Chapter 8. Putting It All Together

In this book, we have given you, the ML practitioner, a framework for how to use Explainable AI (XAI) and where it can be best applied. We also gave you a toolbox of explainable techniques to apply in different scenarios, and guides for crafting responsible and beneficial interactions with explanations. In this chapter, we step back to focus on the bigger picture around Explainable AI. With the tools and capabilities that we covered in this book, how can you approach the entire ML workflow and build with explainability in mind? We also provide a preview of the upcoming AI regulations and standards that will require explainability.

Building with Explainability in Mind

Many times, explainability is approached as an afterthought to model development, an added bonus to developing your most recent top-performing model or a post hoc feature request required by your boss trying to adhere to some new regulatory constraint that's been imposed on the business. However, explainability and the goal of XAI is much more than that.

Throughout this book, we've discussed in detail a number of explainability techniques and seen how they can be applied for tabular, image, and text data. For the most part, we've explored these techniques in isolation so you, the reader, can quickly get up to speed on commonly used methods, how they work, their pros, and their cons. Of course, in practice explainability doesn't occur in isolation. You should consider these techniques as part of your ever-expanding toolkit for analysis in machine learning. This toolkit isn't restricted merely to post hoc model analysis. The true benefit of Explainable AI is that it can be applied across the entire end-to-end machine learning life cycle from collecting, preprocessing, and improving datasets to model development, deployment, and monitoring.

In this section, we'll discuss at a high level how to build more effective machine learning solutions by incorporating the methodology of explainability throughout the entire machine learning life cycle. Explainability isn't just a post hoc feature but rather a lens for viewing and improving the entire machine learning process; there are already many promising directions that XAI can be used to augment AI systems, boosting accuracy, trust, transparency, robustness, and safety.

The ML Life Cycle

Developing any machine learning solution is an iterative process, and the steps you may take from start to finish will depend heavily on your use case and, ultimately, the business goals of the project. However, most end-to-end machine learning projects follow the same basic road map, shown in Figure 8-1, at a high-level aligning along three stages: discovery, development, and deployment.

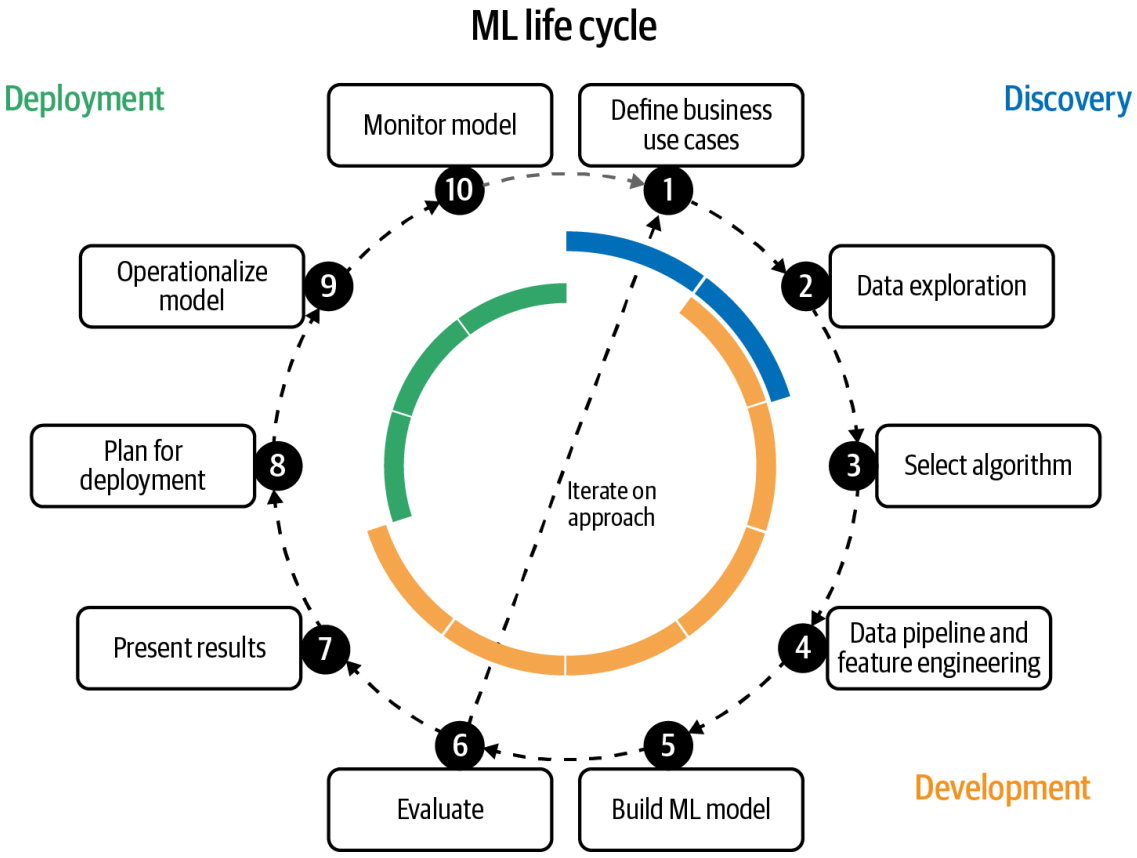

Figure 8-1. The true benefit of Explainable AI is that it can be applied across the entire machine learning life cycle from collecting, preprocessing, and improving datasets to model development, deployment, and monitoring.

Each stage is approached in turn, and there is a canonical order to the individual steps of each stage; however, there is often a healthy amount of iteration, and you may find yourself revisiting an earlier step as you gain

more information from a later step, particularly between discovery and development.

Explainability through discovery

Machine learning exists as a tool to solve a problem but is often only a small component of a much bigger solution. Discovery is the initial stage of any ML project, and the first step is to define the business use case and understand how exactly ML fits into the wider solution (Step 1 of Figure 8-1). Even at this early stage, it's important to consider the role that explainability will (or will not) play in the solution as well. This decision can have ripple effects on the later choices that arise in the ML cycle, and it is important to keep sight of these requirements through each stage of the life cycle.

One thing to consider is how the model predictions will be used in the final system or how they will be presented to the end user, and how explanations may (or may not) be presented (see Chapter 7 for detailed discussion about human factors related to XAI). For example, suppose our goal is to develop a predictive diagnostic model that informs a physician of potential health outcomes for a patient. It is possible that providing model explanations alongside predictions can improve patient outcomes. For example, a 2019 study[1] measured doctors' ability to diagnose diabetic retinopathy from scans by comparing three settings: unassisted (no machine learning model), model predictions only, and model predictions plus XAI heatmap. They found that doctors that were assisted with deep learning model predictions alone had improved accuracy over those who were unassisted. They also found that model predictions paired with XAI heatmaps increased accuracy for those patients with diabetic retinopathy and increased sensitivity without decreasing specificity. However, there could also be unforeseen negative side effects to consider as well; e.g., surfacing explanations could lead to unwanted outcomes as well such as doctors "overrelying" on explanations, and accepting inaccurate model predictions.

Or perhaps there are legal, ethical, or regulatory concerns that are required of the final ML solution that provide additional information as to why a model makes the recommendation it does. Any machine learning project should begin with a comprehensive understanding of the business opportunity and aligning on model constraints and performance require-

ments for deployment. Many times, there are indirect factors related to the business use case that influence development choices that arise later, like speed of inference or model size, or the cost to obtain data for different features in a dataset. It is equally important to discuss explainability or interpretability requirements and agree on what level and type of explanations are necessary or sufficient before embarking on a new project.

Step 2 of Figure 8-1 is data exploration and a deep dive into data understanding. This is also the beginning of the development portion of the ML life cycle. This is perhaps the most crucial step of designing an ML solution. Ultimately, the data guides the process, and it's necessary to understand the quality of the data that is available. From an explainability perspective, the majority of explainability techniques provide insights that are focused on the dataset and features. What are the distributions of the key features? How will missing values be handled? Are there outliers? Are any input values highly correlated? How can we augment the dataset? Is there bias in the dataset? It is important to keep in mind explainability during this stage as well. For example, as we saw in Chapter 3, when discussing explainability techniques for tabular data, some explainability techniques give misleading results when two features are highly correlated. It is important to be aware of these caveats early on.

Premodeling explainability is focused on understanding the data or any feature engineering that is used to train the machine learning model, see also the discussion in Chapter 2. This type of explainability is independent of any model and focused solely on the data. Explanations that focus solely on the dataset are often referred to as exploratory data analysis (EDA). EDA is a collection of statistical techniques and visualizations that are used to gain more insight into a dataset. EDA can take many forms, and visualization plays an important role during this step.

Commonly used tools such as Know Your Data and Facets allow you to quickly get a sense of the statistical properties of the features in your dataset such as the mean, standard deviation, range, and percentage of missing samples, as well as the feature dimensionality and presence of any outliers. Figure 8-2 shows how the Facets Overview looks when applied to the California Housing dataset. In Chapter 3, we apply many different explainability techniques to models built on this dataset. The Facets Overview gives a quick look at the feature distributions of values

in the dataset. This makes it easy to uncover common issues such as unexpected feature values, missing feature values, training/serving skew, and train/test/validation set skew. With a lens of explainability in mind, this knowledge of the data distribution and data quality is important for understanding model behavior, interpreting model predictions, and exposing any biases that might exist.

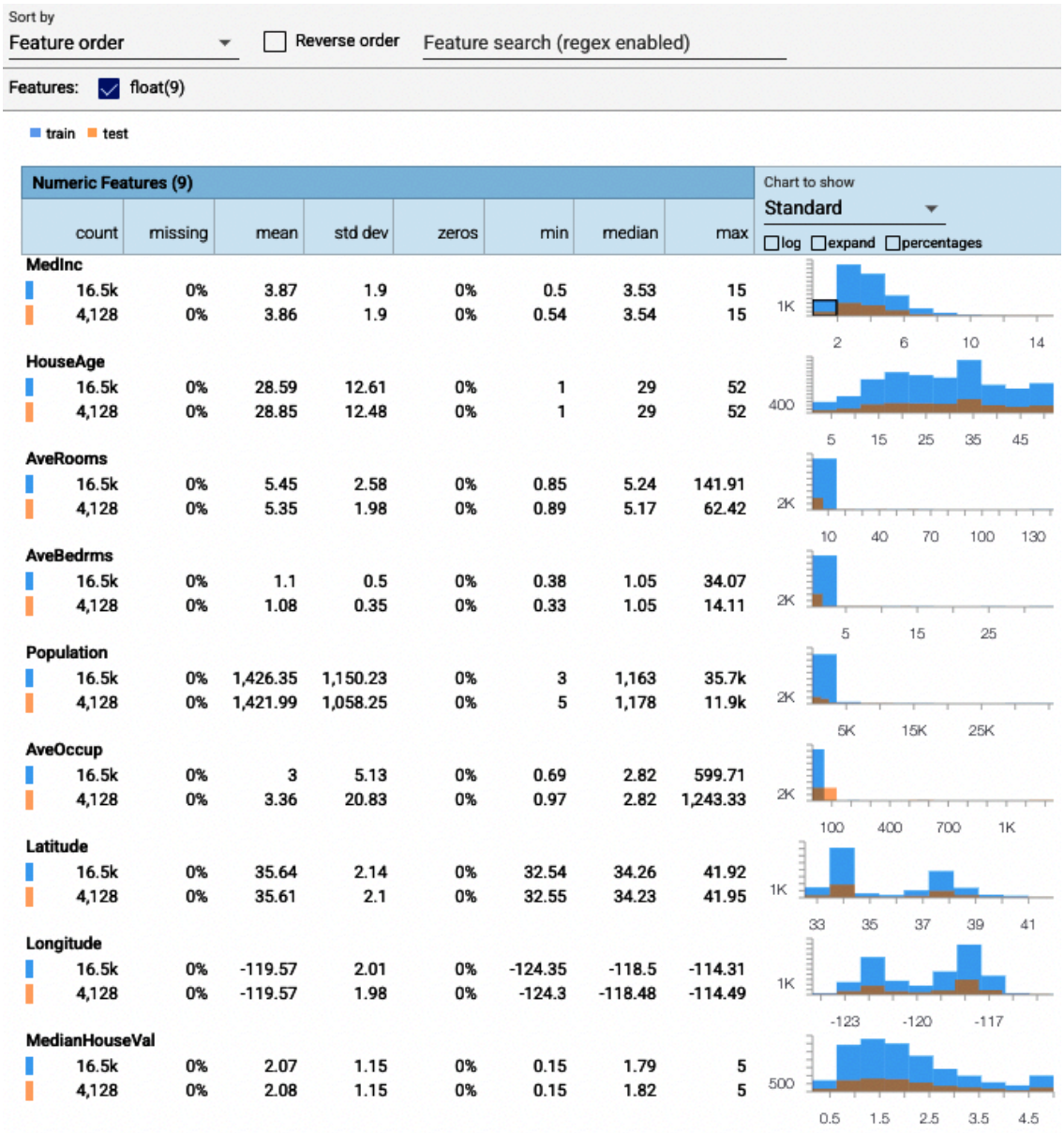

Figure 8-2. The Facets Overview of the California Housing dataset provides summary statistics for each feature and compares the training and test datasets. (Print readers can see the color image at https://oreil.ly/xai-fig-8-2.)

Explainability through development

Entering fully into the development stage, the next step in the ML life cycle (Step 3 in Figure 8-1) is model and algorithm selection. Explainability requirements play an important role in deciding which model to use, or even limit which models you may be able to use. One way to meet this explainability requirement is to employ naturally interpretable models, such as sparse linear models, decision trees, or additive models. These

models inherently provide a means of inspecting the model behavior by examining the model components directly; e.g., a single rule or path in the decision tree (see "Explaining Tree-Based Models"), or the weight of a specific feature in a linear model. If you're working with tabular data, these interpretable models might perform as well or better than more complex deep neural networks, and they have the added benefit of being fully explainable. If the model meets the performance requirements in terms of your predetermined evaluation metrics and uses a reasonable number of internal components (e.g., decision paths or features), such models can provide extremely useful insights. In fact, in most cases, if an interpretable model can be used and has similar performance to a more complex one, then the interpretable model is preferable.

However, for more complex data types, like images or text, you're likely to find neural networks more performant than these more inherently interpretable models. In this case, XAI paired with a complex neural network could be the way to go. Broadly speaking, explanations that are part of the model's prediction itself are known as *intrinsic* explanations. Intrinsic explanations rely on using an interpretable model or may require modifications to the model or training loop itself, as in constrained optimization (see the discussion on constrained optimization and deep lattice in Chapter 6). Intrinsic explanations are a good example of techniques that exist halfway in between pure explainability and interpretability, so don't worry too much if you see them described as either explainability or interpretability. Many of the techniques we've described in this book (see Chapter 3 to Chapter 5) fall into the category of post hoc explainability methods, which means they are applied after model training and rely on model predictions to surface explanations. Knowing how explainability techniques work for different models and their pros and cons can have a large impact on the outcome of the model selection stage of the ML cycle.

We begin by building data pipelines and engineering features (Step 4 of Figure 8-1). Feature engineering is the process of transforming raw input data into features that are more closely aligned with the model's learning objective and expressed in a format that can be fed to the model for training. Feature engineering techniques can involve bucketizing inputs, converting between data formats, tokenizing and stemming text, creating feature embeddings (as discussed in the beginning of Chapter 5), and many others.

Using explainability to perform feature engineering and feature selection is one of the most efficient ways to understand your model's performance and identify ways to improve. Applying an explainability approach during this step can help determine if your dataset includes excessive, or even confounding, features that are not contributing to the performance of your model. Almost all of the feature-based explainability methods that we've discussed throughout Chapters 3, 4, and 5 can be used in feature selection.

To do this correctly, you should apply your preferred XAI technique first on the entire training and test datasets to generate global explanations first, before looking at individual predictions or cohorts. One thing to keep in mind is that feature-based techniques that assign a negative value for a feature does not necessarily mean that it's not important or that you should remove that feature from the model. Instead, look for features that have little (or no) absolute influence on predictions.

After identifying these low-impact features, it's then a good idea to perform a sliced analysis. *Sliced analysis* is a general term for comparing certain quantitative metrics across various cross sections of your dataset. A cross section can be formed by taking all examples with a given feature or label value. In the context of explainability, you can also create a cross section based on cohort explanations. We introduced the concept of cohort explanations in Chapter 2. Cohort explanations are aggregated explanations that are generated for a specific subset of the full dataset. Comparing and contrasting these cohorts is one method of sliced analysis.

Sliced analysis can help you understand the behavior of your model in a way that is generalized enough to lead to broad improvements in model quality but localized enough to avoid the problem of trying to boil the entire ocean. You can often use the outputs of explanations in other sliced analysis tools, or even in follow-on statistical analyses.

DOCUMENT HOW YOU SELECTED YOUR SLICES

When performing sliced analysis, a common regret among practitioners is that they do not extensively document the selection process for the slice, and the parameters of the analysis. This lack of information can make it difficult to compare new slices to previous slices and analyses performed much earlier in the ML workflow, for example, performance after a model has been deployed and serving predictions for a year with the original analyses done during model development.

Sliced analysis is a useful tool for feature selection because it allows you to verify that the average influence of this feature does not appear to be zero, only due to high variance between positive and negative values for different cohorts of predictions. That is, you're able to confirm that the feature importance isn't artificially low merely due to the effects of other features.

In general, when performing this kind of feature selection analysis, there is no universal attribution threshold below which a feature is meaningless to the model's behavior. It requires experimentation and exploration. These explainability techniques simply provide another powerful lens with which to analyze and understand your features. You will find that the cutoff for meaningful features varies due to the complexity of the dataset, the cost of obtaining data for that feature, and if any features are highly correlated.

Once you have identified features to remove from your model, it is important to iteratively remove one feature at a time and then revalidate your model's performance along with regenerating any global or cohort explanations. We have routinely seen that trivial features that were initially ignored by the model become very influential as other, less relevant features are removed.

WHAT ABOUT LAYER SELECTION?

If we can use Explainable AI to select and prune features from our dataset, it seems reasonable to assume one could also use a model-focused technique to find layers that do not contribute to the performance of the model and remove, or simplify, these layers. There has been some research in this area (see Appendix), but unfortunately, as of 2022, we have not seen any techniques emerge that are both robust and simple enough to use to justify what is often a lengthy process of setting up the technique, interpreting the findings, and iteratively trying to change the model. Instead, many other techniques for model architecture selection are more useful at this time, such as model distillation, hyperparameter tuning, and neural architecture searches.

The next steps of the development stage, Steps 5 and 6 in Figure 8-1, are focused on building and evaluating the ML model. This is another area where the power of explainability techniques really shines. Throughout this book, we have discussed how explainability can identify what is influencing a model and how it behaves. Looked at from another perspective, Explainable AI can also give you insight into a model's robustness. By identifying where a model is relying on relevant features, you can understand how well the model will be able to adapt in unforeseen circumstances. A model that arrives at the correct prediction, but relies on seemingly unrelated information, may be easily swayed to make an incorrect prediction with only slight changes to the inputs. Likewise, if you find that the model is unable to identify similar instances from your dataset via example-based explanations (discussed in Chapter 6), it likely indicates the model is brittle.

Explainability can also show where the model is overly dependent on irrelevant features. These may be the background of an image, an ancillary column in structured data, or very specific grammar in text. Any of these could reveal that the model is also vulnerable to an adversarial attack, where a desired prediction can be engineered in advance without an obvious change to the inputs.

In general, when debugging ML models, explainability techniques are often one of the first tools practitioners reach for when they are trying to understand why a model performed poorly. It is important to recognize the value of investing in building a reusable, Explainable AI tool that allows many types of ML consumers in your organization to easily load a model, replay an interference, and receive an explanation. Investing in this tooling up front can help avoid having you, and other practitioners, having to be on call as a messenger who runs Jupyter notebook cells in order to shuttle explanations back to the interested party.

The last step of the development stage is to present results to the stakeholders and regulatory groups within the business (Step 7 of Figure 8-1). This is a critical and necessary step in the ML life cycle as it is often an important decision point for whether the ML model will make its way to production and deployment (the final stage of the ML life cycle) or not. Often this step is focused on creating numbers and visuals for initial reports that will be presented within the organization, and explainability tools provide an effective way to build confidence in the ML model predictions.

During this step, it's important to keep in mind your intended audience, and you should be cognizant of how the explanations are to be presented alongside the model results. In Chapter 7, we discussed in detail a number of aspects to keep in mind when interacting with explainability. Since the target audience during this step is the business stakeholders or end users, you'll most likely communicate to users with domain expertise and less ML experience.

Once again, example-based explanations can be used to complement a model's prediction by showing examples related to the one being predicted. There are different ways to determine "relatedness" of examples, but two types that we have found to be most useful during this step of the ML life cycle are normative examples and contrastive examples.

Normative and *contrastive examples* are examples that are closely related to the input example and whose label either agrees or differs with the model's prediction. For example, in Figure 8-3 the model predicts the label "cat" for the input image. Normative examples are examples from the training dataset that also have the label "cat," while contrastive examples

are examples from the training dataset that have a different ground truth label.

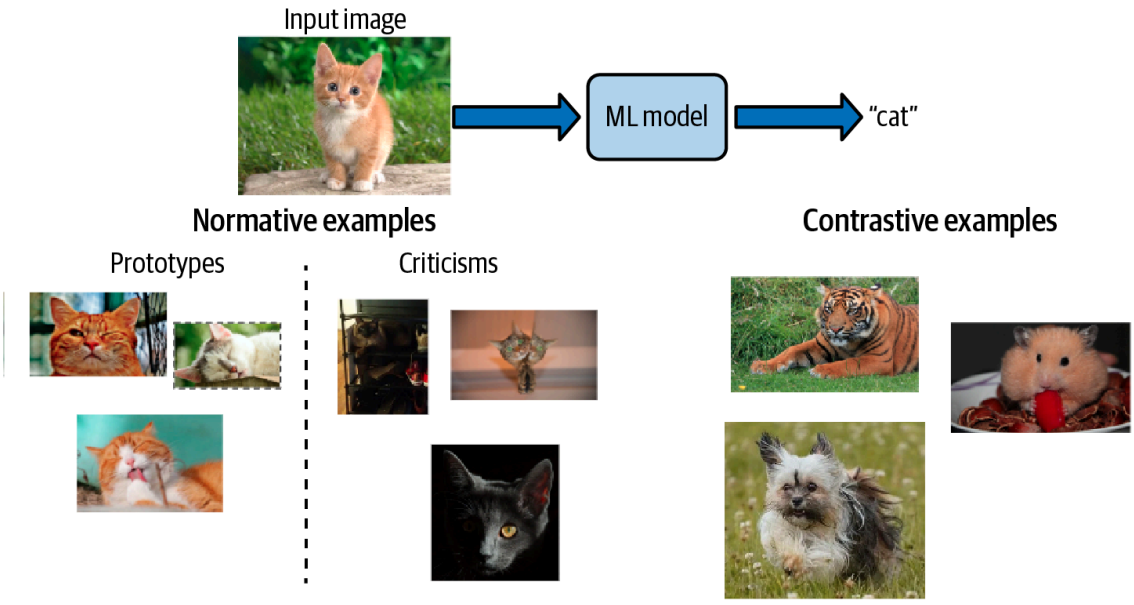

Figure 8-3. Normative examples are examples that come from the same predicted class (e.g., "cat") as the ML model, while contrastive examples are taken from a different class (e.g., "tiger," "hamster," or "dog").

One way to generate normative examples is by choosing a layer close to the model's final label layer and retrieving the *k* closest neighbors from the training set that have the same predicted label as the given instance. These examples are often called "prototypes." That is, these are examples in the dataset which are "close" to our given instance and share the same label. Naturally, we would expect to recognize common features between prototypes and the given instance.

To prevent reasoning errors due to overgeneralization, it is important to also present examples that differ from the norm but are still in the target class. That is, examples in the training set that share the same label as the model prediction but are unrelated to the example in question. These examples are called "criticisms." These kind of high-level, example-based explanations through prototypes and criticisms can be particularly useful to help build trust in a model's prediction.

When the model makes a prediction that is unexpected or goes against the intuition and experience of domain experts or stakeholders, example-based explanations are especially valuable. Consider, for example, a model used by a bank to assist in determining which loans to approve or deny. Suppose a business submits a new loan application and the ML model marks the application as high risk or high likelihood of defaulting. This could be based on a number of features such as the loan amount, the

previous credit history of the applicant, the FICA score, or the current debt liability of the applicant, etc.

Normative examples are instances from the training dataset that share the same label. This could be historical loan applications that were also deemed too risky by human experts or underwriters, or loan applications that had been approved but later defaulted on. A prototype would be a normative example from the training dataset that is similar in some way to the given application. Perhaps they had the same FICO score or they share similar credit history as the applicant. A criticism is a normative example that is quite different from the given application in question. That is, a criticism example would have a different FICO score or debt liability, but nonetheless was also deemed too risky or ultimately defaulted. Surfacing both prototypes and diverse criticisms gives a holistic view of the model's target class and helps stakeholders identify common or diverging trends between those examples and the instance in question.

In addition to prototypes and criticisms taken from the same predictive category, contrastive examples are taken from a different class to the model's prediction, but are nonetheless closely related to the given instance. These examples allow for reasoning about related instances from different categories. In the example of the loan application model we previously discussed, a contrastive example would be a historical loan application from the training dataset that is very similar or closely related to the new loan application but whose ground truth label was "no risk" or "low risk."

Contrastive examples are similar in spirit to counterfactuals (discussed in Chapter 2), derived from real instances instead of synthetic ones and near the model's decision boundary for the predicted class. In this way, they are a more informative type of counterfactual and give insight to the model's learned representations as well as the local nature of the training data distribution.

Figure 8-4 shows an example of both normative and contrastive examples. This image was taken from the game Quick, Draw! introduced by Google in 2016. The game is like an AI-assisted version of Pictionary. Quick, Draw! asks the user to draw a given object, like "pool" or "truck" or "vase," while an ML model tries to guess what the user is drawing. In Figure 8-4, the user was asked to draw a scorpion. Once the neural net-

work has had its turn guessing the image (without success, it seems), the game surfaces similar examples from the training dataset that were labeled as scorpion (on the right) as well as examples from the training dataset that were similar to the user's drawing but were not labeled as scorpion, such as a rake, a cherry, and a garden hose.

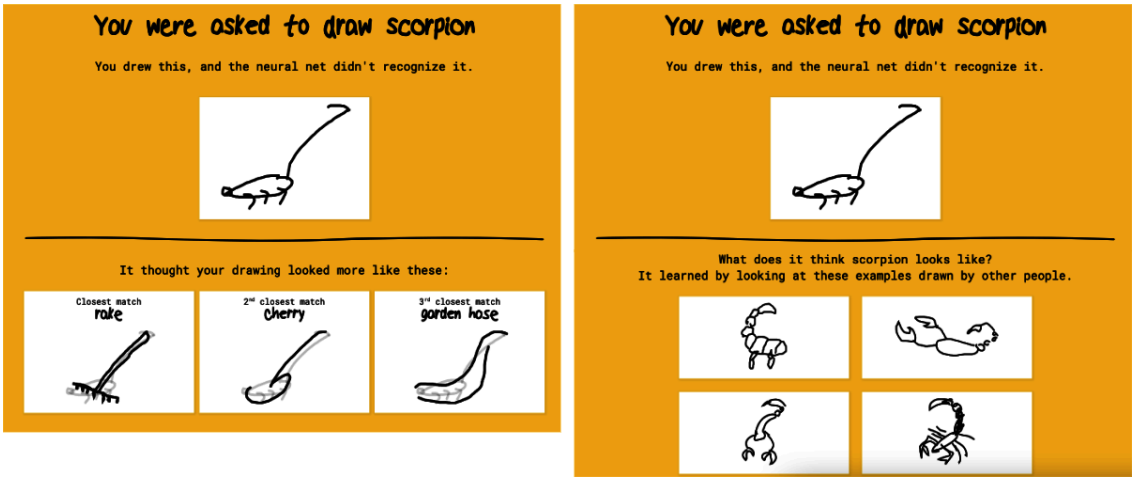

Figure 8-4. The game Quick, Draw! asks the user to draw an object while an ML model guesses the drawing. Once the user's turn is up, the game provides a comparison of normative and contrastive examples from the user's drawing.

Beyond single examples and representations, another approach to understanding model behavior is through concepts. Concepts enable reasoning about the representation of features that a model learns through training. In Chapter 6, we discussed in detail the TCAV (Testing with Concept Activation Vectors) method, which provides representable terms via sets of examples encoding the concepts of interest for a given example.

Explainability through deployment

The final stage of the ML life cycle is deployment (Steps 8–10 in Figure 8-1) and is commonly referred to as MLOps (ML operations). This stage is related to aspects of automating, monitoring, testing, managing, maintaining, and auditing machine learning models in production. It is a necessary component for any company hoping to scale the number of machine learning–driven applications within their organization.

When planning for deployment and through operationalizing the model (Steps 8 and 9 in Figure 8-1), it's important to consider what role explanations may play for the end user of the ML model. As with all things in ML, there is a trade-off between the benefit of providing explanations and the technical burden of maintaining that part of the infrastructure postproduction. Building an ML system that provides explanations alongside pre-

dictions in production will help end users of your system improve their understanding and build trust in the model. Likewise, automatically generating explanations as part of any AI governance process your organization has makes it much easier to demonstrate to stakeholders, such as business executives, regulators, and crossfunctional partners, how the model works and what influences the model. However, deploying a model that provides explanations requires additional technical investment and in some cases may be too much information for the end user who is not familiar with how to interpret those results. For ML systems that serve explanations alongside predictions to end users, the computational intensity of many techniques can prove to be a very difficult hurdle to overcome while also trying to quickly return a response in the range of tens to hundreds of milliseconds. In short, XAI can be costly, time-consuming, and potentially confusing to the end user.

We find there is often a trade-off between the XAI techniques that are most accurate and robust versus those that are computationally fast, but less valid. If you cannot find a good trade-off between the latency of your model serving explanations and the quality of the explanations, it may make sense to investigate using an inherently interpretable ML model, or to explore model extraction, which seeks to train a smaller, more interpretable model from the original model.

Deployment is not the end of a machine learning model's life cycle. Once the model goes into production, it can start to degrade, and its predictions can grow increasingly unreliable. How do you know that your model is working as expected in the real world? What if there are unexpected changes in the incoming data? Or the model no longer produces accurate or useful predictions? How will these changes be detected? These considerations are handled in the final stage, Step 10 of Figure 8-1, model monitoring.

Model monitoring and continued model evaluation provide a way to assess your model's performance over time. Traditional model monitoring focuses on detecting data skew or drift in the model's inputs or outputs using the same evaluation metrics you used during development. However, this approach treats the model itself as an opaque function, often focusing only on the inputs and outputs and when predictions start to deviate from the ground truth. This approach alone poses several challenges and lacks any actionable advice for addressing the root problem. A

much more rich and detailed analysis can be obtained by monitoring feature attribution drift through incorporating XAI during model monitoring as well.

For example, feature attributions can serve as an early indicator that the model's performance may be degrading, and monitoring feature attributions over time is a useful signal for detecting drift and skew in live inputs compared to what the model was trained on. Furthermore, feature attributions provide a number of advantages over more standard skew detection algorithms.

Most skew detection algorithms are specific to the input and are most commonly only able to handle numeric and/or categorical features. Other data modalities, like time-series, or unstructured inputs, like images, are more challenging and often have to be paired with some sort of dimensionality reduction. This added complexity is error prone and can be problematic both because of loss of information through dimension reduction techniques and sensitivity implementation choices. Feature attribution methods, however, are capable of handling multiple input modalities and, since attributions are dense numeric values, they can be compared more easily across data types.

Also, monitoring based on feature attributions requires less fine-tuning to detect data skew or drift. Not all data skews are created equal, and determining the right threshold can be tricky. Alerting too often risks desensitization while alerting too infrequently risks missing important, business-impacting issues. As a result, it's often necessary to tune skew detection algorithms, depending on the skew severity's impact on the final model. As the number of features increases, this can become an increasingly challenging task. XAI provides quantification of input influence that allows thresholds to be tuned in a more informed manner and thus balance the sensitivity and specificity trade-off for better alerts.

Another direction where XAI feature attributions can assist with skew detection is in handling joint distributions. Most skew detection algorithms are developed to detect skew in the univariate distribution of a given feature. However, this misses any data shift or skew that may have occurred in the joint distribution of related features. For example, a model that has two separate features "can fly" and "bird species" that has been put into production can all of a sudden start receiving examples of "flying pen-

guins." This would obviously constitute an outlier and be a sign of some kind of data drift or skew, but the issue could go unnoticed because individually the univariate statistics for the two features haven't changed. However, by using XAI techniques like feature attributions that take into account the joint distribution of features can catch these kinds of issues.

Lastly, and perhaps the most useful aspect of XAI techniques in model monitoring, is that they can provide actionable insights in addition to a diagnosis. Most skew detection algorithms surface outliers or alert data distributional shifts, but this can then require its own detailed analysis to determine how to address or mitigate those issues. XAI can help here as well as a means to augment the training dataset or provide an additional lens with which to better understand model features.

AI auditing often goes hand in hand with model monitoring. There are many use cases where it is advantageous to keep a detailed record of model predictions along with the corresponding model input requests and any other contextual information that may be useful. These details are useful for later diagnosis in the case of an audit.

Having model requests (i.e., instances sent to the model for prediction) and model responses (i.e., predictions) together are necessary for carrying out model monitoring, and you want to capture enough contextual information to re-create the environment exactly as is to assist in any subsequent analysis. While it may seem obvious, this can actually be quite difficult to accomplish effectively. Think, for example, of a self-driving car. This is an extremely complex system with multiple ML models in play as well as complex hardware and software requirements with onboard custom chips and real-time operating systems. In this situation, it may not be possible to re-create the environment precisely as is. However, recording some XAI outputs like attributions can be beneficial for a deeper understanding without having to re-create the exact same environment. XAI outputs can provide a high-level description of the state, which provides valuable information where a precise recreation is not possible.

More broadly, XAI plays an important role in meeting the requirements of AI regulations. Given the increased role that ML models play in our everyday lives, governments across the world are beginning to draft or introduce AI regulations that can be addressed with XAI.

AI Regulations and Explainability

Across the world, governments are beginning to draft or introduce regulations for AI. As of the summer of 2022, AI regulations have been introduced in the EU. The Digital Services Act (DSA) was approved by the European Parliament in July 2022 and will go into effect January 2024. The Artificial Intelligence Act has been proposed by the European Commission, introducing a common regulatory and legal framework applying to all types of AI. It is expected to come into effect in late 2023 or 2024. China also has several ministries drafting regulations. The first set of regulations, the New Generation Artificial Intelligence Ethics Specifications, from the Ministry of Science and Technology, lays out ethical norms for the use of AI in China covering areas such as the use and protection of personal information and responsible AI.

Similarly, in the US, the National Institute for Standards and Technology (NIST) has drafted recommendations for AI governance, the AI Risk Management Framework. The International Standards Organization (ISO) has begun finalizing ISO 42001 on Artificial Intelligence, and is actively working on a draft for Explainable AI (ISO 6254). Although individual regulations may be scoped to countries or sectors of technology, it is eventually expected the vast majority of AI employed by businesses will be subject to some form of accountability.

All of these regulations and standards have one common theme: they expect Explainable AI to have a central role in helping consumers and regulators assess how an ML behaves and whether it is adhering to regulations. So far, regulations and standards have avoided specifying exactly what type of explainability must be used, just that AI should have documentation demonstrating how the model behaves or, in real time or on request, the ML can explain its output.

This is both a blessing and a curse for ML practitioners. On the positive side, you will have the flexibility to determine which explainability techniques are the best match for your ML and use case. However, at the same time, you will likely be asked by stakeholders to demonstrate that whatever choice you made meets the regulatory standards for explainability, which are currently quite vague. In the coming years, we will likely see more clarity on the exact requirements for explainability, but

we expect it will prominently feature many of the classes of techniques we have covered in this book, such as feature attributions, example-based explanations, and counterfactuals.

What to Look Forward To in Explainable AI

What most excites us about the future of explainability is how it will move from individual, narrowly focused techniques to ones that generate richer explanations with less configuration needed in advance. Broadly, there are three trends to keep an eye on in the future of Explainable AI: natural and semantic explanations, interrogative explanations, and more targeted explanations.

Natural and Semantic Explanations

We often find that many types of explanations remain too technical or abstract for nontechnical users. An array of numbers representing feature attributions isn't necessarily very helpful for these users. Instead, techniques that can either present explanations in a more natural way, perhaps via a generative text model to create fluid sentences for the explanation, or are able to generate explanations based on a semantic understanding of the model and its dataset, will be much more helpful. Imagine if instead of being presented with an array of feature attribution values for a weather prediction, a user could be told, "There is an 80% chance of rain this afternoon because the temperature has dropped significantly and humidity is above 90%."

Semantic explanations, which require an even more innate understanding of the behaviors and concepts in the ML system, will also represent a large change in how we explain AIs. For example, an explainability technique may recognize that many of the similar examples where an ML classified a dog as a cat were due to poor lighting and low resolution in the photos. Instead of trying to highlight the pixels, where, for example, poor resolution or poor lighting may result in vague pixel attributions, it could categorically identify more pervasive causes for why the model failed.

Interrogative Explanations

Today, explanations are a one-way dialogue from the ML to the consumer. It is not unusual for someone to receive an explanation and immediately have more questions (see also the discussion in Chapter 7 on the human-interaction components of building explainable ML systems and "How to Effectively Present Explanations"). However, with current methods, that often requires an ML practitioner to roll up their sleeves and implement a new type of explanatory technique or perform additional types of explanations that were not expected before. A better ecosystem of Explainable AI tools will help make this job easier but will not solve the problem.

Instead, we expect a wave of second-generation explanation AI techniques that enable a richer experience where a user can query the ML for further information about a prediction or behavior and can even guide the user in improving their understanding. Imagine this as a conversation between the consumer, perhaps a regulator, and the AI about a credit-rating ML:

> *Regulator: Why was this individual given a credit rating of 520?*
>
> *AI: The most influential features were the high limits on the individual's credit cards, which caused the model to decrease its rating; their history of missed loan payments further drove the rating down.*
>
> *Regulator: Are individuals who live in similar zip codes (often an indirect variable for race in the US) with missed loan payments penalized as much as others?*
>
> *AI: No. Also, examining what the model considers to be 1,000 most similar people to this individual, there is no correlation with the zip code. Similar individuals who paid loans on time 25% more often on average had an increase of 50 points in their credit rating.*

Targeted Explanations

As of 2022, there has been little work performed on assessing whether explanations follow the rule of Occam's razor: that the simplest explanation

is the best. We expect that more robust explanations will be those that are more concise and targeted. For example, Local Explanations via Necessity and Sufficiency,[2], lays the foundation for these types of explanations by demonstrating how the minimal amount of perturbation necessary to flip a prediction provides an optimal explanation. These types of explanations will do much to address the brittleness problem described in Chapter 7, and will also take us a step further toward causal explanations.

Summary

In this chapter, we discussed how to design ML solutions with explainability in mind to build more reliable ML systems and provided a look toward the future of XAI. We've seen how Explainable AI techniques can be incorporated into each step of the ML life cycle, from discovery to development to deployment, assisting in building more robust ML solutions. We encourage you to think about XAI as a toolkit for better understanding machine learning models. We also provided a glimpse into what the future of XAI might hold and current research efforts.

Now, with these techniques and an understanding of how and where to apply them, you can improve both the models themselves and how your consumers work with them. Explainability is a rapidly changing field; we encourage you to view new techniques with optimism, but also give them some time to prove their worth in the rougher seas that constitute models and datasets in industry. It is also important to think responsibly about how you use explanations in high-risk settings, keeping in mind their potential to be brittle and create bias or false confidence in users. Most of all, explanations serve as a way for you to build a richer interaction with the models you work with every day.

1 Rory Sayres et al., "Using a Deep Learning Algorithm and Integrated Gradients Explanation to Assist Grading for Diabetic Retinopathy," *Ophthalmology* 126.4 (2019): 552–64.

2 David Watson et al., "Local Explanations via Necessity and Sufficiency: Unifying Theory and Practice," arXiv, 2021.