adoption of the governance process can only happen if it is communicated clearly and buy-in is sought from each stakeholder group.

Understand all of the steps in the existing process by interviewing those responsible. Where there is no existing formal process, this is often harder than it sounds because the process steps are often not explicitly defined, and ownership is unclear.

Attempting to map the policy-driven governance measures into the understanding of the process will identify issues in the process very quickly. Within one business there may be a range of different styles of project and governance needs, such as:

- One-off self-service analytics
- Internally consumed models
- Models embedded in public websites
- Models deployed to Internet of Things devices

In these cases, the differences between some processes may be so great it is best to think in terms of several parallel processes. Ultimately, every governance measure for each use case should be associated with a process step and with a team that is ultimately responsible, as presented here:

Process step	Example activities and governance considerations
Business scoping	Record objectives, define KPIs, and record sign-off: for internal governance considerations
Ideation	Data discovery: data quality and regulatory compliance constraints Algorithm choice: impacted by explainability requirements
Development	Data preparation: consider PII compliance, separation of legal regional scopes, avoid input bias Model development: consider model reproducibility and auditabilityModel testing and verification: bias and harm testing, explainability
Preproduction	Verify performance/bias with production data Production-ready testing: verify scalability
Deployment	Deployment strategy: driven by the level of operationalization Deployment verification testsUse of shadow challenger or A/B test techniques for in-production verification
Monitoring and feedback	Performance metrics and alerting Prediction log analysis for input drift with alerting

Step 6: Select the Tools for Centralized Governance Management

The MLOps governance process impacts both the complete ML life cycle and many teams across the organization. Each step requires a specific sequence of actions and checks to be executed. Traceability of both the development of the model and the execution of governance activities is a complex challenge.

Most organizations still have a "paper form" mindset for process management, where forms are filled in, circulated, signed, and filed. The forms may be text documents, circulated via email, and filed electronically, but the limitations of paper remain. It is hard to track progress, associate artifacts, review many projects at once, prompt for action, and remind teams of responsibilities. The complete record of events is typically spread across multiple systems and owned by individual teams, making a simple overview of analytics projects effectively impossible.

While teams will always have tools specific to their roles, MLOps governance is much more effective if the overarching process is managed and tracked from one system. This system should:

- Centralize the definition of the governance process flows for each class of analytics use cases
- Enable tracking and enforcement of the complete governance process
- Provide a single point of reference for the discovery of analytics projects
- Enable collaboration between teams, in particular, the transfer of work between teams
- Integrate with existing tools used for project execution

The current workflow, project management, and MLOps tools can only partially support these objectives. A new category of ML governance tools is emerging to support this need directly and more fully. These new tools focus on the specific challenges of ML governance, including:

- A single view of the status of all models (otherwise known as a model registry)
- Process gates with a sign-off mechanism to allow ready traceability of the history of decision making
- Ability to track all versions of a model
- Ability to link to artifact stores, metrics snapshots, and documentation
- Ability to tailor processes specifically for each class of analytics use cases
- Ability to integrate health monitoring from production systems and to track the performance of models against the original business KPIs

Step 7: Engage and Educate

Without a program of engagement and training for the groups involved in overseeing and executing the governance process, the chances of it being even partially adopted are slim. It is essential that the importance of MLOps governance to the business, and the necessity of each team's contribution, is communicated. Building on this understanding, every individual needs to learn what they must do, when, and how.

This exercise will require considerable documentation, training, and, most of all, time.

Start by communicating the broad vision for MLOps governance in the business. Highlight the dangers of the status quo, outline a process, and detail how it is tailored to the range of use cases.

Engage directly with each team involved and build a training program directly with them. Do not be afraid to leverage their experience to shape not only the training, but also the detailed implementation of their governance responsibilities. The result will be much stronger buy-in and more effective governance.

Step 8: Monitor and Refine

Is the newly implemented governance working? Are the prescribed steps being implemented, and are the objectives being met? What actions should be taken if things are going poorly? How do we measure the gap between today's reality and where the business needs to be?

Measuring success requires metrics and checks. It requires people to be tasked with monitoring and a way to address problems. The governance process and the way it is implemented will need to be refined over time, based both on lessons learned and on evolving requirements (including, as discussed earlier in this chapter, evolving regulatory requirements).

A big factor in the success of the process will be the diligence of the individuals responsible for the individual measures in the process, and incentivizing them is key.

Monitoring the governance process starts with a clear understanding of the key performance metrics and targets—KPIs for governance. These should aim to measure both whether the process is being enacted and whether objectives are being achieved. Monitoring and auditing can be time consuming, so look to automate metrics as far as possible and encourage individual teams to own the monitoring of metrics that relate to their area of responsibility.

It's hard to make people carry out tasks that seemingly deliver nothing concrete to those doing the work. One popular tactic to address this is gamification. This is not about making everything look like a video game, but about introducing incentives for people to carry out tasks where the main benefit is derived by others.

Look to gamify the governance process in simple ways: publishing KPI results widely is the simplest place to start. Just being able to see targets being met is a source of satisfaction and motivation. Leaderboards, whether at the team or individual level, can add some constructive element of competition. For example, people whose work consistently passes compliance checks the first time, or meets deadlines, should be able to feel their efforts are visible.

However, excessive competition can be disruptive and demotivating. A balance must be struck, and this is best achieved by building up gamification elements slowly over time. Start with the least competition-oriented and add new elements one by one, measuring their effectiveness before adding the next.

Monitoring changes in the governance landscape is essential. This might be regulatory, or it might be about public opinion. Those with responsibility for the strategic vision must continue to monitor it as well as have a process to evaluate potential changes.

Finally, all monitoring of the process is only worthwhile if issues are acted upon. Establish a process for agreeing on change and for enacting it. This may result in revisiting policies, processes, tools, responsibilities, education, and monitoring! It's necessary to iterate and refine, but the balance between efficiency and effectiveness is hard to find; many lessons can only be learned the hard way. Build a culture where people see iteration and refinement as a measure of a successful process, not a failed one.

Closing Thoughts

It's hard to separate MLOps from its governance. It's not possible to successfully manage the model life cycle, mitigate the risks, and deliver value at scale without governance. Governance impacts everything from how the business can acceptably exploit ML, to the data and algorithms that can be used, to the style of operationalization, monitoring, and retraining.

MLOps at scale is in its infancy. Few businesses are doing it, and even fewer are doing it well. While governance is the key to improving the effectiveness of MLOps, there are few tools today that directly address this challenge, and there is only piecemeal advice.

Public trust in ML is at risk. Even slow-moving organizations like the EU understand this. If trust is lost, then so too will be many of the benefits to be derived from ML. Additional legislation is being prepared, but even without this, businesses need to worry about the potential damage to their public image that can be caused by an inadvertently harmful model.

When planning to scale MLOps, start with governance and use it to drive the process. Don't bolt it on at the end. Think through the policies; think about using tooling to give a centralized view; engage across the organization. It will take time and iteration, but ultimately the business will be able to look back and be proud that it took its responsibilities seriously.

PART III

MLOps: Real-World Examples

CHAPTER 9

MLOps in Practice: Consumer Credit Risk Management

In the final chapters of this book, we explore three examples of how MLOps might look in practice. We explicitly chose these three examples because they represent fundamentally different use cases for machine learning and illustrate how MLOps methodology might differ to suit the needs of the business and its ML model life cycle practices.

Background: The Business Use Case

When a consumer asks for a loan, the credit institution has to make a decision on whether or not to grant it. Depending on the case, the amount of automation in the process may vary. However, it is very likely that the decision will be informed by scores that estimate the probability that the loan will or will not be repaid as expected.

Scores are routinely used at different stages of the process:

- At the prescreen stage, a score computed with a small number of features allows the institution to quickly discard some applications.
- At the underwriting stage, a score computed with all the required information gives a more precise basis for the decision.
- After the underwriting stage, scores can be used to assess the risk associated with loans in the portfolio.

Analytics methods have been used for decades to compute these probabilities. For example, the FICO score has been used since 1995 in the United States. Given the direct impact they have on the institutions' revenues and on customers' lives, these predictive models have always been under great scrutiny. Consequently, processes,

methods, and skills have been formalized into a highly regulated environment to ensure the sustainable performance of models.

Whether the models are based on expert-made rules, on classical statistical models, or on more recent machine learning algorithms, they all have to comply with similar regulations. Consumer credit risk management can therefore be seen as a precursor of MLOps: parallels with other use cases as well as best practices can be analyzed based on this use case.

At the time a credit decision is made, information about the customer's historical and current situation is usually available. How much credit does the customer hold? Has the customer ever not repaid a loan (in credit jargon, is the customer a delinquent)? In some countries, organizations called credit bureaus collect this information and make it available to creditors either directly or through the form of a score (like the aforementioned FICO score).

The definition of the target to be predicted is more complex. A customer not repaying as expected is a "bad" outcome in credit risk modeling. In theory, one should wait for the complete repayment to determine a "good" outcome and for the loss charge off to determine a "bad" outcome. However, it may take a long time to obtain these ultimate figures, and waiting for them would deter reactivity to changing conditions. As a result, trade-offs are usually made, based on various indicators, to declare "bad" outcomes before the losses are certain.

Model Development

Historically, credit risk modeling is based on a mix of rules ("manual feature engineering" in modern ML jargon) and logistic regression. Expert knowledge is vital to creating a good model. Building adapted customer segmentation as well as studying the influence of each variable and the interactions between variables requires enormous time and effort. Combined with advanced techniques like two-stage models with offset, advanced general linear models based on Tweedie distribution, or monotonicity constraints on one side and financial risk management techniques on the other side, this makes the field a playground for actuaries.

Gradient boosting algorithms like XGBoost have reduced the cost to build good models. However, their validation is made more complex by the black box effect: it's hard to get the feeling that such models give sensible results whatever the inputs. Nevertheless, credit risk modelers have learned to use and validate these new types of models. They have developed new validation methodologies based, for example, on individual explanations (e.g., Shapley values) to build trust into their models, which is a critical component of MLOps, as we've explored throughout this book.

Model Bias Considerations

The modeler also has to take into account selection biases, as the model will inevitably be used to reject applicants. As a result, the population to which a loan is granted is not representative of the applicant population.

By training a model version on the population selected by the previous model version without care, the data scientist would make a model unable to accurately predict on the rejected population because it is not represented in the training dataset, while it is exactly what is expected from the model. This effect is called cherry-picking. As a result, special methods, like reweighting based on the applicant population or calibrating the model based on external data, have to be used.

Models that are used for risk assessment and not only to make decisions about granting loans have to produce probabilities and not only yes/no outcomes. Usually, the probability produced directly by prediction models is not accurate. While it is not an issue if data scientists apply thresholding to obtain a binary classification, they will usually need a monotonous transformation called a *calibration* to recover "true" probabilities as evaluated on historical data.

The model validation for this use case typically consists of:

- Testing its performance on out-of-sample datasets, chosen after (or, in some cases, before, as well) the training datasets.
- Investigating the performance not only overall, but also per subpopulation. The subpopulations would typically have customer segments based on revenue, and with the rise of Responsible AI, other segmenting variables like gender or any protected attribute according to local regulation. Risks of not doing so can result in serious damages, as Apple learned the hard way in 2019 when its credit card was said to be "sexist" against women applying for credit.

Prepare for Production

Given the significant impact of credit risk models, their validation process involves significant work with regard to the modeling part of the life cycle, and it includes the full documentation of:

- The data used
- The model and the hypothesis made to build it
- The validation methodology and the validation results
- The monitoring methodology

The monitoring methodology in this scenario is twofold: data and performance drift. As the delay between the prediction and obtaining the ground truth is long (typically the duration of the loan plus a few months to take into account late payments), it is not enough to monitor the model performance: data drift also has to be monitored carefully.

For example, should an economic recession occur or should the commercial policy change, it is likely that the applicant population would change in such a way that the model's performance could not be guaranteed without further validation. Data drift is usually performed by customer segment with generic statistical metrics that measure distances between probability distributions (like Kolmogorov-Smirnov or Wasserstein distances) and also with metrics that are specific to financial services, like population stability index and characteristic stability index. Performance drift is also regularly assessed on subpopulations with generic metrics (AUC) or specific metrics (Kolmogorov-Smirnov, Gini).

The model documentation is usually reviewed by an MRM team in a very formal and standalone process. Such an independent review is a good practice to make sure that the right questions are asked of the model development team. In some critical cases, the validation team may rebuild the model from scratch given the documentation. In some cases, the second implementation is made using an alternative technology to establish confidence in documented understanding of the model and to highlight unseen bugs deriving from the original toolset.

Complex and time-consuming model validation processes have an implication on the entire MLOps life cycle. Quick-fixes and rapid model iteration are not possible with such lengthy QA and lead to a very slow and deliberate MLOps life cycle.

Deploy to Production

In a typical large financial services organization, the production environment is not only separate from the design environment, but also likely to be based on a different technical stack. The technical stack for critical operations—like transaction validation, but also potentially loan validation—will always evolve slowly.

Historically, the production environments have mainly supported rules and linear models like logistic regression. Some can handle more complex models such as PMML or JAR file. For less critical use cases, Docker deployment or deployment through integrated data science and machine learning platforms may be possible. As a result, the operationalization of the model may involve operations that range from clicking on a button to writing a formula based on a Microsoft Word document.

Activity logging of the deployed model is essential for monitoring model performance in such a high-value use case. Depending on the frequency of the monitoring, the feedback loop may be automated or not. For example, automation may not be

necessary if the task is performed only once or twice a year and the largest amount of time is spent asking questions of the data. On the other hand, automation might be essential if the assessment is done weekly, which may be the case for short-term loans with durations of a few months.

Closing Thoughts

Financial services have been developing schemes for prediction model validation and monitoring for decades. They have been able to continuously adapt to new modeling technologies like gradient boosting methods. Given their important impact, the processes around the life cycle management of these models are well formalized and even incorporated into many regulations. As a result, they can be a source of best practices for MLOps in other domains, though adaptations are needed as the trade-off between robustness on one side and cost efficiency, time to value, and—importantly—team frustration on the other may be different in other businesses.

CHAPTER 10

MLOps in Practice: Marketing Recommendation Engines

Makoto Miyazaki

Recommendation engines have become very popular in the last 20 years, from the first Amazon book recommendations to today's generalized use in digital shops, advertisements, and music and video streaming. We have all become accustomed to them. However, throughout the years, the underlying technologies behind these recommendation engines have evolved.

This chapter covers a use case that illustrates the adaption of and need for MLOps strategies given the particularities of a fast-paced and rapidly changing machine learning model life cycle.

The Rise of Recommendation Engines

Historically, marketing recommendations were human-built. Based on qualitative and quantitative marketing studies, marketing moguls would set up rules that statically defined the impression (in the sense of advertising views) sent to a customer with given characteristics. This technique gave rise to the marketing data mining urban legend that a grocery chain discovered that men who bought diapers on Thursdays and Saturdays were more likely to buy beer as well and hence placing the two next to each other will increase beer sales.

Overall, recommendation engines created manually presented numerous bottlenecks that resulted in a significant amount of wasted money: it was hard to build rules based on many different customer features because the rule creation process was manual, it was hard to set up experiments to test many different kinds of impressions, and it was hard to update the rules when the behavior of the customers changed.

The Role of Machine Learning

The rise of ML has brought a new paradigm to recommendation engines, allowing for the elimination of rules based on human insight. A new class of algorithm called *collaborative filtering* dominates the field. This algorithm is able to analyze customer and purchase data with millions of customers and tens of thousands of products to perform recommendations without any prior marketing knowledge. By identifying efficiently what customers that look like the current customer bought, marketers can rely on automatic strategies that outperform traditional ones both in cost and efficiency.

Because the process of building strategies is automatic, it is possible to update them regularly and to compare them using A/B testing or shadow scoring schemes (including the way to choose the impression among all possibilities). Note that these algorithms may be combined with more classical business rules for various reasons—e.g., avoiding the filtering bubble, not selling a product in a given geographical area, or preventing the use of a specific association that is statistically meaningful but unethical to use (like proposing alcohol to a recovering alcoholic), to name a few.

Push or Pull?

When implementing a recommendation engine, it is important to keep in mind that its structure will depend on whether the recommendations are pushed or pulled. Push channels are the easiest to handle; for example, they can consist of sending emails or making outbound calls.

The recommendation engine can be run on a regular basis in batch mode (typically between once a day and once a month), and it is easy to split the customer dataset into several parts to perform analysis within a sound experimental design. The regularity of the process allows for regular review and optimization of the strategy.

Pull channels are often more effective because they provide information to customers when they need it—for example, when doing an online search or when calling a customer service line. Specific information from the session can be used (e.g., what the user has searched for) to precisely tailor the recommendation. Music streaming platforms, for instance, provide pull-channel recommendations for playlists.

Recommendations can be prebaked, as illustrated in the in-depth example in this chapter, or made in real time. In the latter case, a special architecture has to be set up to compute recommendations on the fly.

Comparing strategies in a pull context is more challenging. First, the customers who call in on a given channel are likely to differ from the average customer. In simple cases, it is possible to randomly choose the strategy to use for each recommendation, but it also happens that the strategy needs to be used consistently over a given period for a given customer. This usually results in an unbalanced proportion of

recommendations made with each strategy, which makes the statistical treatment to decide which one is the best more complex. However, once a good framework is set, this allows a very quick improvement cycle, as many strategies can be tested in real time.

Data Preparation

The customer data that is usually accessible to a recommendation engine is composed of the following:

- Structural information about the customer (e.g., age, gender, location)
- Information about historical activities (e.g., past views, purchases, searches)
- Current context (e.g., current search, viewed product)

Whatever the technique used, all customer information has to be aggregated into a vector (a list of fixed size) of characteristics. For example, from the historical activities, the following characteristics could be extracted:

- Amount of purchases during the last week or the last month
- Number of views during past periods
- Increase in spending or in views during the last month
- Previously seen impressions and customer's response

In addition to customer data, the recommendation context can also be taken into account. For example, days to summer for seasonal products like above-ground swimming pools or days to monthly pay day, as some customers delay purchases for cash flow reasons.

Once the customer and context data is formatted, it is important to define the set of possible recommendations, and there are many choices to make. The same product may be presented with different offers, which may be communicated in different ways.

It is of the utmost importance not to forget the "do not recommend anything" option. Indeed, most of us have the personal experience that not all recommendations have a positive impact. Sometimes not showing anything might be better than the alternatives. It's also important to consider that some customers may not be entitled to see certain recommendations, for example depending on their geographical origin.

Design and Manage Experiments

To leverage the continuous improvement potential of recommendation engines, it is necessary to experiment with different strategies within a sound framework. When designing a prediction model for a recommendation engine, the data scientist might well focus on a simple strategy, such as predicting the probability that a given customer clicks on a given recommendation.

This may seem a reasonable compromise compared to the more precise approach of trying to gather information about whether the customer purchased the product and whether to attribute this purchase to a given recommendation. However, this is not adequate from a business perspective, as phenomena like cannibalization may occur (i.e., by showing a low-margin product to a customer, one might prevent them from buying a high-margin product). As a result, even if the predictions were good and resulted in increased sales volume, the overall revenue might be reduced.

On the other hand, bluntly promoting the organization's interest and not the customer's could also have detrimental long-term consequences. The overarching KPI that is used to assess if a given strategy yields better results should be carefully chosen, together with the time period over which it is evaluated. Choosing the revenue over a two-week period after the recommendation as the main KPI is common practice.

To be as close as possible to an experimental setting, also called A/B testing, the control group and the experimental groups have to be carefully chosen. Ideally, the groups are defined before the start of the experiment by randomly splitting the customer base. If possible, customers should not have been involved in another experimentation recently so that their historical data is not polluted by its impact. However, this may not be possible in a pull setting in which many new customers are coming in. In this case, the assignment could be decided on the fly. The size of the groups as well as the duration of the experiments depend on the expected magnitude of the KPI difference between the two groups: the larger the expected effect, the smaller the group size and the shorter the duration.

The experimentation could also be done in two steps: in the first one, two groups of equal but limited size could be selected. If the experimentation is positive, the deployment could start with 10% on the new strategy, a proportion that can be raised progressively if results are in line with expectations.

Model Training and Deployment

To better illustrate the MLOps process for this type of use case, the following sections focus on the specific example of a hypothetical company deploying an automated pipeline to train and deploy recommendation engines. The company is a global software company (let's call it MarketCloud) headquartered in Silicon Valley.

One of MarketCloud's products is a software-as-a-service (SaaS) platform called SalesCore. SalesCore is a B2B product that allows its users (businesses) to drive sales to customers in a simple manner by keeping track of deals, clearing tedious administration tasks off their desks, and creating customized product offers for each customer (see Figure 10-1).

From time to time, MarketCloud's clients use the cloud-based SalesCore while on a call with their customers, adjusting their sales strategy by looking at past interactions with the customers as well as at the product offers and discounts suggested by SalesCore.

MarketCloud is a mid-sized company with an annual revenue of around $200 million and a few thousand employees. From salespeople at a brewing company to those in a telecommunication entity, MarketCloud's clients represent a range of businesses.

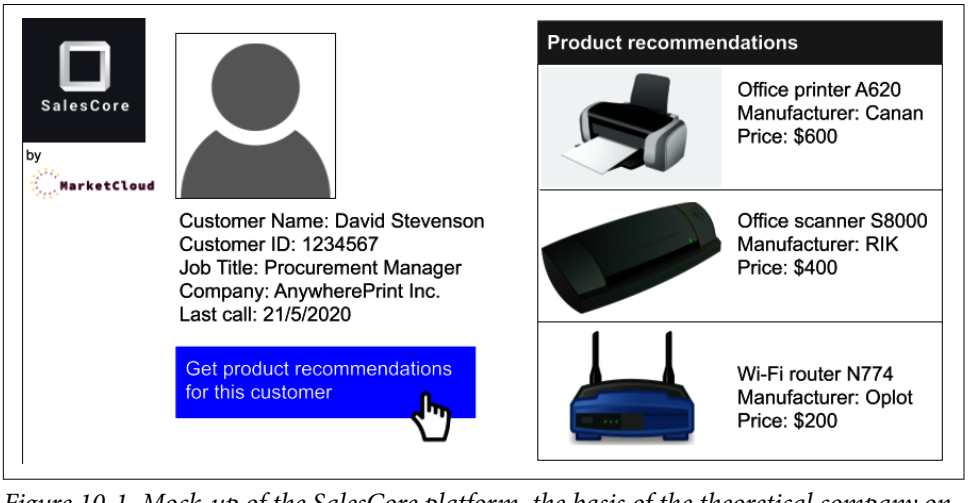

Figure 10-1. Mock-up of the SalesCore platform, the basis of the theoretical company on which this section's example is based

MarketCloud would like to automatically display product suggestions on SalesCore to the salespeople trying to sell products to the customers. Suggestions would be made based on customers' information and their past interaction records with the salesperson; suggestions would therefore be customized for each customer. In other words, SalesCore is based on a recommendation engine used in a pull (inbound calls) or push (outbound calls) context. Salespeople would be able to incorporate in their sales strategy the suggested products while on a call with their customers.

To implement this idea, MarketCloud needs to build a recommendation engine and embed it into SalesCore's platform, which, from a model training and deployment standpoint, presents several challenges. We'll present these challenges in this section,

and in the next section we'll show MLOps strategies that allow the company to handle each of them.

Scalability and Customizability

MarketCloud's business model (selling software for client companies to help them sell their own products) presents an interesting situation. Each client company has its own dataset, mainly about its products and customers, and it doesn't wish to share the data with other companies.

If MarketCloud has around four thousand clients using SalesCore, that means instead of having a universal recommender system for all the clients, it would need to create four thousand different systems. MarketCloud needs to come up with a way to build four thousand recommendation systems as efficiently as possible since there is no way it can handcraft that many systems one by one.

Monitoring and Retraining Strategy

Each of the four thousand recommendation engines would be trained on the customer data of the corresponding client. Therefore, each of them would be a different model, yielding a different performance result and making it nearly impossible for the company to manually keep an eye on all four thousand. For example, the recommendation engine for client A in the beverage industry might consistently give good product suggestions, while the engine for client B in the telecommunication sector might seldom provide good suggestions. MarketCloud needed to come up with a way to automate the monitoring and the subsequent model retraining strategy in case the performance degraded.

Real-Time Scoring

In many situations, MarketCloud's clients use SalesCore when they are talking to their customers on the phone. The sales negotiation evolves every single minute during the call, and the salesperson needs to adjust the strategy during the interaction with the customer, so the recommendation engine has to be responsive to real-time requests.

For example, imagine you as a salesperson are on a call with your customer to sell telecommunication devices. The customer tells you what his office looks like, the existing infrastructure at the office such as an optic fiber, the type of WiFi network, and so forth. Upon entering this information in SalesCore, you want the platform to give you a suggestion for the products that your customer could feasibly purchase. This response from the platform needs to be in real time, not 10 minutes later, after the call, or on the following day.

Ability to Turn Recommendations On and Off

Responsible AI principles acknowledge that retaining human involvement is important. This can be done through a human-in-command design,[1] by which it should be possible *not* to use the AI. In addition, adoption is likely to be low if users cannot override AI recommendations. Some clients value using their own intuition about which products to recommend to their customers. For this reason, MarketCloud wants to give its clients full control to turn the recommendation engine on and off so that the clients can use the recommendations when they want.

Pipeline Structure and Deployment Strategy

To efficiently build four thousand recommendation engines, MarketCloud decided to make one data pipeline as a prototype and duplicate it four thousand times. This prototype pipeline consists of the necessary data preprocessing steps and a single recommendation engine, built on an example dataset. The algorithms used in the recommendation engines will be the same across all four thousand pipelines, but they will be trained with the specific data associated with each client (see Figure 10-2).

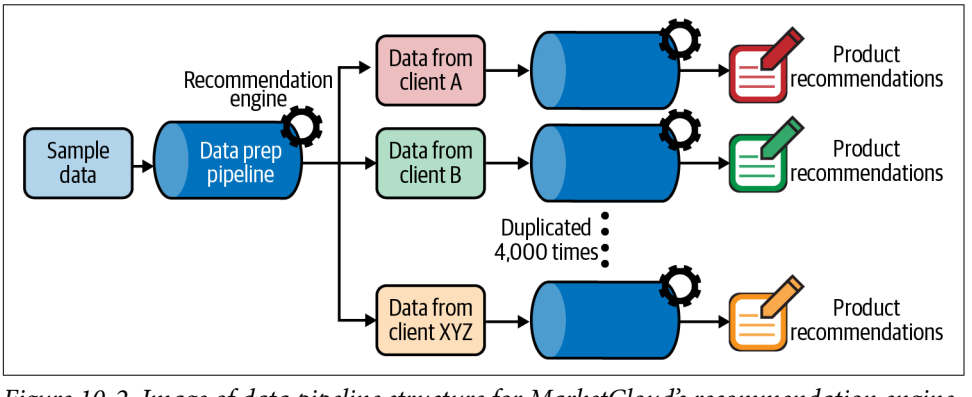

Figure 10-2. Image of data pipeline structure for MarketCloud's recommendation engine project

In this way, MarketCloud can efficiently launch four thousand recommendation systems. The users will still retain some room for customization, because the engine is trained with their own data, and each algorithm will work with different parameters —i.e., it's adopted to the customer and product information of each client.

What makes it possible to scale up a single pipeline to four thousand pipelines is the universal schema of the dataset. If a dataset from client A has 100 columns whereas

1 For an explanation of human-in-command design, see Karen Yeung, "Responsibility and AI" (Council of Europe study, DGI(2019)05), 64, footnote 229.