2

Machine Learning Lifecycle

Imagine that you are a project manager on the innovation team of m-Udhār Solar, a (fictional) pay-as-you-go solar energy provider to poor rural villages that is struggling to handle a growing load of applications. The company is poised to expand from installing solar panels in a handful of pilot districts to all the districts in the state, but only if it can make loan decisions for 25 times as many applications per day with the same number of loan officers. You think machine learning may be able to help.

Is this a problem to address with machine learning? How would you begin the project? What steps would you follow? What roles would be involved in carrying out the steps? Which stakeholders' buy-in would you need to win? And importantly, what would you need to do to ensure that the system is *trustworthy*? Making a machine learning system trustworthy should not be an afterthought or add-on, but should be part of the plan from the beginning.

The end-to-end development process or *lifecycle* involves several steps:

1. problem specification,
2. data understanding,
3. data preparation,
4. modeling,
5. evaluation, and
6. deployment and monitoring.

Narrow definitions consider only the modeling step to be the realm of machine learning. They consider the other steps to be part of the broader endeavor of data science and engineering. Most books and research on machine learning are similarly focused on the modeling stage. However, you cannot really execute the development and deployment of a trustworthy machine learning system without focusing on all parts of the lifecycle. There are no shortcuts. This chapter sketches out the master plan.

2.1 A Mental Model for the Machine Learning Lifecycle

The six steps of the machine learning lifecycle given above, also illustrated in Figure 2.1, are codified in the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology. This is the mental model to keep in mind of how machine learning systems should be developed and deployed. Although the flow largely proceeds sequentially through the steps, there are several opportunities to go back and redo earlier steps. This description is stylized; even good examples of real-world lifecycles are messier, but the main idea continues to hold.

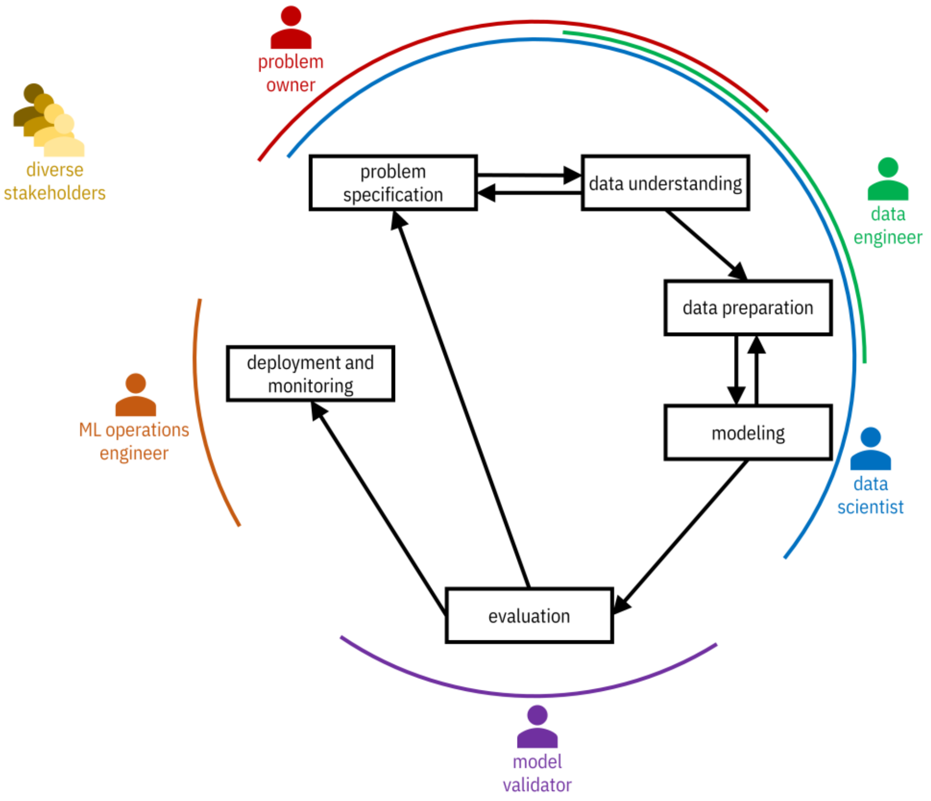

Figure 2.1. *Steps of the machine learning lifecycle codified in CRISP-DM. Different personas participate in different phases of the lifecycle.* Accessible caption. A series of six steps arranged in a circle: (1) problem specification; (2) data understanding; (3) data preparation; (4) modeling; (5) evaluation; (6) deployment and monitoring. There are some backward paths: from data understanding to problem specification; from modeling to data preparation; from evaluation to problem specification. Five personas are associated with different steps: problem owner with steps 1–2; data engineer with steps 2–3; data scientist with steps 1–4; model validator with step 5; ML operations engineer with step 6. A diverse stakeholders persona is on the side overseeing all steps.

Because the modeling stage is often put on a pedestal, there is a temptation to use the analogy of an onion in working out the project plan: start with the core modeling, work your way out to data understanding/preparation and evaluation, and then further work your way out to problem specification and deployment/monitoring. This analogy works well for a telecommunications system for example,[1] both pedagogically and in how the technology is developed, but a sequential process is more appropriate for a trustworthy machine learning system. Always start with understanding the use case and specifying the problem.

> "People are involved in every phase of the AI lifecycle, making decisions about which problem to address, which data to use, what to optimize for, etc."
>
> —Jenn Wortman Vaughan, research scientist at Microsoft

The different steps are carried out by different parties with different personas including problem owners, data engineers, data scientists, model validators, and machine learning (ML) operations engineers. Problem owners are primarily involved with problem specification and data understanding. Data engineers work on data understanding and data preparation. Data scientists tend to play a role in all of the first four steps. Model validators perform evaluation. ML operations engineers are responsible for deployment and monitoring.

Additional important personas in the context of trustworthiness are the potential trustors of the system: human decision makers being supported by the machine learning model (m-Udhār loan officers), affected parties about whom the decisions are made (rural applicants; they may be members of marginalized groups), regulators and policymakers, and the general public. Each stakeholder has different needs, concerns, desires, and values. Systems must meet those needs and align with those values to be trustworthy. Multi-stakeholder engagement is essential and cannot be divorced from the technical aspects of design and development. Documenting and transparently reporting the different steps of the lifecycle help build trust among stakeholders.

2.2 Problem Specification

The first step when starting the development of a machine learning system is to define the problem. What is the problem owner hoping to accomplish and why? The director of m-Udhār Solar wishes to automate a task that is cumbersome and costly for people to do. In other scenarios, problem owners may want to augment the capabilities of human decision makers to improve the quality of their decisions. They may have other goals altogether. In some cases, the problem should not even be solved to begin with, because doing so may cause or exacerbate societal harms and breach the lines of ethical behavior.[2] A *harm* is an outcome with a severe unwanted effect on a person's life. This definition of harm is made more precise in Chapter 3. Let's repeat this important point: do not solve problems that would lead to harms for someone or some group.

---

[1]C. Richard Johnson, Jr. and William A. Sethares. *Telecommunication Breakdown: Concepts of Communication Transmitted via Software-Defined Radio*. Upper Saddle River, New Jersey, USA: Prentice Hall, 2003.
[2]Andrew D. Selbst, danah boyd, Sorelle A. Friedler, Suresh Venkatasubramanian, and Janet Vertesi. "Fairness and Abstraction in Sociotechnical Systems." In: *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency*. Barcelona, Spain, Jan. 2020, pp. 59–68.

"We all have a responsibility to ask not just, 'can we do this?', but 'should we do this?'"

—Kathy Baxter, ethical AI practice architect at Salesforce

Problem identification and understanding is best done as a dialogue between problem owners and data scientists because problem owners might not have the imagination of what is possible through machine learning and data scientists do not have a visceral understanding of the pain points that problem owners are facing. Problem owners should also invite representatives of marginalized groups for a seat at the problem understanding table to voice their pain points.[3] Problem identification is arguably the most important and most difficult thing to do in the entire lifecycle. An inclusive design process is imperative. Finding the light at the end of the tunnel is actually not that hard, but finding the tunnel can be very hard. The best problems to tackle are ones that have a benefit to humanity, like helping light up the lives and livelihoods of rural villagers.

Once problem owners have identified a problem worth solving, they need to specify metrics of success. Being a social enterprise, the metric for m-Udhār Solar is number of households served with acceptable risk of defaulting. In general, these metrics should be in real-world terms relevant to the use case, such as lives saved, time reduced, or cost avoided.[4] The data scientist and problem owner can then map the real-world problem and metrics to machine learning problems and metrics. This specification should be as crisp as possible, including both the quantities to be measured and their acceptable values.

The goals need not be specified only as traditional key performance indicators, but can also include objectives for maintenance of performance across varying conditions, fairness of outcomes across groups and individuals, resilience to threats, or number of insights provided to users. Defining what is meant by fairness and specifying a threat model are part of this endeavor. For example, m-Udhār aims not to discriminate by caste or creed. Again, these real-world goals must be made precise through a conversation between problem owners, diverse voices, and data scientists. The process of eliciting objectives is known as *value alignment*.

One important consideration in problem scoping is resource availability, both in computing and human resources. A large national or multinational bank will have many more resources than m-Udhār Solar. A large technology company will have the most of all. What can reasonably be accomplished is gated by the skill of the development team, the computational power for training models and evaluating new samples, and the amount of relevant data.

Machine learning is not a panacea. Even if the problem makes sense, machine learning may not be the most appropriate solution to achieve the metrics of success. Oftentimes, back-of-the-envelope calculations can indicate the lack of fit of a machine learning solution before other steps are undertaken. A common reason for machine learning to not be a viable solution is lack of appropriate data, which brings us to the next step: data understanding.

[3]Meg Young, Lassana Magassa, and Batya Friedman. "Toward Inclusive Tech Policy Design: A Method for Underrepresented Voices to Strengthen Tech Policy Documents." In: *Ethics and Information Technology* 21.2 (Jun. 2019), pp. 89–103. The input of diverse stakeholders, especially those from marginalized groups, should be monetarily compensated.
[4]Kiri L. Wagstaff. "Machine Learning that Matters." In: *Proceedings of the International Conference on Machine Learning*. Edinburgh, Scotland, UK, Jun.–Jul. 2012, pp. 521–528.

2.3 Data Understanding

Once the problem has been identified and specified, a relevant dataset must be collected. In instances where the problem is to automate an existing decision-making process, identifying the relevant dataset is fairly straightforward. M-Udhār's dataset consists of attributes and other inputs that loan officers used to make decisions in the past, along with their decisions. The inputs constitute the *features* and the historical decisions constitute the *labels* for a supervised machine learning task. But there may also be data that loan officers did not use that could be leveraged by a machine learning system. A promise of so-called 'big data' is the inclusion of large sets of attributes, many weakly correlated to the label, that would overwhelm a person but not a machine. For the machine learning system to make even better decisions than people, true outcomes rather than decisions should ideally be the labels, e.g. whether an applicant defaulted on their loan in the future rather than the approval decision.

Machine learning can also be applied in use cases that are new processes for an organization and no exact historical data exists. Here, proxy data must be identified. For example, a health system may wish to start offering home nursing care to indisposed individuals proactively, but may not have data directly applicable for understanding this decision. Data from previous interactions of patients with the health system may be used as a proxy. In other cases, it may be that new data must be collected. In yet other cases, it may be that relevant data neither exists nor can be collected, and the problem must be specified differently.

Once a dataset has been identified or collected, it is critical for the data scientist and data engineer to understand the semantics of the various features and their values by consulting the problem owner and other subject matter experts as well as by consulting a *data dictionary* (a document describing the features) if one exists. They should also conduct exploratory data analysis and visualization. This understanding can help identify problems in the data such as *leakage*, the presence of information in the features helpful in predicting the label that would not be available in new inputs to a deployed system, and various forms of *bias*. One important form of bias is *social bias* in which a proxy for the label does not well-reflect the true label of interest. For example, using past number of doctor visits may not be a good proxy of how sick an individual is if there are socioeconomic reasons why some people visit the doctor more than others at the same level of ill health. A similar social bias stems from prejudice: labels from historical human decisions contain systematic differences across groups. Other important biases include *population bias*: the dataset underrepresents certain inputs and overrepresents others, and *temporal bias*: issues stemming from the timing of data collection.

The data understanding stage also requires the development team to consider data usage issues. Just because features are available (and may even improve the performance of a model), that does not mean that they can and should be used. Use of certain features may be prohibited by law, be unethical, or may not have appropriate consent in place. For example, m-Udhār Solar may have the surname of the applicant available, which indicates the applicant's caste and religion and may even help a model achieve better performance, but it is unethical to use. The use of other features may pose privacy risks. A more detailed treatment of data-related issues is presented in Part 2 of the book.

2.4 Data Preparation

Data integration, data cleaning, and feature engineering constitute the data preparation step of the lifecycle. The end goal of this stage is a final training dataset to be used in modeling. Starting from the insights gleaned in the data understanding phase, *data integration* starts by extracting, transforming, and loading (ETL) data from disparate relevant databases and other data sources. Next, the data from the disparate sources is joined into a single dataset that is maintained in a format amenable to downstream modeling. This step is most challenging when dealing with humongous data sources.

Data cleaning is also based on data understanding from the previous stage. Some of the key components of data cleaning are:

- filling in missing values (known as *imputation*) or discarding them,
- binning continuous feature values to account for outliers,
- grouping or recoding categorical feature values to deal with rarely occurring values or to combine semantically similar values, and
- dropping features that induce leakage or should not be used for legal, ethical, or privacy reasons.

Feature engineering is mathematically transforming features to derive new features, including through interactions of several raw features. Apart from the initial problem specification, feature engineering is the point in the lifecycle that requires the most creativity from data scientists. Data cleaning and feature engineering require the data engineer and data scientist to make many choices that have no right or wrong answer. Should m-Udhār's data engineer and data scientist group together any number of motorcycles owned by the household greater than zero? How should they encode the profession of the applicant? The data scientist and data engineer should revisit the project goals and continually consult with subject matter experts and stakeholders with differing perspectives to help make appropriate choices. When there is ambiguity, they should work towards safety, reliability, and aligning with elicited values.

2.5 Modeling

The modeling step receives a clear problem specification (including metrics of success) and a fixed, clean training dataset. A mental model for trustworthy modeling includes three main parts:

1. pre-processing the training data,
2. training the model with a machine learning algorithm, and
3. post-processing the model's output predictions.

This idea is diagrammed in Figure 2.2. Details of this step will be covered in depth throughout the book, but an overview is provided here.

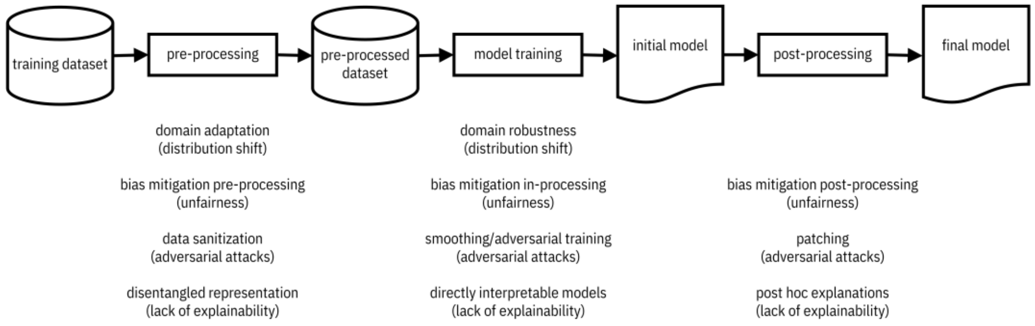
training dataset
pre-processing
pre-processed dataset
model training
initial model
post-processing
final model
domain adaptation (distribution shift)
bias mitigation pre-processing (unfairness)
data sanitization (adversarial attacks)
disentangled representation (lack of explainability)
domain robustness (distribution shift)
bias mitigation in-processing (unfairness)
smoothing/adversarial training (adversarial attacks)
directly interpretable models (lack of explainability)
bias mitigation post-processing (unfairness)
patching (adversarial attacks)
post hoc explanations (lack of explainability)

Figure 2.2. *Main parts of trustworthy machine learning modeling. Distribution shift, unfairness, adversarial attacks, and lack of explainability can be mitigated using the various techniques listed below each part. Details of these methods are presented in the remainder of the book.* Accessible caption. A block diagram with a training dataset as input to a pre-processing block with a pre-processed dataset as output. The pre-processed dataset is input to a model training block with an initial model as output. The initial model is input to a post-processing block with a final model as output. The following techniques are examples of pre-processing: domain adaptation (distribution shift); bias mitigation pre-processing (unfairness); data sanitization (adversarial attacks); disentangled representation (lack of explainability). The following techniques are examples of model training: domain robustness (distribution shift); bias mitigation in-processing (unfairness); smoothing/adversarial training (adversarial attacks); directly interpretable models (lack of explainability). The following techniques are examples of post-processing: bias mitigation post-processing (unfairness); patching (adversarial attacks); post hoc explanations (lack of explainability).

Different from data preparation, data *pre-processing* is meant to alter the statistics or properties of the dataset to achieve certain goals. *Domain adaptation* overcomes a lack of robustness to changing environments, including temporal bias. *Bias mitigation pre-processing* changes the dataset to overcome social bias and population bias. *Data sanitization* aims to remove traces of data poisoning attacks by malicious actors. Learning *disentangled representations* overcomes a lack of human interpretability of the features. All should be performed as required by the problem specification.

The main task in the modeling step is to use an algorithm that finds the patterns in the training dataset and generalizes from them to fit a model that will predict labels for new unseen data points with good performance. (The term *predict* does not necessarily imply forecasting into the future, but simply refers to providing a guess for an unknown value.) There are many different algorithms for fitting models, each with a different *inductive bias* or set of assumptions it uses to generalize. Many machine learning algorithms explicitly minimize the objective function that was determined in the problem specification step of the lifecycle. Some algorithms minimize an approximation to the specified objective to make the mathematical optimization easier. This common approach to machine learning is known as *risk minimization*.

The *no free lunch theorem* of machine learning says that there is no one best machine learning algorithm for all problems and datasets.[5] Which one works best depends on the characteristics of the

[5]David H. Wolpert. "The Lack of A Priori Distinctions Between Learning Algorithms." In: *Neural Computation* 8.7 (Oct. 1996), pp. 1341–1390.

dataset. Data scientists try out several different methods, tune their parameters, and see which one performs best empirically. The empirical comparison is conducted by randomly splitting the training dataset into a training partition on which the model is fit and a testing partition on which the model's performance is validated. The partitioning and validation can be done once, or they can be done several times. When done several times, the procedure is known as *cross-validation*; it is useful because it characterizes the stability of the results. Cross-validation should be done for datasets with a small number of samples.

The basic machine learning algorithm can be enhanced in several ways to satisfy additional objectives and constraints captured in the problem specification. One way to increase reliability across operating conditions is known as *domain robustness*. Machine learning algorithms that reduce unwanted biases are known as *bias mitigation in-processing*. One example category of methods for defending against data poisoning attacks is known as *smoothing*. A defense against a different kind of adversarial attack, the evasion attack, is *adversarial training*. Certain machine learning algorithms produce models that are simple in form and thus *directly interpretable* and understandable to people. Once again, all of these enhancements should be done according to the problem specification.

Post-processing rules change the predicted label of a sample or compute additional information to accompany the predicted label. Post-processing methods can be divided into two high-level categories: *open-box* and *closed-box*. Open-box methods utilize information from the model such as its parameters and functions of its parameters. Closed-box methods can only see the output predictions arising from given inputs. Open-box methods should be used if possible, such as when there is close integration of model training and post-processing in the system. In certain scenarios, post-processing methods, also known as *post hoc* methods, are isolated from the model for logistical or security reasons. In these scenarios, only closed-box methods are tenable. Post-processing techniques for increasing reliability, mitigating unwanted biases, defending against adversarial attacks, and generating explanations should be used judiciously to achieve the goals of the problem owner. For example, post hoc explanations are important to provide to m-Udhār Solar's loan officers so that they can better discuss the decision with the applicant.

The specification of certain use cases calls for *causal* modeling: finding generalizable instances of cause-and-effect from the training data rather than only correlative patterns. These are problems in which input interventions are meant to change the outcome. For example, when coaching an employee for success, it is not good enough to identify the pattern that putting in extra hours is predictive of a promotion. Good advice represents a causal relationship: if the employee starts working extra hours, then they can expect to be promoted. It may be that there is a common cause (e.g. conscientiousness) for both doing quality work and working extra hours, but it is only doing quality work that causes a promotion. Working long hours while doing poor quality work will not yield a promotion; causal modeling will show that.

2.6 Evaluation

Once m-Udhār's data scientists have a trained and tested model that they feel best satisfies the problem owner's requirements, they pass it on to model validators. A model validator conducts further independent testing and evaluation of the model, often with a completely separate *held-out* dataset that the data scientist did not have access to. It is important that the held-out set not have any leakage from the training set. To stress-test the model's safety and reliability, the model validator can and should evaluate it on data collected under various conditions and data generated to simulate unlikely events.

The model validator persona is part of *model risk management*. Model risk is the chance of decisions supported by statistical or machine learning models yielding gross harms. Issues can come from any of the preceding lifecycle steps: from bad problem specification to data quality problems to bugs in the machine learning algorithm software. Even this late in the game, it is possible that the team might have to start over if issues are discovered. It is only after the model validator signs off on the model that it is put into production. Although not standard practice yet in machine learning, this 'signing off' can be construed as a *declaration of conformity*, a document often used in various industries and sectors certifying that a product is operational and safe.

2.7 Deployment and Monitoring

The solar panels are loaded on the truck and the electricians are just waiting to find out which households to install them at. The last step on the long road to the productive use of the machine learning system is finally here! The ML operations engineer takes center stage to deploy the model. Starting with a certified model, there are still questions to be answered. What infrastructure will bring new data to the model? Will predictions be made in batch or one-by-one? How much latency is allowable? How will the user interact with the system? The engineer works with different stakeholders to answer the questions and implements the infrastructure to meet the needs, resulting in a deployed model.

Important for making the model trustworthy, the ML operations engineer must also implement tools to monitor the model's performance to ensure it is operating as expected. As before, performance includes all relevant metrics of success in the problem specification, not only traditional key performance indicators. The performance of trained models can degrade over time as the incoming data statistically drifts away from the training data. If drift is detected, the monitoring system should notify the development team and other relevant stakeholders. All four attributes of trustworthiness (basic performance, reliability, human interaction, and aligned purpose) permeate throughout the machine learning lifecycle and must be accounted for in the development, deployment, and monitoring plan from the beginning to the end. M-Udhār Solar has now deployed its loan origination automation system and is able to easily serve applicants not just in one entire state, but a few neighboring ones as well.

2.8 Summary

- The machine learning lifecycle consists of six main sequential steps: (1) problem specification, (2) data understanding, (3) data preparation, (4) modeling, (5) evaluation, and (6) deployment and monitoring, performed by people in different roles.
- The modeling step has three parts: (1) pre-processing, (2) model training, and (3) post-processing.
- To operationalize a machine learning system, plan for the different attributes of trustworthiness starting from the first step of problem specification. Considerations beyond basic performance should not be sprinkled on at the end like pixie dust, but developed at every step of the way with input from diverse stakeholders, including affected users from marginalized groups.