Chapter 1. Overview of Machine Learning Systems

In November 2016, Google announced that it had incorporated its multilingual neural machine translation system into Google Translate, marking one of the first success stories of deep artificial neural networks in production at scale.[1] According to Google, with this update, the quality of translation improved more in a single leap than they had seen in the previous 10 years combined.

This success of deep learning renewed the interest in machine learning (ML) at large. Since then, more and more companies have turned toward ML for solutions to their most challenging problems. In just five years, ML has found its way into almost every aspect of our lives: how we access information, how we communicate, how we work, how we find love. The spread of ML has been so rapid that it's already hard to imagine life without it. Yet there are still many more use cases for ML waiting to be explored in fields such as health care, transportation, farming, and even in helping us understand the universe.[2]

Many people, when they hear "machine learning system," think of just the ML algorithms being used such as logistic regression or different types of neural networks. However, the algorithm is only a small part of an ML system in production. The system also includes the business requirements that gave birth to the ML project in the first place, the interface where users and developers interact with your system, the data stack, and the logic for developing, monitoring, and updating your models, as well as the infrastructure that enables the delivery of that logic. Figure 1-1 shows you the different components of an ML system and in which chapters of this book they will be covered.

THE RELATIONSHIP BETWEEN MLOPS AND ML SYSTEMS DESIGN

Ops in MLOps comes from DevOps, short for Developments and Operations. To operationalize something means to bring it into production, which includes deploying, monitoring, and maintaining it. MLOps is a set of tools and best practices for bringing ML into production.

ML systems design takes a system approach to MLOps, which means that it considers an ML system holistically to ensure that all the components and their stakeholders can work together to satisfy the specified objectives and requirements.

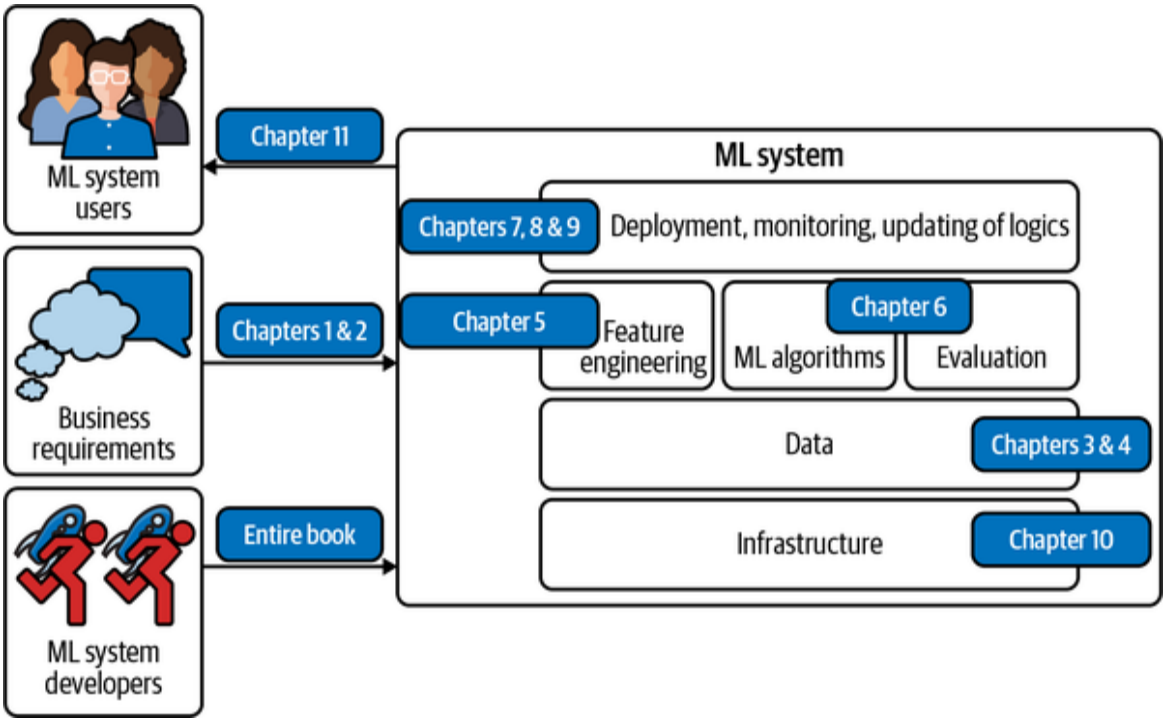

Figure 1-1. Different components of an ML system. "ML algorithms" is usually what people think of when they say machine learning, but it's only a small part of the entire system.

There are many excellent books about various ML algorithms. This book doesn't cover any specific algorithms in detail but rather helps readers understand the entire ML system as a whole. In other words, this book's goal is to provide you with a framework to develop a solution that best works for your problem, regardless of which algorithm you might end up using. Algorithms might become outdated quickly as new algorithms are

constantly being developed, but the framework proposed in this book should still work with new algorithms.

The first chapter of the book aims to give you an overview of what it takes to bring an ML model to production. Before discussing how to develop an ML system, it's important to ask a fundamental question of when and when not to use ML. We'll cover some of the popular use cases of ML to illustrate this point.

After the use cases, we'll move on to the challenges of deploying ML systems, and we'll do so by comparing ML in production to ML in research as well as to traditional software. If you've been in the trenches of developing applied ML systems, you might already be familiar with what's written in this chapter. However, if you have only had experience with ML in an academic setting, this chapter will give an honest view of ML in the real world and set your first application up for success.

When to Use Machine Learning

As its adoption in the industry quickly grows, ML has proven to be a powerful tool for a wide range of problems. Despite an incredible amount of excitement and hype generated by people both inside and outside the field, ML is not a magic tool that can solve all problems. Even for problems that ML can solve, ML solutions might not be the optimal solutions. Before starting an ML project, you might want to ask whether ML is necessary or cost-effective.[3]

To understand what ML can do, let's examine what ML solutions generally do:

> *Machine learning is an approach to (1) learn (2) complex patterns from (3) existing data and use these patterns to make (4) predictions on (5) unseen data.*

We'll look at each of the italicized keyphrases in the above framing to understand its implications to the problems ML can solve:

1. Learn: the system has the capacity to learn

A relational database isn't an ML system because it doesn't have the capacity to learn. You can explicitly state the relationship between two columns in a relational database, but it's unlikely to have the capacity to figure out the relationship between these two columns by itself.

For an ML system to learn, there must be something for it to learn from. In most cases, ML systems learn from data. In supervised learning, based on example input and output pairs, ML systems learn how to generate outputs for arbitrary inputs. For example, if you want to build an ML system to learn to predict the rental price for Airbnb listings, you need to provide a dataset where each input is a listing with relevant characteristics (square footage, number of rooms, neighborhood, amenities, rating of that listing, etc.) and the associated output is the rental price of that listing. Once learned, this ML system should be able to predict the price of a new listing given its characteristics.

2. Complex patterns: there are patterns to learn, and they are complex

ML solutions are only useful when there are patterns to learn. Sane people don't invest money into building an ML system to predict the next outcome of a fair die because there's no pattern in how these outcomes are generated.[4] However, there are patterns in how stocks are priced, and therefore companies have invested billions of dollars in building ML systems to learn those patterns.

Whether a pattern exists might not be obvious, or if patterns exist, your dataset or ML algorithms might not be sufficient to capture them. For example, there might be a pattern in how Elon Musk's tweets affect cryptocurrency prices. However, you wouldn't know until you've rigorously trained and evaluated your ML models on his tweets. Even if all your models fail to make reasonable predictions of cryptocurrency prices, it doesn't mean there's no pattern.

Consider a website like Airbnb with a lot of house listings; each listing comes with a zip code. If you want to sort listings into the states they

are located in, you wouldn’t need an ML system. Since the pattern is simple—each zip code corresponds to a known state—you can just use a lookup table.

The relationship between a rental price and all its characteristics follows a much more complex pattern, which would be very challenging to manually specify. ML is a good solution for this. Instead of telling your system how to calculate the price from a list of characteristics, you can provide prices and characteristics, and let your ML system figure out the pattern. The difference between ML solutions and the lookup table solution as well as general traditional software solutions is shown in Figure 1-2. For this reason, ML is also called Software 2.0.[5]

ML has been very successful with tasks with complex patterns such as object detection and speech recognition. What is complex to machines is different from what is complex to humans. Many tasks that are hard for humans to do are easy for machines—for example, raising a number of the power of 10. On the other hand, many tasks that are easy for humans can be hard for machines—for example, deciding whether there’s a cat in a picture.

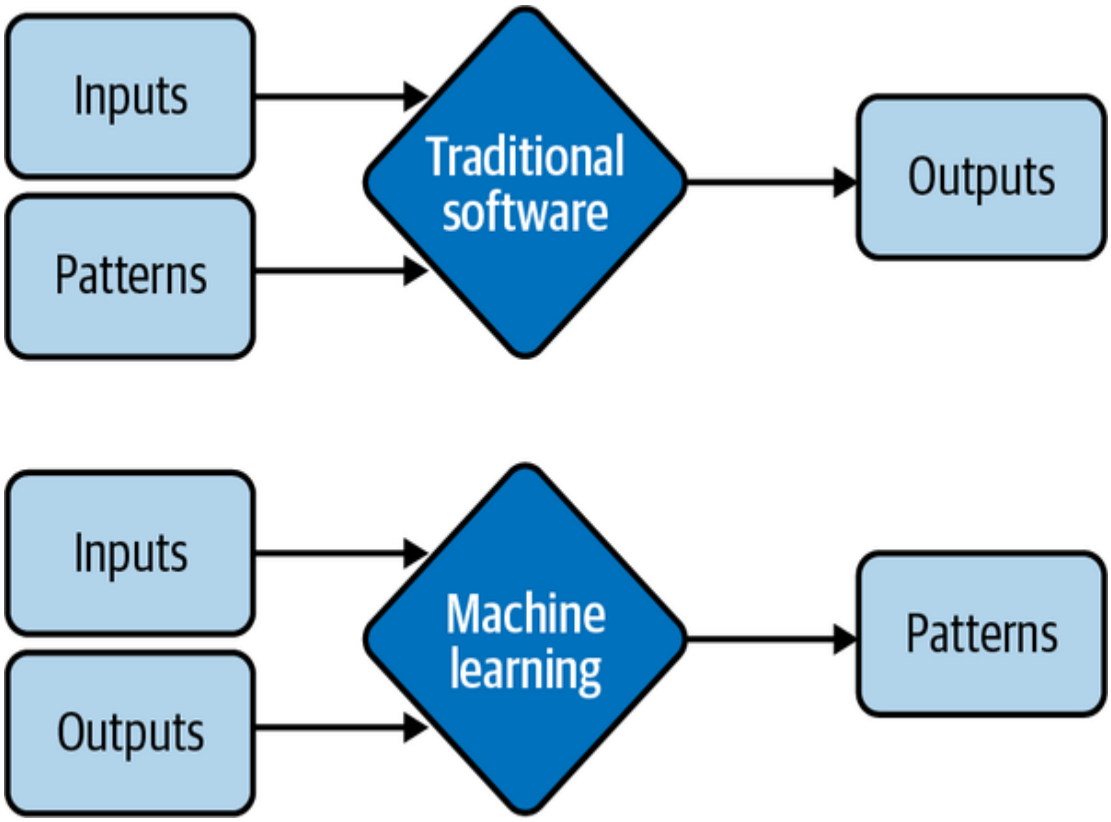

Figure 1-2. Instead of requiring hand-specified patterns to calculate outputs, ML solutions learn patterns from inputs and outputs

3. Existing data: data is available, or it's possible to collect data

Because ML learns from data, there must be data for it to learn from. It's amusing to think about building a model to predict how much tax a person should pay a year, but it's not possible unless you have access to tax and income data of a large population.

In the zero-shot learning (sometimes known as zero-data learning) context, it's possible for an ML system to make good predictions for a task without having been trained on data for that task. However, this ML system was previously trained on data for other tasks, often related to the task in consideration. So even though the system doesn't require data for the task at hand to learn from, it still requires data to learn.

It's also possible to launch an ML system without data. For example, in the context of continual learning, ML models can be deployed without having been trained on any data, but they will learn from incoming data

in production.[6] However, serving insufficiently trained models to users comes with certain risks, such as poor customer experience.

Without data and without continual learning, many companies follow a "fake-it-til-you make it" approach: launching a product that serves predictions made by humans, instead of ML models, with the hope of using the generated data to train ML models later.

4. Predictions: it's a predictive problem

ML models make predictions, so they can only solve problems that require predictive answers. ML can be especially appealing when you can benefit from a large quantity of cheap but approximate predictions. In English, "predict" means "estimate a value in the future." For example, what will the weather be like tomorrow? Who will win the Super Bowl this year? What movie will a user want to watch next?

As predictive machines (e.g., ML models) are becoming more effective, more and more problems are being reframed as predictive problems. Whatever question you might have, you can always frame it as: "What would the answer to this question be?" regardless of whether this question is about something in the future, the present, or even the past.

Compute-intensive problems are one class of problems that have been very successfully reframed as predictive. Instead of computing the exact outcome of a process, which might be even more computationally costly and time-consuming than ML, you can frame the problem as: "What would the outcome of this process look like?" and approximate it using an ML model. The output will be an approximation of the exact output, but often, it's good enough. You can see a lot of it in graphic renderings, such as image denoising and screen-space shading.[7]

5. Unseen data: unseen data shares patterns with the training data

The patterns your model learns from existing data are only useful if unseen data also share these patterns. A model to predict whether an app will get downloaded on Christmas 2020 won't perform very well if it's