

Real-World

Machine Learning

Henrik Brink
Joseph W. Richards

 MANNING



**MEAP Edition
Manning Early Access Program
Real-World Machine Learning
Version 4**

Copyright 2014 Manning Publications

For more information on this and other Manning titles go to
www.manning.com

Welcome

Thank you for purchasing the MEAP for *Real-World Machine Learning*. We're very excited about this release and we're looking forward to getting your feedback in the development of this book. The aim of this book is to teach you practical aspects of modern machine learning in an engaging and visual way, from the initial data munging through model building and advanced optimization techniques.

This book will focus on how you can use and implement machine learning in your projects while keeping the mathematical and algorithmic details to a minimum. We will introduce the advantages of particular methods with real-world examples and highlight common pitfalls based on extensive experience.

We're initially releasing the first two chapters and Appendix A. Chapter 1 provides an introduction to machine learning and motivation for the remaining chapters of the book. We will introduce the essential machine learning workflow that will be the topic of the first part of the book. The second part of the book will introduce some more advanced techniques that can boost the accuracy and scalability of your models even further.

Chapter 2 will start our venture into the essential machine learning workflow by talking about real-world data and how to explore and process various types of data for machine learning modeling. We will introduce powerful visualization techniques that drive our pre-processing requirements and later in chapter 3 will influence the choice of modeling algorithm.

The appendix introduces a table of commonly used machine learning algorithms. This table is meant as a reference for choosing the algorithm that best applies to the data at hand. The book in general will not limit your machine learning understanding to a few currently popular algorithms, but this table will serve as a good reference when choosing the optimal algorithm.

Looking ahead, chapters 3-5 will finish the discussion of the essential machine learning workflow and introduce model building, evaluation and optimization. Chapters 6-9 will highlight advanced techniques that can improve your model in various ways, including feature engineering, clustering and scalability aspects. We will conclude the book with chapter 10 on interesting current and future developments in the rapidly growing field of machine learning.

As you're reading, we hope you'll take advantage of the Author Online forum. We'll be reading your comments and responding. We appreciate any feedback, as it is very helpful in the development process.

— Henrik Brink & Joseph Richards

brief contents

1 Machine Learning?

2 Real World Data

3 Model Building

4 Model evaluation and optimization

5 Basic Feature Engineering

6 Advanced Feature Engineering

7 Unsupervised Learning

8 Scaling With Size and Speed

9 Streaming and Online Machine Learning

10 The Future of Machine Learning

Appendix A Popular Machine Learning Algorithms

1

What is Machine Learning?

This chapter will cover:

- Introduction to practical machine learning
- Advantages of machine learning over the traditional approaches
- Overview of the basic machine learning workflow
- Overview of advanced methods for improving model performance

Machine learning (ML) is the science of building systems that automatically learn from data. ML is a *data-driven* approach to problem solving, in which the hidden patterns and trends present within data are first detected and then leveraged to help make better decisions. This process of automatically learning from data and in turn using that acquired knowledge to inform future decisions is extremely powerful. Indeed, machine learning is rapidly becoming the engine which powers the modern data-driven economy.

At the core of machine learning lies a set complicated algorithms which have been developed over the course of the past few decades by academics in a diverse set of disciplines such as statistics, computer science, robotics, computer vision, physics, and applied mathematics. The aim of this book is not to describe the gory mathematical details of the underlying algorithms (although we will peel back a few layers of the onion to provide the reader some insight into the inner workings of the most common ML algorithms). Rather, the book's primary purpose is to instruct non-experts on how to properly employ machine learning in real-world systems and how to integrate ML into computing pipelines. The book details the end-to-end ML workflow and provides detailed descriptions of each of the steps required to build and integrate an ML system. Wherever possible, the book uses

real business problems to demonstrate the role of machine learning in solving practical problems.

In this first chapter, we will use a real business problem—reviewing loan applications—to demonstrate the advantages of using machine learning over some of the most common alternatives. We will then introduce the essentials of the basic end-to-end ML workflow—from initial data ingest to employment of the ML output—which makes up the content of the first half the book (chapters 2-5). Finally, we introduce the more advanced material in the second half of the book (chapters 6-9), which teaches how to boost the performance of ML in practice.

1.1 *Using Data to Make Decisions*

Every day, humans use data—in the form of past experiences—to make decisions. In that same spirit, machine learning employs past data—in the form of sets of numerical and categorical attributes about past events—to train automated systems to make accurate decisions when future events occur.

In the business world, the ML approach to decision making is able to solve a wide range of business problems, from fraud detection, customer targeting and product recommendation to real-time industrial monitoring, sentiment analysis and demand forecasting. ML allows traditionally labor-intensive tasks to be automated, enabling faster and higher throughput decision making. The most sophisticated ML algorithms can distill messy and heterogeneous data into a salient set of information to make optimized decisions. When built properly, ML systems can make very accurate decisions—often more accurate than a manual approach—and can be engineered to continuously and automatically learn from new data.

In the following example, we describe a real-world business problem that can benefit from a machine-learning approach. We will run through the various alternatives that are commonly used today and demonstrate the advantages of the machine-learning approach.

Imagine that you are in charge of a microlending company that provides loans to individuals that want to start small businesses in troubled communities. Early on, the company receives a few applications per week and you're able to manually read each application and do the necessary background checks on each individual applicant to decide whether or not to approve each loan request in a few days' time. The schematic of this process is shown in Figure 2.1. Your early borrowers are pleased with your short turnaround time and personal service. Word of your company starts to spread.

As your company continues to gain popularity, the number of applicants begins to increase. Soon, you're receiving hundreds of applications per week. You try to stay up with the increased rate of applications by working extra hours, but the backlog of applications continues to grow. Some of your applicants grow weary of waiting and seek loans from your competitors. It is obvious to you that manually processing each application by yourself is not a sustainable business process and frankly is not worth the stress.

So, what should you do? We will explore several possible ways to scale up your application vetting process to meet your increasing business needs.

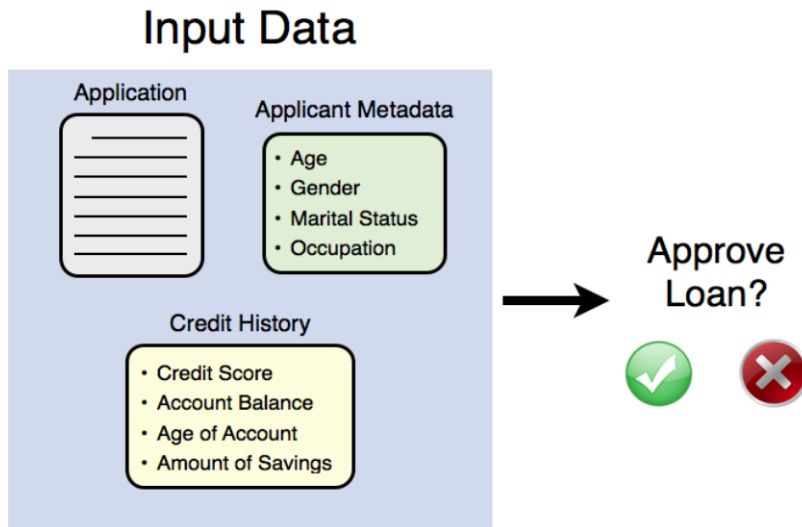


Figure 1.1: Schematic of the loan approval process for the microlending example. For each applicant, a variety of input data—including application text, applicant metadata, and applicant credit history—all contribute to your decision of whether or not to approve the loan.

1.1.1 Traditional Approaches

In this subsection, we explore two traditional data analysis approaches as applied to the application vetting process: manual analysis and business rules. For each of these approaches, we will walk through the process of implementing the technique and highlight the ways in which it falls short of enabling you to build a scalable business.

HIRE ANOTHER ANALYST

Instead of changing your application inspection process, you decide to hire another analyst to help you out. You are not thrilled with the idea of spending some of your profit on a new hire, but with a second person vetting applications, you are able to process roughly twice as many applications in the same amount of time. This new analyst allows you to flush out the application backlog within a week.

For the first couple of weeks, the two of you are able to stay up with demand. Yet, the number of applications continues to grow, doubling within a month to 1000 per week. To keep up with this increased demand, you now must hire two more analysts. Projecting forward, you determine that this pattern of hiring is not sustainable: all of your increased revenue from new loan applicants is going directly to your new hires instead of to more critical areas such as your microlending fund. Simply put, hiring **more analysts as demand increases hinders the growth of your business**. Further, you find that the hiring process is lengthy and expensive, sapping your business of more of its revenue.

Finally, each new hire is less experienced and slower at processing applications than the last and the added stress of managing a team of individuals is wearing on you.

EMPLOY SIMPLE BUSINESS RULES

After your failed experiment of hiring additional analysts, you decide that to grow your business you need to change your application examination process by automating some of the decision-making process. Since you have already been in business for a few months, you now have seen the outcomes of many of the loans that were approved (Figure 1.2). Imagine that of the 1000 loans whose repayment date has passed, 70% were repaid on time. This set of *training data*—loans for which the ultimate outcome of interest is known—is extremely valuable to begin building automation into your loan approval system.

Using the 1000 loans whose repayment status is known, you are now in the position to begin looking for trends. In particular, you perform a manual search for a set of filtering rules that produce a subset of “good” loans that were primarily paid on time. Then, when new applications come in, you can quickly filter them through these hard-coded *business rules* to reduce the number of applications which must be vetted by hand.

So how do we come up with these decision rules? Luckily, through the process of manually analyzing hundreds of applications, you’ve gained extensive experience about what makes each application good or bad¹. Through some introspection and back-testing to loan repayment status, you’ve noticed a few trends in the credit background checks data²:

- most borrowers with a credit line of more than \$7500 defaulted on their loan, and
- most borrowers that had no checking account repaid their loan on time.

Now you can design a filtering mechanism to pare down the number of applications that you need to process manually through those two rules.

¹ Note that you could also use statistical correlation techniques to determine which input data attributes are most strongly associated with the outcome event of loan repayment.

² In this example, we use the German credit data set as example credit-check data. These data are available for download at <http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29>

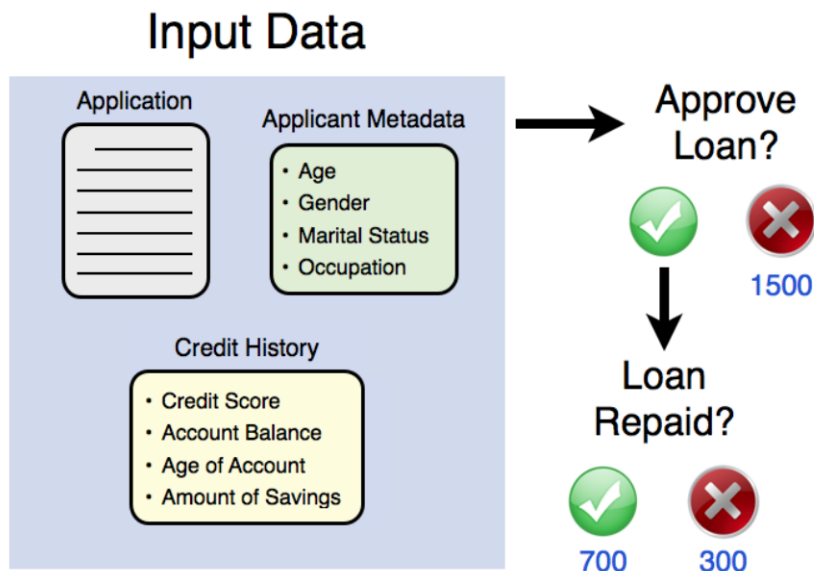


Figure 1.2: After a few months of business, you now have observed the outcome of 1000 loans that were approved. This information is critical to start building automation into your loan approval process.

Your first filter is to automatically reject any applicant that has a credit line of more than \$7500. Looking through your historical data, you find that 44 of the 86 applicants with a credit line of more than \$7500 defaulted on their loan. So, roughly 51% of these high-credit line applicants defaulted, compared to 28% of the rest. This filter seems like a good way to exclude high-risk applicants, but you realize that only 8.6% of your accepted applicants had a credit line that was so high, meaning that you'll still need to manually process more than 90% of applications. You need to do some more filtering to get that number down to something more manageable.

Your second filter is to automatically accept any applicant that does not have a checking account. This seems to be an excellent filter, as 348 of the 394 (88%) of the applicants that did not have a checking account repaid their loans on time. Including this second filter brings the percentage of applications that are automatically accepted or rejected up to 45%. Thus, you only need to manually analyze roughly half of the new incoming applications. These filtering rules are demonstrated in Figure 2.3.

With these two business rules, you can now scale your business up to twice the amount of volume without having to hire a second analyst, since you now only need to manually accept or reject 52% of new applications. Additionally, based on the training set of 1000 applications with known outcome, you expect your filtering mechanism to erroneously reject 42 out of every 1000 applications (4.2%) and to erroneously accept 46 of every 1000 applications (4.6%).

These error rates might be acceptable for now, but perhaps in the future (as demand grows) we will require a smaller rate of erroneously accepting applications (this is called the *false positive* rate). Under a manual filtering process there is no way of lowering these error rates besides adding additional business rules. This can be a debilitating drawback.

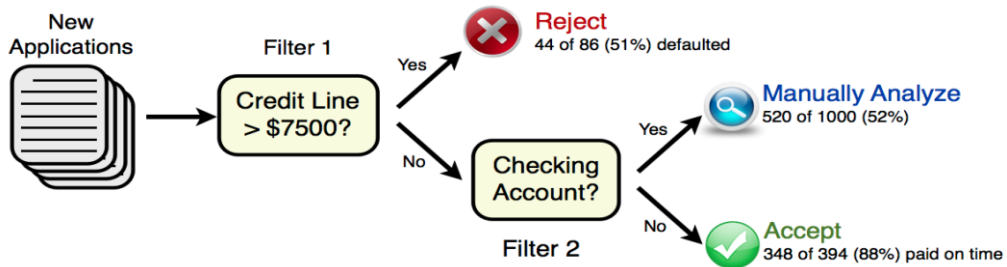


Figure 1.3: Example filtering of new applications through two business rules. Use of the filters allows us to manually analyze only 52% of the incoming applications.

Further, as business grows, you will require your system to automatically accept or reject a larger and larger percentage of applications. To do this, you again need to add more business rules. You will soon find that this kind of system is not a sustainable business process, as a number of problems will arise:

1. Manually finding effective filters becomes harder and harder—if not impossible—as the filtering system grows in complexity.
2. The business rules become so complicated and opaque that debugging them and ripping out old, irrelevant rules becomes virtually impossible.
3. There is no statistical rigor in the construction of your rules. You are pretty sure that better “rules” can be found by better exploration of the data.
4. As the patterns of loan repayment change over time—perhaps due to changes in the population of applicants—the system does not adapt to those changes. To stay up to date, you would have to manually adjust the rules or build a new system from scratch.

All of these drawbacks can be traced to a single debilitating weakness in a business rules approach: The system does not automatically learn from data.

Data-driven systems, from simple statistical models to more sophisticated machine learning workflows, can overcome these problems.

1.1.2 The Machine Learning Approach

Finally, you decide to look into an entirely automated, data-driven approach to your microlending application vetting process. Machine learning is an attractive option because

the completely automated nature of the process will allow your operation to keep pace with the increasing inflow of applications. Further, unlike business rules, ML learns the optimal decisions *directly from the data* without having to arbitrarily hard-code decision rules. This graduation from rules-based to ML-based decision making means that your decisions will be more accurate and will improve over time as more loans are made. In short you can be assured that your ML system produces optimized decisions with minimal hand-holding.

In machine learning, the data provides the foundation for deriving insights about the problem at hand. To determine whether or not to accept each new loan application, ML utilizes historical *training data* to predict the best course of action for each new application. To get started with ML for loans approval, you begin by assembling the training data for the 1000 loans that have been granted. This training data consists of the input data for each loan application along with the known outcome of whether or not each loan was repaid on time. The input data in turn consist of a set of *features*—numerical or categorical metrics that capture the relevant aspects of each application—such as the credit score of the applicant, their gender, and their current occupation.

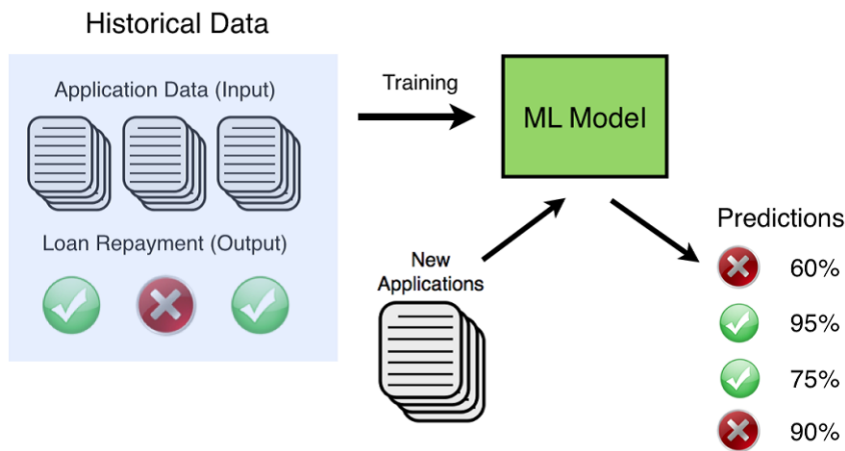


Figure 1.4: Basic ML workflow, as applied to the microloan example. Here, historical data are used to train the machine learning model. Then, as new applications come in, probabilistic predictions of future repayment are generated instantaneously from the application data.

ML modeling, then, determines how the input data for each applicant can be used to *best predict* the loan outcome. By finding and utilizing patterns in the training set, ML produces a model (you can think of this as a black box, for now) that produces a prediction of the outcome for each new applicant, based on that applicant's data.

FROM SIMPLE STATISTICAL MODELS TO MORE POWERFUL ML TOOLS

The next step is to select a ML algorithm to use. Machine learning comes in many flavors, ranging from simple statistical models to more sophisticated, non-parametric approaches. Here, we describe and compare the two basic types of ML model: parametric and non-parametric.

The first type of model is simple *parametric* statistical models (most business analytics tools fall under this category). These models use simple, fixed equations to express the relationship between the outcome variable and the input variables. Data are then used to learn the best values of the unknown terms in the equation. Approaches such as linear regression, logistic regression and autoregressive models all fit under this category.

For instance, imagine that we use logistic regression to model the loans approval process. In logistic regression, the logarithm of the odds that each loan is repaid is modeled as a linear function of the input features. For example, if each new application contained three relevant features: Credit Line of the applicant, Education Level of the applicant, and Age of the applicant, then logistic regression attempts to predict the log-odds that the applicant will default on the loan (we'll call this y) via the equation

$$y = \beta_0 + \beta_1 * \text{Credit_Line} + \beta_2 * \text{Education_Level} + \beta_3 * \text{Age}$$

The optimal values of each of the coefficients of the equation (in this case, β_0 , β_1 , β_2 , and β_3) are learned from the 1000 training data examples.

The amount by which we can learn from data in parametric models such as logistic regression is quite restrictive. Take the data set in the left-hand panel of Figure 1.5 for example. Here, there are two input features and the task is to classify each data point into one of two classes (red circle or green square). The two classes are separated, in the two-dimensional feature space, by a non-linear curve (this is the so-called decision boundary). In the center panel, we see the result of fitting a logistic regression model on this data set. Clearly, the predicted separation boundary (commonly called the *decision boundary*) between the reds and the greens is not close to the true boundary between the classes (denoted by the blue curve). As a result, the simple statistical model makes many classification errors (greens incorrectly predicted as reds and vice versa).

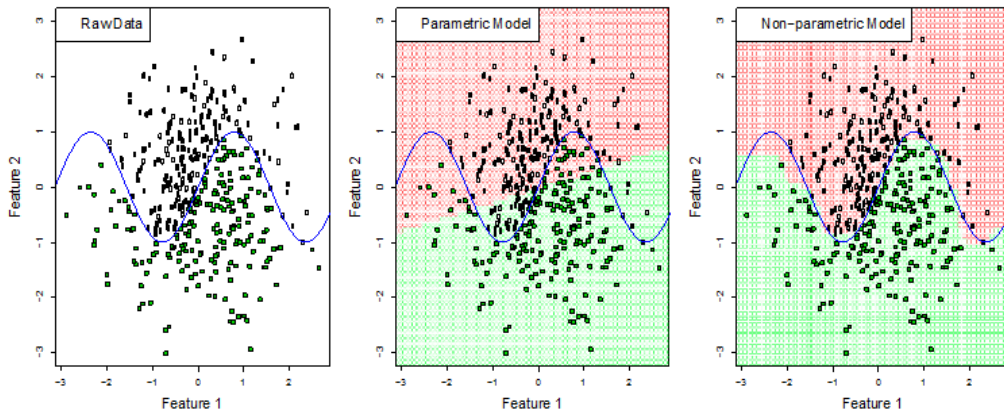


Figure 1.5: Classification in a two-dimensional feature space with a non-linear boundary. Whereas a simple statistical model does quite poorly (center) a non-parametric ML model (right) is able to discover the true class boundary with little effort.

The problem here is that the parametric model is attempting to explain a complicated, non-linear phenomenon with a simple linear model, much like jamming a square peg in a round hole. What we need are more flexible models that can automatically discover complex trends and structure in data without being told what the patterns look like. This is where so-called *non-parametric* machine learning approaches come to the rescue. In the right-hand panel of Figure 1.5, we see the result of applying a non-parametric learning algorithm (in this case, a random forest classifier) to the red/green problem. Clearly, the predicted decision boundary is much closer to the true boundary and as a result, the classification accuracy is much higher than that of the parametric model.

Because they attain such high levels of accuracy on complicated, high-dimensional, real-world data sets, non-parametric ML models are the approach of choice for data-driven problems. Examples of non-parametric approaches include some of the most widely used methods in machine learning, such as k-nearest neighbors, kernel smoothing, support vector machines, decision trees, and ensemble methods. We will describe all of these approaches later in the book.

Returning to the microlending problem, the best choice for scaling up your business is to employ a non-parametric ML model. In addition to providing an automated workflow, you'll also attain higher accuracy, which directly translates to higher business value. For instance, imagine that a non-parametric ML model yields a 25% higher accuracy than a logistic regression approach. In this case, that means that your ML model will make fewer mistakes on new applications: accepting fewer applicants who will not repay their loan and rejecting fewer applicants who would have repaid their loan. Overall, this means a higher average

return on the loans that you do make, enabling you to make more loans overall and to generate higher revenues for your business.

1.1.3 The Five Advantages to Machine Learning

To wrap up our discussion of the microlending example, we list some of the most prominent advantages to using a machine learning system. The five advantages to machine learning over the most common alternatives—manual analysis, hard-coded business rules, and simple statistical models—are that machine learning is:

1. **Accurate** - ML uses data to discover the optimal decision-making engine for your problem. As you collect more data, the accuracy can increase automatically.
2. **Automated** - As new data come in, the output interest can be automatically estimated. This allows users to embed ML directly into an automated workflow.
3. **Fast** - ML can generate answers in a matter of milliseconds as new data stream in, allowing systems to react in real time.
4. **Customizable** - A large number of data-driven problems can be addressed with machine learning. ML models are custom built from your own data, and can be configured to optimize whatever metric drives your business.
5. **Scalable** - As your business grows, ML easily scales to handle increased data rates. Most ML processes are parallelizable, enabling infinite scalability.

In the microlending example, each of these benefits of using a ML system is clear. On accuracy, better decision making leads to a smaller number of good applicants that are rejected and bad applicants that are accepted. On automation, it allows you to process new applications with little manual effort. On speed, it gives you the capability to accept or reject applications more quickly, which benefits your customer and helps attract new applicants. On customizability, it allows you both to optimize whatever the driving factor of your business is and also permits you to use a variety of custom input data into the model to further boost the accuracy. On scalability, it allows you to keep up with increasing demand, growing seamlessly as your business grows.

1.2 The ML workflow: from Data to Deployment

In this section we will introduce the main workflow for integrating machine learning models into your applications and data pipelines. There are 5 main components in the ML workflow: **data** collection; model **building**, **evaluation** and **optimization**; and **prediction** on new data. There is an inherent order to the application of these steps, but most real-world machine learning deployments require us to revisit most of these components multiple times in an iterative process. These 5 components will be detailed in chapters 2 through 5, but we aim to outline them in this introduction and hopefully whet your appetite for getting started. Figure 1 outlines this workflow and the following sections will introduce these concepts from top to bottom. We will be seeing this figure a lot throughout the book as we introduce the various components of the ML workflow.

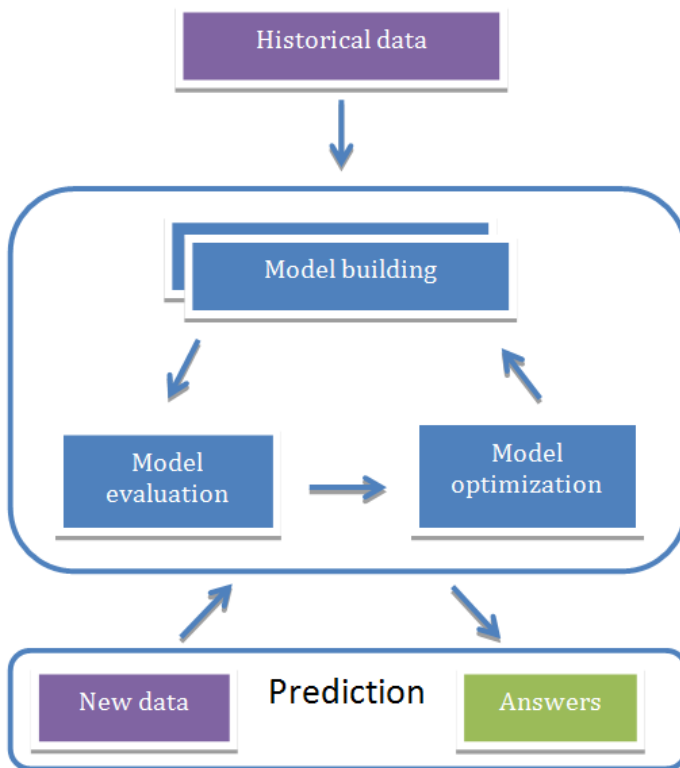
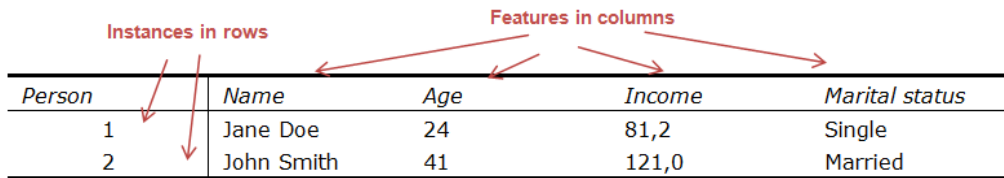


Figure 1.6: The workflow of real-world machine learning systems. On historical input data we enter the model building > evaluation > optimization phase. With the final model we can now make predictions on new data.

1.2.1 Data collection and preparation

Collecting and preparing data for machine learning systems usually entails getting the data into a tabular format, if it's not already. Think of the tabular format as a spreadsheet where data is distributed in rows and columns, each row corresponding to an instance of interest and each column representing some measurement on this instance. There are a few exceptions and variations to this, but it's fair to say that most machine learning algorithms require data in this format. Don't worry; we will deal with the exceptions as we encounter them. Figure 1.6 shows a simple dataset in this format.



<i>Person</i>	<i>Name</i>	<i>Age</i>	<i>Income</i>	<i>Marital status</i>
1	Jane Doe	24	81,2	Single
2	John Smith	41	121,0	Married

Figure 1.7: An example tabular dataset. Rows are called instances of the dataset with instance features in columns.

The first thing to notice about tabular data is that individual columns usually include the same type of data while each row can have columns of various types. In figure 1 we can already identify 4 different types of data: *Name* is a string variable, *Age* is an integer variable, *Income* is a floating point variable, and *Marital status* is a categorical variable (taking on a discrete number of categories). Such a dataset is called heterogeneous (in contrast to homogeneous), and in chapter 2 we will explain how and why we will coerce some of these types of data into other types depending on the particular machine-learning algorithm at hand.

Real-world data can be “messy” in a variety of other ways. Suppose that a particular measurement was unavailable for an instance in the data-gathering phase, and there was no way of going back to find the missing piece of information. In this case our table will contain a **missing value** in one or more cells, and this can complicate both model building and subsequent predictions. In some cases humans are involved in the data-gathering phase, and we all know how easy it is to make mistakes in repetitive tasks such as data recording. This can lead to some of the data being flat-out wrong and we will have to be able to handle such scenarios, or at least know how well a particular algorithm behaves in the presence of **misleading data**. We will look closer at methods for dealing with missing and misleading data in chapter 2, along with other challenging properties of real-world data such as large dataset sizes and continuously streaming data.

1.2.2 Learning a Model from Data

The first part of building a successful machine learning system is to ask a question that can be answered by the data. With our simple *Person* table, we could build an ML model that could predict whether a person is married or single. This information would be useful for showing relevant ads, for example. In this case, we’d use the *Marital status* variable as the **target** and the remaining variables as **features**. The job of the ML algorithm will then be to find how the set of input features can successfully predict the target. Then, for people whose marital status is unknown, we can use the model to predict their marital status based on the input variables for each individual. Figure 1.7 shows this process on our toy dataset.

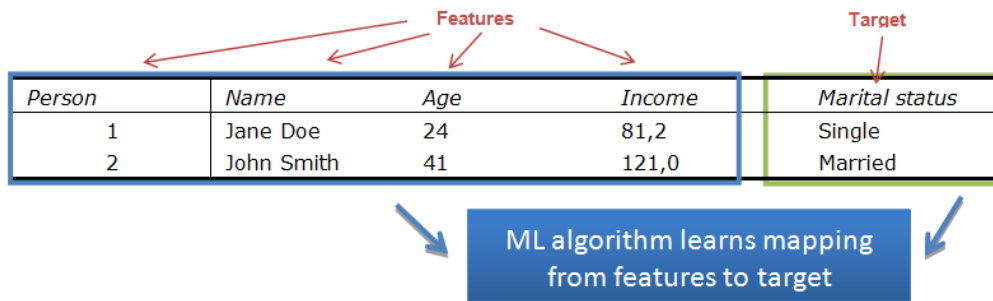


Figure 1.8: The machine learning modeling process.

At this point, we will simply regard the ML algorithm to be a magic box that performs the mapping from input features to output data. To build a useful model, we would of course need more than two rows. One of the advantages of machine learning algorithms, compared with other widely used methods, is the ability to handle a large number of features. In figure 2 there are only four features of which the *Person* ID and *Name* is probably not very useful in predicting the marital status of the person. Some algorithms will be relatively immune to uninformative features, while others may yield higher accuracy if you leave those features out. In chapter 3 we will look closer at the various types of algorithms and their performance on different kinds of problems and datasets.

It is worth noting, however, that valuable information can sometimes be extracted from seemingly uninformative features. A location feature may not be very informative by itself, for example, but can lead to informative features such as population density. This type of data enhancement, called *feature extraction*, is very important in real-world ML projects and will be the topic of chapter 6.

With our ML model in hand we can now make predictions on new data; data where the target variable is unknown. Figure 1.8 shows this process using the magic box model that we built in figure 1.7.

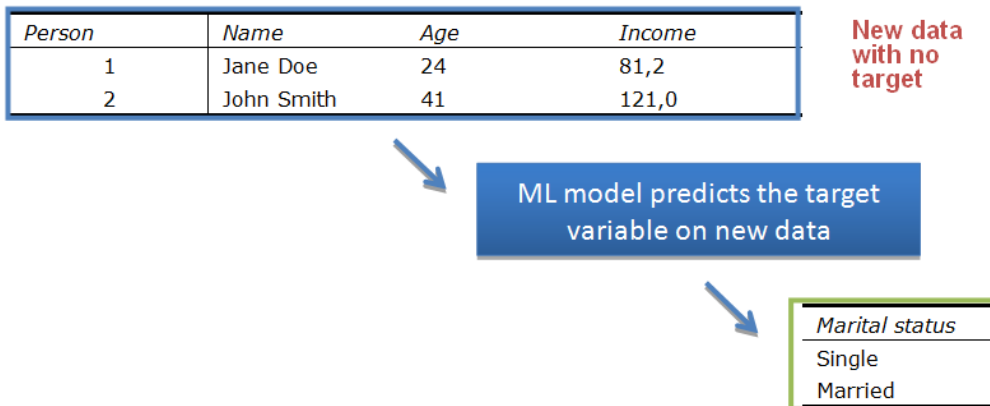


Figure 1.9: Using the model for prediction on new data.

The target predictions are returned in the same form as it appeared in the original data used to learn the model – also known as the *training* data. Using the model to make predictions can be seen as filling out the blank target column of the new data. Some ML algorithms can also output the probabilities associated with each class. In our married/single example, a *probabilistic* ML model would output two values for each new person: the probability of this person being married and the probability of the person being single.

We left out a few details on the way here, but in principle we have just built our first ML system. Every machine learning system is about building models and using those models to make predictions. Let's take a look at the basic machine learning workflow in pseudo-code, just to get another view of how simple it actually is:

Code listing 1.1: Initial structure of an ML workflow program

```
data = load_data("data/people.csv")
model = build_model(data, target="Marital status")
new_data = load_data("data/new_people.csv")
predictions = model.predict(new_data)
```

Of course, we have not filled out any of the functions yet, but the basic structure is in place. By chapter 3 we will understand most of these functions and the rest of the book - chapters 4 through 9 - is about making sure we are building the very best model for the problem at hand.

1.2.3 Evaluating Model Performance

Very rarely is an ML system put to use without some kind of validation of the performance of the model. Even though we've skipped a lot of details in this chapter, let's pretend that we know how to build a model and make predictions. We can now apply a clever trick to get some sense of how well our model is actually working before we use it to predict on new

data. We simply take out some of the training data and pretend that we don't know the target variable. We then build a model on the remaining data and use the held-out data (testing data) to make predictions. Simple, right? Figure 1.9 visualizes this model testing process.

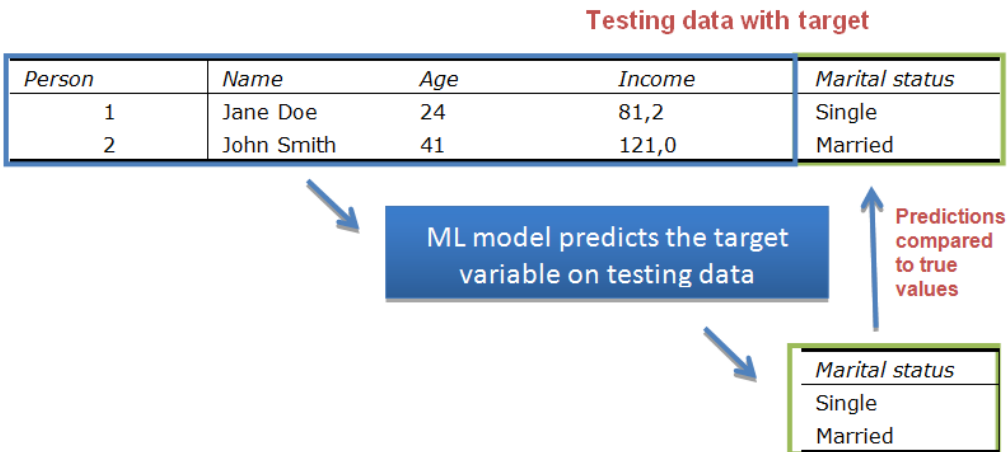


Figure 1.10: Using a testing set for model performance evaluation. We “pretend” that the target variable is unknown and compare the predictions with the true values.

Let’s also look at the pseudo-code for this workflow:

Code listing 1.2: Our ML workflow program with model evaluation

```
data = load_data(...)
training_data, testing_data = split_data(data)
model = build_model(training_data, target="Marital status")
true_values = testing_data.extract_column("Marital status")
predictions = model.predict(testing_data)
accuracy = compare_predictions(predictions, true_values)
```

We can now compare the predicted results with the known “true” values to get a feeling for how accurate the model is. In the pseudo-code this functionality is hidden behind the `compare_predictions` function, and most of chapter 4 is dedicated to understanding how this function looks for different types of machine learning problems.

1.2.4 Optimizing Model Performance

The last piece of the essential machine-learning puzzle is covered in chapter 5: how to use the results of our model evaluation to go back and actually make the model better. There are generally 3 main ways of achieving better model accuracy, all of which we will consider in much more details in chapter 5.

- **Tuning the model parameters.** ML algorithms are configured with a number of parameters specific to the underlying algorithm, and the optimal value of these parameters often depends on the type and structure of the data. The value of each of these parameters, or any of them combined, can have a large impact on the performance of the model. We will introduce various ways to find and select the best parameter values, and show how this can help in determining the best algorithm for the dataset in question.
- **Selecting a subset of features.** Because many ML problems include a large number of features, it can sometimes be hard for the algorithm to find the real signal in the data because of the noise from all of the features, even though they might still be informative on their own. For many ML problems it is a good thing to have a lot of data, but it can sometimes be a curse. And because we don't know beforehand when this will affect our model performance, we have to carefully determine the features that make up the most general and accurate model.
- **Pre-processing the data.** If you search on the Internet for machine learning datasets, you will find easy-to-use datasets that many ML algorithms can be quickly applied to. Most real-world datasets, however, are not in such a clean state and you will have to perform some amount of cleaning and processing, a process widely referred to as *data munging*. The dataset may include names that are spelled differently, although they refer to the same entity, or have missing or incorrect values, and all these things can hurt the performance of the model.
- *Feature engineering*, the extraction of value from what could otherwise be considered uninformative features, is another kind of data pre-processing that can radically improve model performance. In any ML problem there is valuable domain-knowledge to be extracted from the data. If you know how a biological system works, for example, you have a better chance at knowing what variables or derived variables might be important for predicting a specific target in a biological system.

With the machine learning essentials in place, we will look briefly towards more advanced features in the next section before we go into some more details with the main components covered in this section.

1.3 *Boosting Model Performance with Advanced Techniques*

In the last section we introduced the essential steps in any real-world machine-learning project, and we will now look at some additional techniques that are often used to improve the model performance even further. Depending on the data and problem at hand, some of these techniques can provide significant gains in accuracy, but sometimes at the cost of speed in both training and prediction. These sections will be explained in more details in chapters 6 through 9, but we will use this section to outline the main ideas.

1.3.1 Data Pre-processing and Feature Engineering

We will look at different kinds of data and how to deal with common types of “messy”-ness in chapter 2, but in addition to this quite essential data cleaning, we can go a step further and extract **additional value** from the data that might improve our model performance. In any problem domain there will be specific knowledge that goes into deciding the data to collect, but this valuable **domain knowledge** can also be used to extract value from the collected data, in effect adding to the features of the model before model building. We call this process *feature engineering* and when the previously introduced essential ML workflow has become second nature to you, you can find yourself spending almost all your time in this part of the optimization process. This is also the **creative** part of machine learning, where you get to use your knowledge and imagination to come up with ways to improve the model by digging into the data and extracting hidden value, but we will make extensive use of our statistically validated model evaluation and optimization steps in order to distinguish what seemed like a good idea and what is actually useful. Here are a few important examples of feature engineering:

- **Dates and times.** You'll see a date or time variable in many datasets, but by themselves they are not really useful for ML algorithms which tends to require raw numbers or categories. The information might be valuable, though. If you wanted to predict which ad to show, it will certainly be important to know the time of day, the day of the week and the time of year. With feature engineering, this information can be extracted from the dates and made available to the model.
- **Location.** Location-data such as latitude/longitude coordinates or location names are available in some datasets. This information can sometimes be used by itself, but we may be able to extract additional information that is useful for the specific problem. For example, if we wanted to predict election results in a county we might want to extract the population density, mean income, poverty rate etc. to use as numbers in our model.
- Digital media data such as **text and documents** and **images and video**. All of these are not directly usable by the ML algorithms, but can be enabled by feature engineering.

Hopefully it's clear that feature engineering can be important for real-world ML projects, but we will go into much more detail in chapter 6. We will introduce specific feature engineering techniques, like the ones listed above, but we will also talk about how it all feeds into our ML workflow so our model performance improves without becoming too complex and be prone to over-fitting. Figure 1.10 illustrates the features engineering integration into the larger ML workflow introduced in section 1.2:

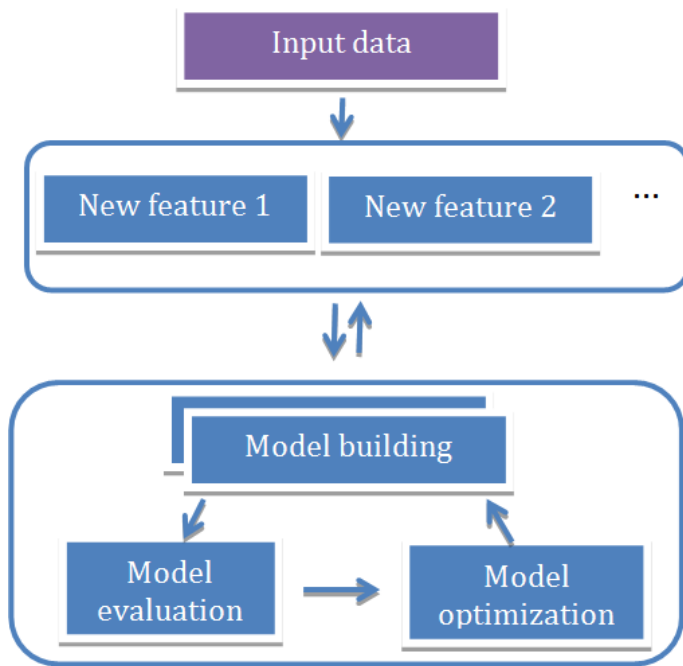


Figure 1.11: Feature engineering phase inserted in the original ML workflow.

1.3.2 Data Exploration and Unsupervised Learning

In the previous section we talked about using domain knowledge to extract additional value from the data and improve model building. In this section we will explore the possibility of using the raw data itself to discover new structure that can improve model performance and reveal valuable insight. It is said that there's no such thing as a free lunch, so it's understandable if you're skeptical about this. However, as we'll see here and in more details in chapter 7, there are many well-known and some recent techniques that we can consider for this strictly data-based (in contrast to knowledge-based) exploration and value-extraction process.

Arguably some of the most well known types of data-based exploration and value-extraction is what we call **dimensionality reduction**. Remember how we sometimes call our data columns or features for dimensions? Imagine that you wanted to plot person information in a graph. You may plot the age on the x-axis and the height on the y-axis. You could add a third dimension and plot the income on the z-axis, but then you've basically run out of dimensions that can be grasped by humans. This might be OK for datasets with relatively few dimensions, but when you are dealing with thousands or even millions of dimensions, which is very common in real-world machine-learning, it is clear that we need to deal with this limitation in clever ways in order to visualize the dataset. But although

computers are not bound to our three dimensions in a data-handling sense, the same underlying limitations apply to ML algorithms: the larger the dimensionality, the harder it will be to find meaningful signals. Dimensionality reduction attempts to solve this problem by reducing many dimensions to a few meaningful ones, providing a way to visualize large-dimensional data, but also a set of new features that makes it easier for the ML model to find the signal, thus increasing the performance. Simple, right? Don't worry if your head hurts by now, we will go more carefully through these ideas in chapter 7.

Another related technique is known as **clustering**. Clustering can connect rows of the data into groups based on the raw columns or features. We can use this to discover natural groupings of the data in the same way as humans often do. Imagine being presented with a list of 100 movies and asked to group them into maximally 5 piles. You'd probably identify main genres and other factors that are important for you while you sift through the list and build out the piles. Now what if you had a million movies? It would take you a long time to do the sorting, and this is where clustering algorithms can help. By identifying the features that should be used for the input data, and running it through the clustering algorithm, the computer may be able to identify some or all of the piles, and may even discover some that you hadn't. While clustering can be used for much more diverse problems, this example is good because it is something that many of us has already seen at work. Netflix is a company that makes money by streaming a huge library of movies, and they used clustering to identify interesting piles for the movies. They ran the Netflix Prize, a competition to improve their movie recommendations. Figure 1.11 shows an example of connected movies from one of the participants.

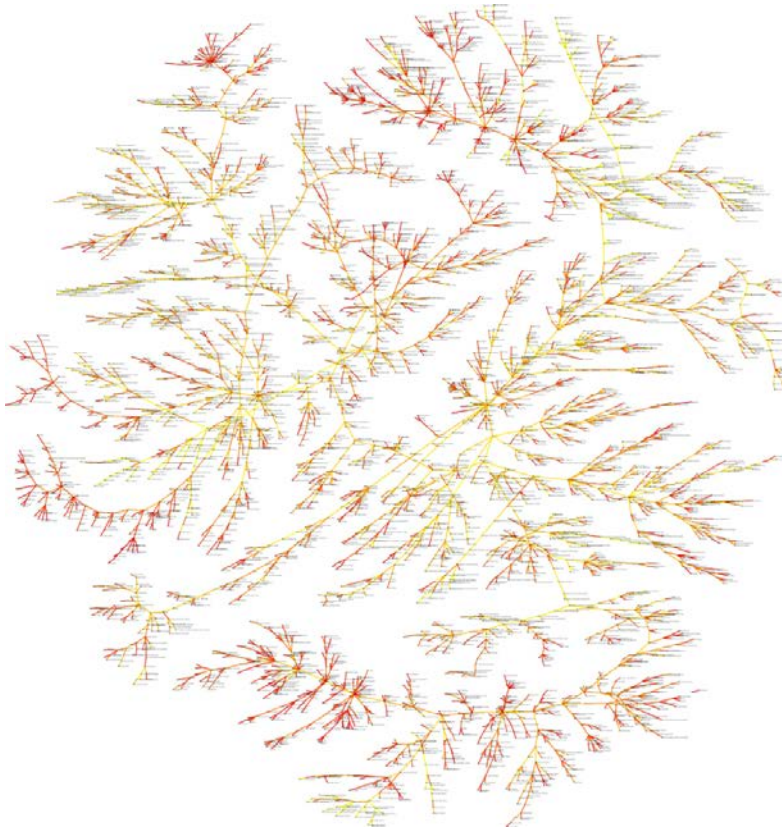


Figure 1.12: Machine learned connections between 5000 Netflix movies. The goal is to predict Netflix user ratings from a training set of over 100 million examples. Image by chef_ele on Flickr (http://www.flickr.com/photos/chef_ele)

Methods like these are collectively known as **unsupervised learning** and can be used stand-alone, in essence as data exploration processes that reveal structure in the data and have value in themselves, but they can also be successfully used for the predictive modeling workflow that we've introduced in the previous sections. Netflix is again a good example, as the clusters identified are used stand-alone to improve the user experience, but they are also used to improve movie recommendations for users, i.e. predicting whether a user would be interested in a movie or not.

1.3.3 Improving models with time with online methods

Most traditional ML models are static or only rarely rebuilt. But in many cases we will have data and predictions flowing back into the system, and we want the model to improve with time and adapt to changes in the data. Several ML algorithms support this type of **online**

learning and we will introduce these and potential pitfalls in chapter 8. Figure 1.12 shows how continuous re-learning would be integrated into the ML workflow.

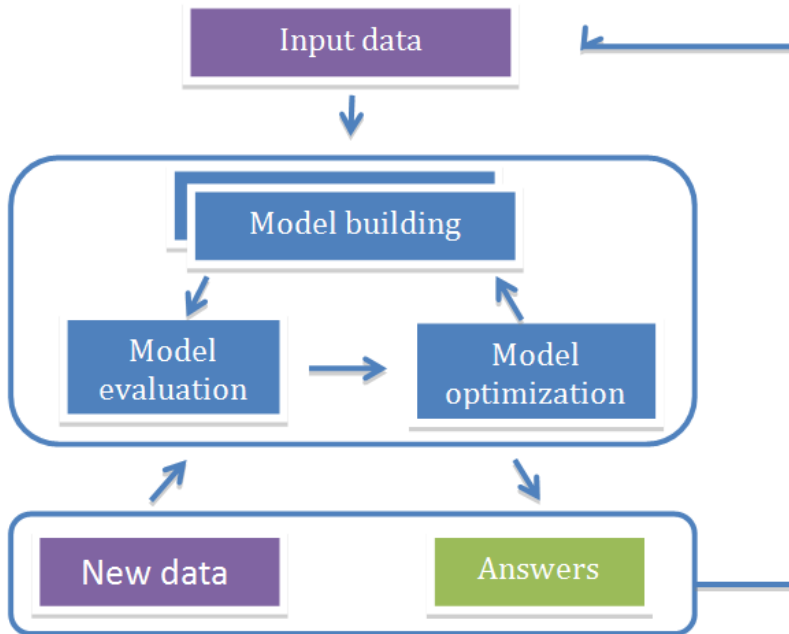


Figure 1.13: Flow of an online ML system where predictions are fed back to the model for iterative improvements.

1.3.4 Scaling Models with Data Volume and Velocity

It is well known that datasets are increasing in size and velocity at a higher pace than ever. Datasets for supervised methods, where the target answers are in the training set, have traditionally not been very large because humans were needed in order to acquire the answers. Today, a lot of data (including answers) are produced directly by sensors, machines or computers, and we are beginning to see requirements for scalable ML algorithms in order to handle these data volumes.

In chapter 9 we will go into details with the machine learning methods that are capable of scaling with growing dataset sizes, and how they compare to each other and non-scaling algorithms.

1.4 Summary

In this chapter, we introduced machine learning as a better, more data-driven approach to making decisions. The main points to take away from this chapter are:

- With a real-world example of deciding which loan applicants should be approved, we

explored several alternative approaches to data analysis. Ultimately, machine learning was the superior approach due in large part to the five advantages of ML: it is more **accurate**, **automated**, **fast**, **customizable**, and **scalable**. We will see these five properties in action throughout the book.

- We introduced basic machine learning **workflow**, which consists of data collection, model building, model evaluation and optimization, and prediction on new data. In the next four chapters, we will dive into these topics and provide the reader with the essential know-how to get their own ML project up and running.
- Beyond the basic ML workflow, we gave a hint at the more advanced material in chapters 6-9, which describe methods of **boosting the performance** of real-world ML systems. These chapters arm the reader with the tools needed to more highly leverage training data to attain higher levels of accuracy and to use modern compute infrastructures to beef up the speed and scalability of ML systems.

To conclude the chapter, we provide the reader with a set of example machine learning use-cases in Table 1.1. This list is not meant to be exhaustive, but rather is intended to show the broad applicability of supervised machine learning. Indeed, real-world machine learning is a driving force in today's economy.

Table 1.1: Use cases for supervised machine learning, organized by the type of problem.

Problem	Description	Example Use Cases
Classification	Determine the discrete class to which each individual belongs, based on input data	spam filtering, sentiment analysis, fraud detection, customer ad targeting, churn prediction, support case flagging, content personalization, detection of manufacturing defects, customer segmentation, event discovery, genomics, drug efficacy
Regression	Predict the real-valued output for each individual, based on input data	stock market prediction, demand forecasting, price estimation, ad bid optimization, risk management, asset management, weather forecasting, sports prediction
Recommendation	Predict which alternatives a user would prefer	product recommendation, job recruiting, 'Netflix prize'
Imputation	Infer the values of missing input data	incomplete patient medical records, missing customer data, census data