
values = [value for value in range(30, 70)]
probabilities = [dist.pdf(value) for value in values]
pyplot.plot(values, probabilities)
pyplot.show()
# plot cdf
cprobs = [dist.cdf(value) for value in values]
pyplot.plot(values, cprobs)
pyplot.show()
```

Listing 1.3: Example of plotting the PDF and CDF for the normal distribution.

Running the example first calculates the probability for integers in the range [30, 70] and creates a line plot of values and probabilities. The plot shows the Gaussian or bell-shape with the peak of highest probability around the expected value or mean of 50 with a probability of about 8%.

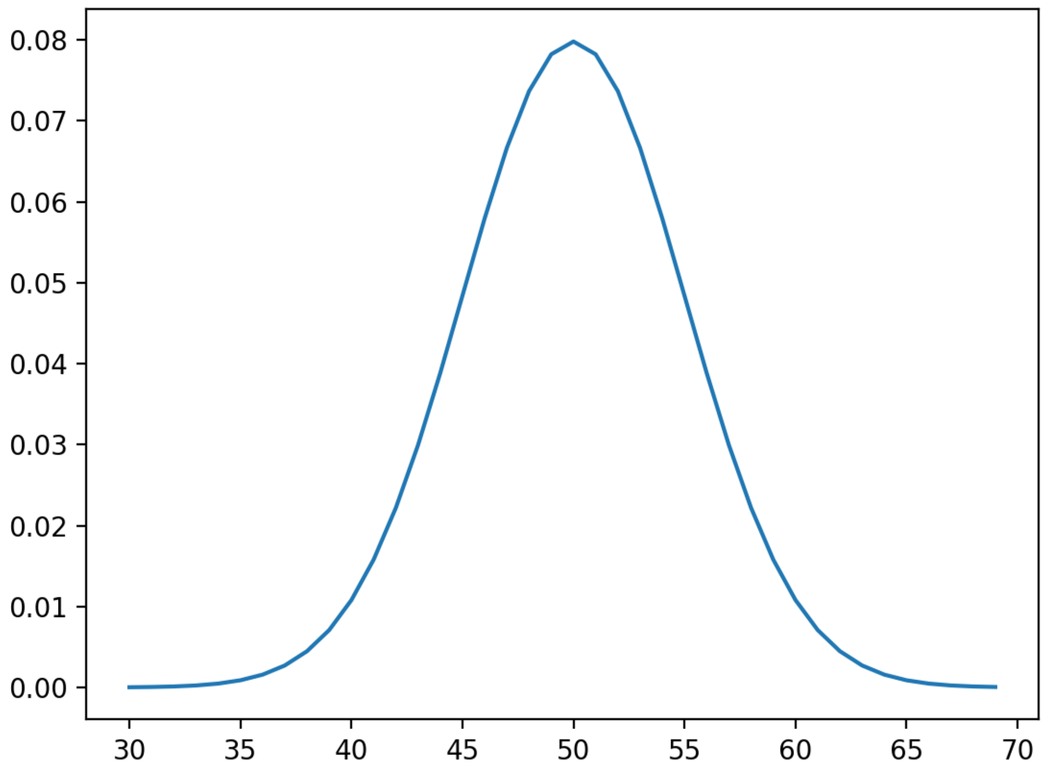

Figure 1.1: Line Plot of Events vs Probability or the Probability Density Function for the Normal Distribution.

The cumulative probabilities are then calculated for observations over the same range, showing that at the mean, we have covered about 50% of the expected values and very close to 100% after the value of about 65 or 3 standard deviations from the mean $(50 + (3 \times 5))$.

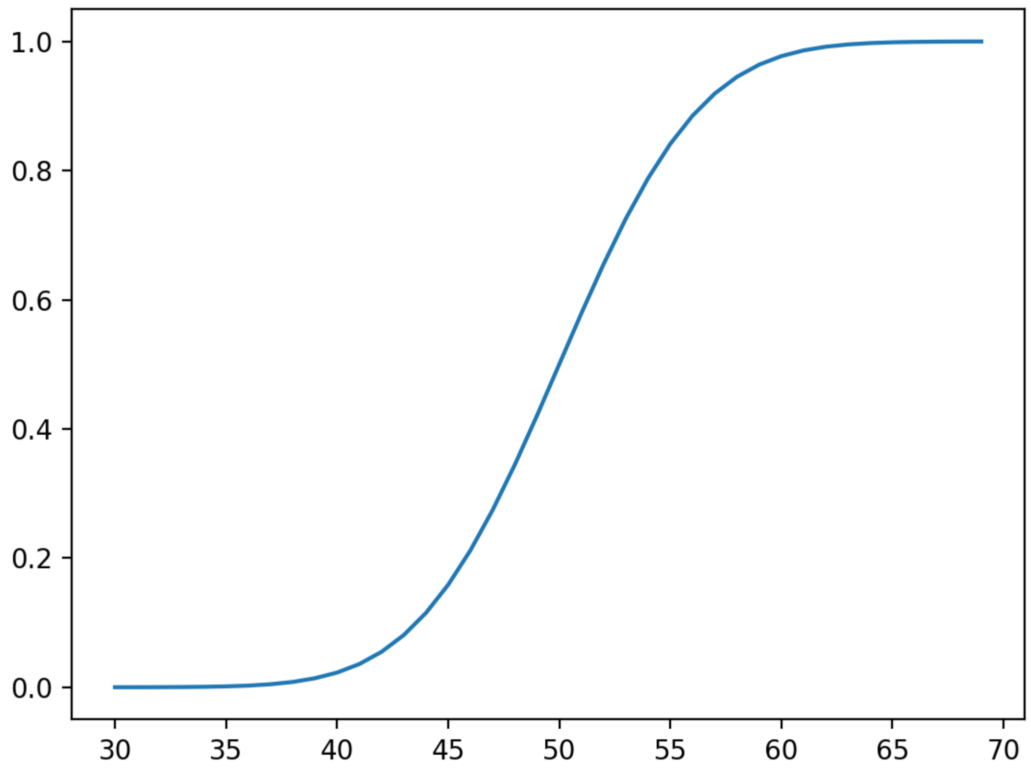

Figure 1.2: Line Plot of Events vs. Cumulative Probability or the Cumulative Density Function for the Normal Distribution.

In fact, the normal distribution has a heuristic or rule of thumb that defines the percentage of data covered by a given range by the number of standard deviations from the mean. It is called the 68-95-99.7 rule, which is the approximate percentage of the data covered by ranges defined by 1, 2, and 3 standard deviations from the mean. For example, in our distribution with a mean of 50 and standard deviation of 5, we would expect 95% of the data to be covered by values that are 2 standard deviations from the mean, or $50 - (2 \times 5)$ and $50 + (2 \times 5)$ or between 40 and 60. We can confirm this by calculating the exact values using the percentage-point function. The middle 95% would be defined by the percentage point function value for 2.5% at the low end and 97.5% at the high end, where 97.5 to 2.5 gives the middle 95%. The complete example is listed below.

```python
# calculate the values that define the middle 95%
from scipy.stats import norm
# define distribution parameters
mu = 50
sigma = 5
# create distribution
dist = norm(mu, sigma)
low_end = dist.ppf(0.025)
high_end = dist.ppf(0.975)
print('Middle 95%% between %.1f and %.1f' % (low_end, high_end))
```

Listing 1.4: Example of calculating the values that define the middle of the distribution.

Running the example gives the exact outcomes that define the middle 95% of expected outcomes that are very close to our standard-deviation-based heuristics of 40 and 60.

```
Middle 95% between 40.2 and 59.8
```

Listing 1.5: Example output from calculating the values that define the middle of the distribution.

An important related distribution is the Log-Normal probability distribution.

1.4 Exponential Distribution

The exponential distribution is a continuous probability distribution where a few outcomes are the most likely with a rapid decrease in probability to all other outcomes. It is the continuous random variable equivalent to the geometric probability distribution for discrete random variables. Some examples of domains that have exponential distribution events include:

- The time between clicks on a Geiger counter.
- The time until the failure of a part.
- The time until the default of a loan.

The distribution can be defined using one parameter:

- **Scale** (Beta or β): The mean and standard deviation of the distribution.

Sometimes the distribution is defined more formally with a parameter lambda or rate. The beta parameter is defined as the reciprocal of the lambda parameter ($\beta = \frac{1}{\lambda}$)

- **Rate** (lambda or λ) = Rate of change in the distribution.

We can define a distribution with a mean of 50 and sample random numbers from this distribution. We can achieve this using the `exponential()` NumPy function. The example below samples and prints 10 numbers from this distribution.

```python
# sample an exponential distribution
from numpy.random import exponential
# define the distribution
beta = 50
n = 10
# generate the sample
sample = exponential(beta, n)
print(sample)
```

Listing 1.6: Example of sampling from an exponential distribution.

Running the example prints 10 numbers randomly sampled from the defined distribution.

```
[  3.32742946 39.10165624 41.86856606 85.0030387 28.18425491
  68.20434637 106.34826579 19.63637359 17.13805423 15.91135881]
```

Listing 1.7: Example output from sampling from a exponential distribution.

We can define an exponential distribution using the `expon()` SciPy function and then calculate properties such as the moments, PDF, CDF, and more. The example below defines a range of observations between 50 and 70 and calculates the probability and cumulative probability for each and plots the result.

```
# pdf and cdf for an exponential distribution
from scipy.stats import expon
from matplotlib import pyplot
# define distribution parameter
beta = 50
# create distribution
dist = expon(beta)
# plot pdf
values = [value for value in range(50, 70)]
probabilities = [dist.pdf(value) for value in values]
pyplot.plot(values, probabilities)
pyplot.show()
# plot cdf
cprobs = [dist.cdf(value) for value in values]
pyplot.plot(values, cprobs)
pyplot.show()
```

Listing 1.8: Example of plotting the PDF and CDF for the exponential distribution.

Running the example first creates a line plot of outcomes versus probabilities, showing a familiar exponential probability distribution shape.

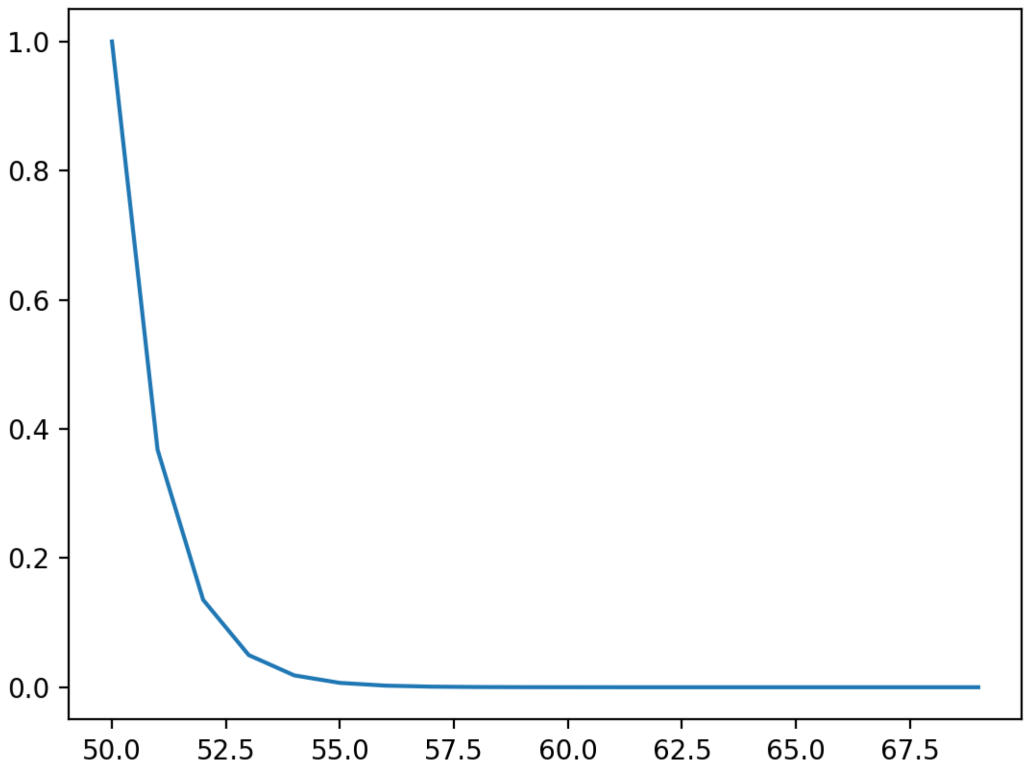

Figure 1.3: Line Plot of Events vs. Probability or the Probability Density Function for the Exponential Distribution.

Next, the cumulative probabilities for each outcome are calculated and graphed as a line plot, showing that after perhaps a value of 55 that almost 100% of the expected values will be observed.

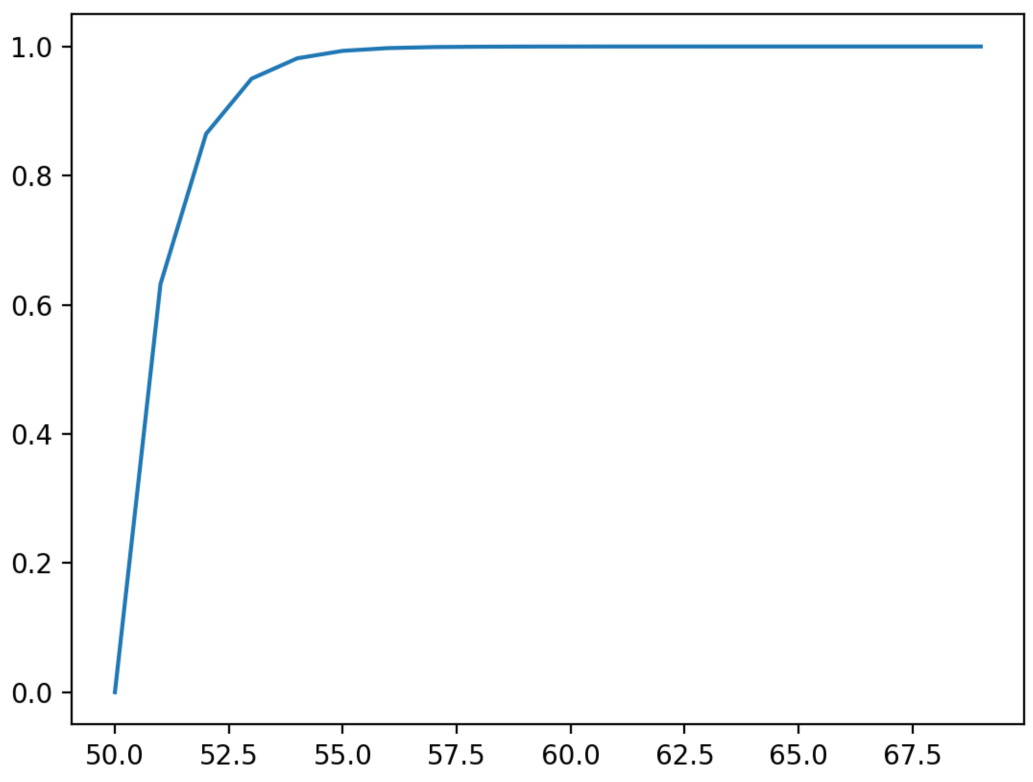

Figure 1.4: Line Plot of Events vs. Cumulative Probability or the Cumulative Density Function for the Exponential Distribution.

An important related distribution is the double exponential distribution, also called the Laplace distribution.

1.5 Pareto Distribution

A Pareto distribution is named after Vilfredo Pareto and is may be referred to as a power-law distribution. It is also related to the Pareto principle (or 80/20 rule) which is a heuristic for continuous random variables that follow a Pareto distribution, where 80% of the events are covered by 20% of the range of outcomes, e.g. most events are drawn from just 20% of the range of the continuous variable. The Pareto principle is just a heuristic for a specific Pareto distribution, specifically the Pareto Type II distribution, that is perhaps most interesting and on which we will focus. Some examples of domains that have Pareto distributed events include:

- The income of households in a country.
- The total sales of books.
- The scores by players on a sports team.

The distribution can be defined using one parameter:

- **Shape** (alpha or α): The steepness of the decease in probability.

Values for the shape parameter are often small, such as between 1 and 3, with the Pareto principle given when alpha is set to 1.161. We can define a distribution with a shape of 1.1 and sample random numbers from this distribution. We can achieve this using the `pareto()` NumPy function.

```
# sample a pareto distribution
from numpy.random import pareto
# define the distribution
alpha = 1.1
n = 10
# generate the sample
sample = pareto(alpha, n)
print(sample)
```

Listing 1.9: Example of sampling from an Pareto distribution.

Running the example prints 10 numbers randomly sampled from the defined distribution.

```
[0.5049704 0.0140647 2.13105224 3.10991217 2.87575892 1.06602639
 0.22776379 0.37405415 0.96618778 3.94789299]
```

Listing 1.10: Example output from sampling from a Pareto distribution.

We can define a Pareto distribution using the `pareto()` SciPy function and then calculate properties, such as the moments, PDF, CDF, and more. The example below defines a range of observations between 1 and about 10 and calculates the probability and cumulative probability for each and plots the result.

```
# pdf and cdf for a pareto distribution
from scipy.stats import pareto
from matplotlib import pyplot
# define distribution parameter
alpha = 1.5
# create distribution
dist = pareto(alpha)
# plot pdf
values = [value/10.0 for value in range(10, 100)]
probabilities = [dist.pdf(value) for value in values]
pyplot.plot(values, probabilities)
pyplot.show()
# plot cdf
cprobs = [dist.cdf(value) for value in values]
pyplot.plot(values, cprobs)
pyplot.show()
```

Listing 1.11: Example of plotting the PDF and CDF for the Pareto distribution.

Running the example first creates a line plot of outcomes versus probabilities, showing a familiar Pareto probability distribution shape.

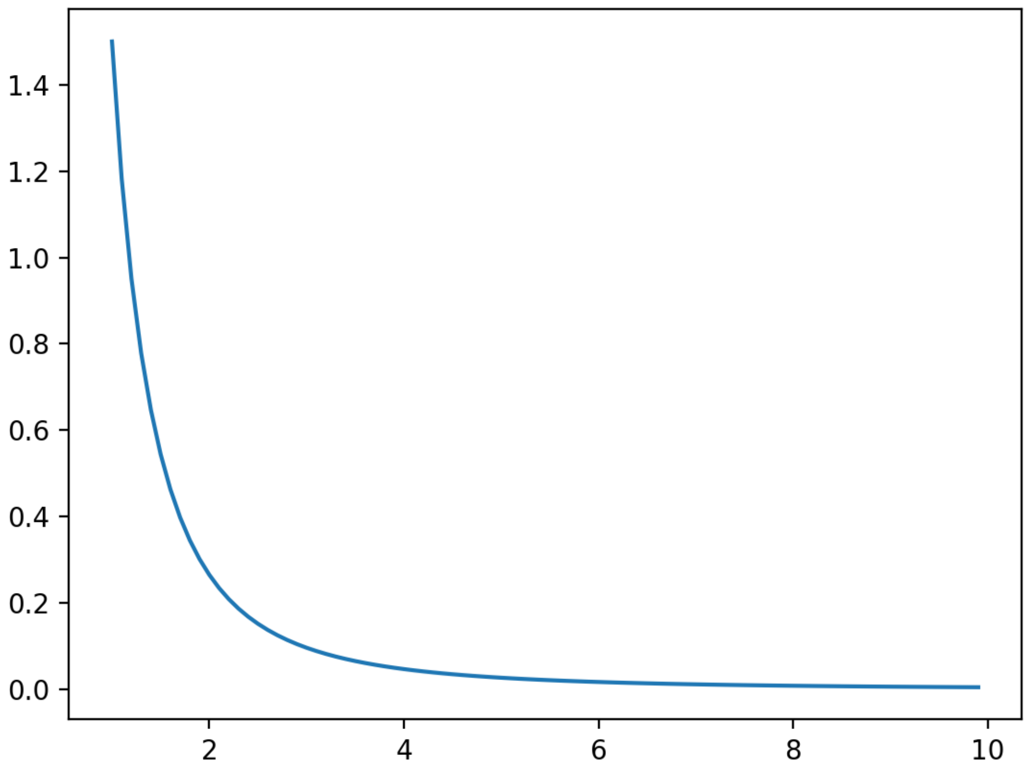

Figure 1.5: Line Plot of Events vs. Probability or the Probability Density Function for the Pareto Distribution.

Next, the cumulative probabilities for each outcome are calculated and graphed as a line plot, showing a rise that is less steep than the exponential distribution seen in the previous section.

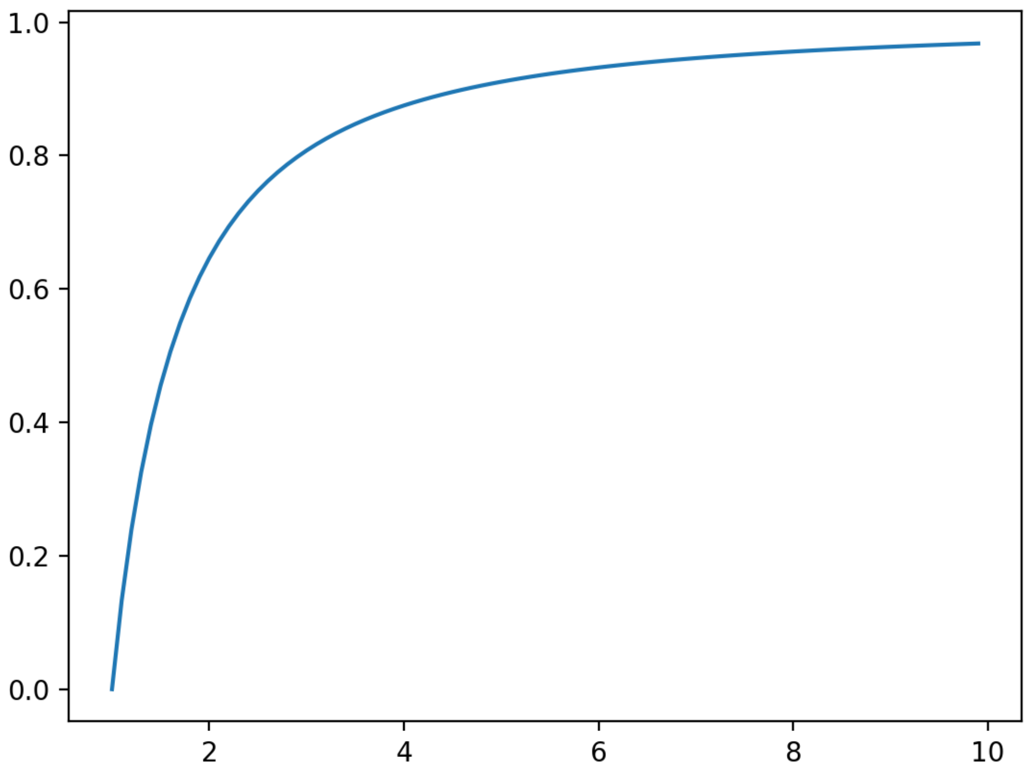

Figure 1.6: Line Plot of Events vs. Cumulative Probability or the Cumulative Density Function for the Pareto Distribution.

1.6 Further Reading

This section provides more resources on the topic if you are looking to go deeper.

1.6.1 Books

- Chapter 2: Probability Distributions, *Pattern Recognition and Machine Learning*, 2006.
  https://amzn.to/2JwHE7I
- Section 3.9: Common Probability Distributions, *Deep Learning*, 2016.
  https://amzn.to/2lnc3vL
- Section 2.3: Some common discrete distributions, *Machine Learning: A Probabilistic Perspective*, 2012.
  https://amzn.to/2xKSTCP

1.6.2 API

- Continuous Statistical Distributions, SciPy.
  https://docs.scipy.org/doc/scipy/reference/tutorial/stats/continuous.html
- Random sampling (`numpy.random`), NumPy.
  https://docs.scipy.org/doc/numpy/reference/routines.random.html

1.6.3 Articles

- Normal distribution, Wikipedia.
  https://en.wikipedia.org/wiki/Normal_distribution
- 68-95-99.7 rule, Wikipedia.
  https://en.wikipedia.org/wiki/68%E2%80%9395%E2%80%9399.7_rule
- Exponential distribution, Wikipedia.
  https://en.wikipedia.org/wiki/Exponential_distribution
- Pareto distribution, Wikipedia.
  https://en.wikipedia.org/wiki/Pareto_distribution

1.7 Summary

In this tutorial, you discovered continuous probability distributions used in machine learning. Specifically, you learned:

- The probability of outcomes for continuous random variables can be summarized using continuous probability distributions.
- How to parameterize, define, and randomly sample from common continuous probability distributions.
- How to create probability density and cumulative density plots for common continuous probability distributions.

1.7.1 Next

In the next tutorial, you will discover the challenge of density estimation and how to fit density estimation models.