client B has 50, or if the column "number of purchased items" from client A is an integer whereas the same column from client B is a string, they would need to go through different preprocessing pipelines.

Although each client has different customer and product data, at the point that this data is registered on SalesCore, it acquires the same number of columns and the same data types for each column. This makes things easier, as MarketCloud simply needs to copy a single pipeline four thousand times.

Each recommendation system embedded in the four thousand pipelines will have different API endpoints. On the surface, it looks like when a user clicks the "show product recommendations" button, SalesCore displays a list of suggested products. But in the background, what is happening is that by clicking the button, the user is hitting the specific API endpoint associated with the ranked product lists for the specific customer.

Monitoring and Feedback

Maintaining four thousand recommendation systems is not an easy task, and while there have already been many MLOps considerations until this point, this is maybe the most complex part. Each system's performance needs to be monitored and updated as needed. To implement this monitoring strategy at a large scale, MarketCloud can automate the scenario for retraining and updating the models.

Retraining Models

Clients obtain new customers, some of the customers churn, every once in a while new products are added to or dropped from their catalogs; the bottom line is that customer and product data are constantly changing, and recommendation systems have to reflect the latest data. It's the only way they can maintain the quality of the recommendation, and, more importantly, avoid a situation such as recommending a WiFi router that is outdated and no longer supported.

To reflect the latest data, the team could program a scenario to automatically update the database with the newest customer and product data, retraining the model with the latest datasets every day at midnight. This automation scenario could then be implemented in all four thousand data pipelines.

The retraining frequency can differ depending on the use case. Thanks to the high degree of automation, retraining every night in this case is possible. In other contexts, retraining could be triggered by various signals (e.g., signification volume of new information or drift in customer behavior, be it aperiodic or seasonal).

In addition, the delay between the recommendation and the point in time at which its effect is evaluated has to be taken into account. If the impact is only known with a

delay of several months, it is unlikely that retraining every day is adequate. Indeed, if the behavior changes so fast that retraining it every day is needed, it is likely that the model is completely outdated when it is used to make recommendations several months after the most recent ones in the training data.

Updating Models

Updating models is also one of the key features of automation strategies at scale. In this case, for each of the four thousand pipelines, retrained models must be compared to the existing models. Their performances can be compared using metrics such as RMSE (root-mean-square error), and only when the performance of the retrained model beats the prior one does the retrained model get deployed to SalesCore.

Runs Overnight, Sleeps During Daytime

Although the model is retrained every day, users do not interact directly with the model. Using the updated model, the platform actually finishes calculating the ranked list of products for all the customers during the night. On the following day, when a user hits the "show product recommendations" button, the platform simply looks at the customer ID and returns the ranked list of products for the specific customer.

To the user, it looks as if the recommendation engine is running in real time. In reality, however, everything is already prepared overnight, and the engine is sleeping during daytime. This makes it possible to get the recommendation instantly without any downtime.

Option to Manually Control Models

Although the monitoring, retraining, and updating of the models is fully automated, MarketCloud still leaves room for its clients to turn the models on and off. More precisely, MarketCloud allows the users to choose from three options to interact with the models:

- Turn on to get the recommendation based on the most updated dataset
- Freeze to stop retraining with the new data, but keep using the same model
- Turn off to completely stop using the recommendation functionality of SalesCore

Machine learning algorithms attempt to convert practical knowledge into meaningful algorithms to automate processing tasks. However, it is still good practice to leave room for users to rely on their domain knowledge, as they are presumed to be far more capable of identifying, articulating, and demonstrating day-to-day process problems in business.

The second option is important because it allows users to stay in the current quality of the recommendation without having the recommendation engines updated with the newer data. Whether the current model is replaced with a retrained one depends on the mathematical evaluation based on metrics such as the RMSE. However, if users feel that the product recommendations on SalesCore are already working well for pushing sales, they have the choice not to risk changing the recommendation quality.

Option to Automatically Control Models

For those that don't want to manually handle the models, the platform could also propose A/B testing so that the impact of new versions is tested before fully switching to them. Multi-armed bandit algorithms (an algorithm that allows for maximization of the revenue of a user facing multiple slot machines, each with a different probability to win and a different proportion of the money given back on average) are used for this purpose.

Let's assume that several model versions are available. The goal is to use the most efficient one, but to do that, the algorithm obviously has to first learn which is the most efficient. Therefore, it balances these two objectives: sometimes, it tries algorithms that may not be the most efficient to learn if they are efficient (exploration), and sometimes it uses the version that is likely to be the most efficient to maximize the revenue (exploitation). In addition, it forgets past information, as the algorithm knows the most efficient today may not be the most efficient tomorrow.

The most advanced option consists in training different models for different KPIs (click, buy, expected revenue, etc.). A method inspired from ensemble models would then allow for the solving of conflicts between models.

Monitoring Performance

When a salesperson suggests a customer buy the products recommended by SalesCore, the interaction of the customer with the recommended products as well as whether the customer bought them or not is recorded. This record can then be used to keep track of the performance of the recommender system, overwriting the customer and product dataset with this record to feed the most updated information to the model when it is retrained.

Thanks to this ground truth recording process, dashboards showing model performance can be presented to the user, including performance comparison from A/B testing. Because the ground truth is obtained quickly, data drift monitoring is secondary. A version of the model is trained every night, but, thanks to the freeze mechanism, the user can choose the active version based on the quantitative information. It is customary to keep the human in the loop on these high-impact decisions where the performance metrics have a hard time capturing the full context around the decision.

In the case of A/B testing, it is important that only one experiment be done at a time on a group of customers; the impact of combined strategies cannot be simply added. With such considerations in mind, it is possible to build a sound baseline to perform a counterfactual analysis and derive the increased revenue and/or the decreased churn linked to a new strategy.

Apart from this, MarketCloud can also monitor the algorithm performance at a macro level, by checking how many clients froze or turned off the recommender systems. If many clients turned off the recommender systems, that's a strong indicator that they are not satisfied with the recommendation quality.

Closing Thoughts

This use case is peculiar in the sense that MarketCloud built a sales platform that many other companies use to sell products, where the ownership of the data belongs to each company, and the data cannot be shared across companies. This brings a challenging situation where MarketCloud must create different recommender systems for each of the users instead of pooling all the data to create a universal recommendation engine.

MarketCloud can overcome this obstacle by creating a single pipeline into which data from many different companies can be fed. By having the data go through an automated recommendation engine training scenario, MarketCloud created many recommendation engines trained on different datasets. Good MLOps processes are what allow the company to do this at scale.

It's worth noting that though this use case is fictionalized, it is based on reality. The real-life team that tackled a similar project took around three months to finish. The team used a data science and machine learning platform to orchestrate the duplication of a single pipeline to four thousand copies and to automate the processes to feed corresponding datasets to each pipeline and train the models. Of necessity, they accepted trade-offs between the recommendation quality and scalability to efficiently launch the product. If the team had carefully crafted a custom recommendation engine for each of the four thousand pipelines by, for example, choosing the best algorithm for each client, the recommendation engines would have been of a higher quality, but they would have never been able to complete the project with such a small team in such a short period of time.

CHAPTER 11

MLOps in Practice: Consumption Forecast

Nicolas Omont

Predictions at various times and geographical scales play an important role in the operation of a power grid. They allow for simulation of possible future states of the system and for making sure that it can safely operate. This chapter will walk through a machine learning model life cycle and MLOps use case for consumption forecasting, including business considerations, data collection, and implementation decisions. Though this particular chapter is focused on power grids, the considerations and particularities of the use case can be generalized to other industrial cases that use consumption forecasting.

Power Systems

Bulk power systems are the backbone of power grids. Also called transmission networks, they form the core of the system that keeps the lights on. These systems are mainly composed of lines and transformers, which are indirectly connected with most producers and consumers through distribution networks that take care of the last few kilometers of transmission. As illustrated in Figure 11-1, only the largest producers and consumers are directly connected to the bulk system.

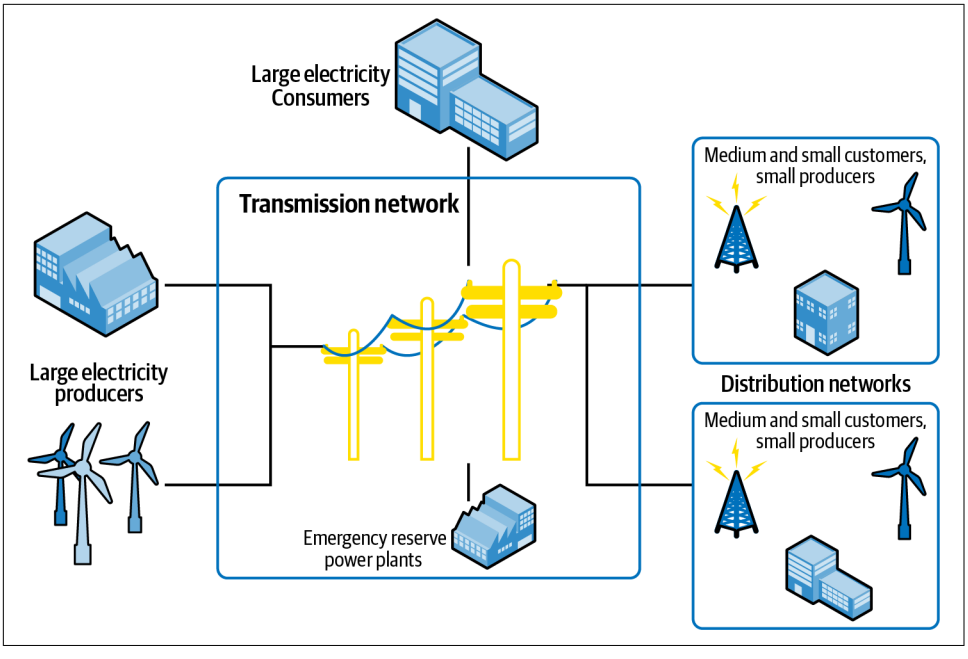

Figure 11-1. A sample bulk power system, to which only the largest producers and consumers are directly connected

The longer the transmission distance and the larger the energy volume to be transmitted, the higher the voltage used: on the lower end, a few tens of kilovolts for a few tens of megawatts over a few tens of kilometers; on the upper end, one million volts for a few thousand megawatts over a few thousand kilometers. (A line with a capacity of one megawatt can be used to provide power to around one thousand inhabitants in Europe.) The operation of transmission systems has always required a lot of communications and computations because of its properties:

No energy storage
: The network stores a meaningless amount of energy—less than one second of consumption in the grid and up to 30 seconds in the alternators and motors. By way of contrast, a gas network stores several hours of consumption in its pipeline. Therefore, actions have to be taken very quickly to balance production and consumption and avoid blackouts.

Weak flow control
: On telecommunication networks, congestions are handled by dropping packets or by not establishing a call. There is no equivalent mechanism in power grids, which means the power flow on a grid element can be higher than its operating limit. Actions have to be taken after a few seconds to a few hours of overload depending on the technology and the severity. Flow control technologies do exist,

but there is a trade-off between flow control and instantaneous balance: the power has to find a path from generation to consumption.

Because of these two properties, the grid operator always has to anticipate the contingencies: if this grid element fails, will the overload on the remaining elements remain acceptable? The anticipation is done on several timescales, from the next five minutes to the next five decades. The actions to be taken depend on the horizon. For example:

- Below five minutes: no human action is possible. Automatic actions should already be well defined.
- Between five minutes and a few hours ahead: production schedule and grid topology adaptation (opening of breakers and other flow control technologies).
- A few days ahead: maintenance schedule adaptations.
- A few seasons ahead: maintenance schedule adaptations, contracts with producers or consumers to guarantee power capacity or limit power generation or consumption.
- From 5 to 50 years ahead: investment in grid elements. Lines and transformers have standard life expectancies of several decades; practically, it is expected that some grid elements will last over one hundred years.

Another concern is anticipating at different geographical scales. While some contingencies only have effects on a small part of the grid, some may have a continental impact and may require coordinated actions among several countries to mitigate their effects. As a result, operating the grid requires:

1. Collecting data over a wide geographical area with strong time constraints.
2. Processing data to anticipate and act accordingly.

Data Collection

Collecting past data is the first step to making forecasts. There are two largely independent sources of data: the SCADA (supervisory control and data acquisition) system and the metering system. Depending on the prediction use case, one or the other may be used.

The SCADA system collects data in real time to provide an up-to-date view of the system to the operator. It also allows commands to be sent to network equipment—for example to open and close a breaker. The most impressive representation of the system is the synoptic screen found in most control rooms as shown in Figure 11-2.

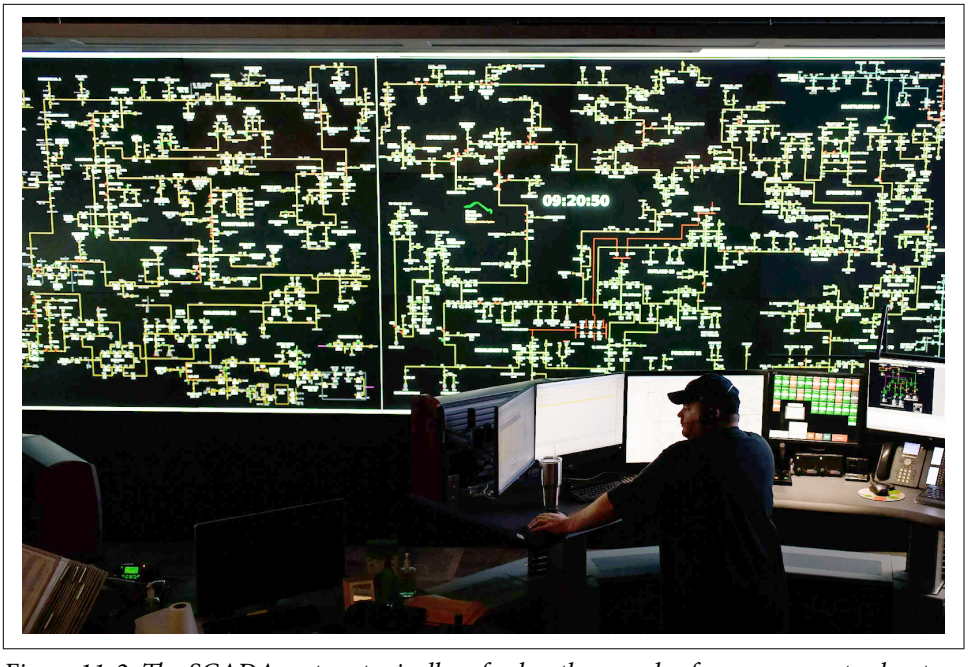

Figure 11-2. The SCADA system typically refreshes thousands of measurements about flows, consumption, and generation on the grid every 10 seconds or less

Some measurements are intentionally redundant, such as measuring power loss. If the power flow is measured at each end of a line, then the difference between them is equal to the losses on the line. These losses can be physically estimated so that it is possible to handle the case when one measure is missing, to detect anomalies, or to improve the precision of the estimates.

The process that uses this redundancy to produce a state of the network is called the state estimation, and it is run every few minutes. When an operating limit is not satisfied, the SCADA system raises an alarm. However, the SCADA cannot raise an alarm when an operating limit would not be satisfied if one of the grid elements went out of order.

Simulations of network element loss (N-1 simulation) on the consistent state produced by the state estimation are run on a regular basis, and the value of SCADA data fades quickly; therefore, when it is historized, it is not consolidated; missing values are usually not input, and anomalies are usually not corrected. State estimations are used by a variety of processes so that they are usually historized over a few months to a few years.

The metering system that is used for invoicing does not need to be as reactive as the SCADA system, but should be precise. It focuses on generation and consumption, not

flow. Rather than monitoring instantaneous power, it records the withdrawn or injected energy over a period of time that ranges between a few minutes and one hour.

The information it gathers was previously made available after a delay of a day or more. Newer systems make it available within a few minutes. However, consolidation and validation are usually needed when there are missing measurements or anomalies so that the final data is still usually available within a few working days. This data is well historized.

Problem Definition: Machine Learning, or Not Machine Learning?

Not all use cases are appropriate for machine learning. Some can be solved more easily and cheaply in other ways. The techniques used to make forecasts for the type of use case presented here are different in these three situations as shown in Table 11-1.

Table 11-1. Forecasting techniques by use case

Use case	Forecasting technique
The forecast uncertainty comes from a part of the system that the operator cannot change.	Changing the weather is, practically speaking, impossible. As a result, wind and photovoltaic (PV) generation, as well as heating and air conditioning, can safely be considered exogenous. This makes them good candidates for direct machine learning forecasting. These forecasts can leverage meteorological forecasts or climatic scenarios, depending on the horizon. Meteorological forecasts are available only a few days ahead, though some models now predict trends over a few months.
The forecast uncertainty comes from a part of the system that the operator can somehow influence.	For example, strictly speaking, the consumption should not be forecasted, but rather the demand. The difference between consumption and demand is that the consumption is somehow at the hand of the operator that can choose not to serve the demand by switching off the consumers. For the same reason, the photovoltaic and wind production potential is forecasted, not the actual production.
The forecast uncertainty comes from a part of the system that some other actors can control and anticipate.	For example, for dispatchable power units where the operator can switch them on or off, it is better to ask for the schedules from the operator. If this is not possible, it may be better to reproduce the way the schedules are made—for instance, the operator may start the plant if the power price is higher than the plant fuel cost. In such cases, the forecasts may rely on techniques like agent-based modeling. Large factories are likely to have consumption schedules based on their operational production schedules. Distribution grid topology is also likely to be scheduled ahead of time, as maintenance operations require advanced planning. In all these cases, it is often better to ask for the schedules than to use machine learning to forecast them.

Spatial and Temporal Resolution

Due to the law of large numbers, the forecast uncertainty decreases when the consumption is spatially or temporally aggregated. While it is hard to forecast the hourly

consumption of an individual household because people are not machines, it is surprisingly easy for populations of a few million, and is relatively easy to forecast the monthly consumption of such a population as well.

As a result, a forecast system is often hierarchical, with several levels of forecasts that are linked together by constraints. That is, regional forecasts should sum up to the country-wide forecasts, and hourly forecasts should sum up to the daily forecast.

Let's take a striking example to illustrate this. Electric traction trains have a worrying consumption pattern for grid operators because they move, with a typical train line being fed by a different substation every 10 to 50 kilometers. As a result, the operator sees consumption of a few megawatts switching from substation to substation every 10 minutes or so. It creates several issues:

- The forecast is relatively easy at the line level because the train is always consuming somewhere and because trains usually circulate at fixed hours. As a result, a machine learning approach is likely to work.
- The forecast of the energy withdrawn over a long period at a given substation is also relatively easy, because the train will go through the corresponding part of the line.
- But because the operator wants to know if the train will create an overload when circulating, a consistent set of forecasts is needed:
  - The train should withdraw power at one location at a time only.
  - Each substation should see a consumption spike at some point in time so that a fine-grained time resolution is needed.

As a result, the solution depends on the goal of the prediction:

- On a day-to-day basis, an average solution that splits the train consumption over all substations is not acceptable, as potential overloads may be missed. A worst-case solution that assigns the train consumption to all substations may be more acceptable, though it will anticipate spurious overloads as the overall consumption will be too large.
- However, to schedule the maintenance of one of the lines that feeds the region, the exact location of the consumption is likely to have no impact as long as it is not counted several times.

When designing the forecast system, trade-offs will have to be made, as the perfect system is unlikely to exist. If the system has a lot of margin, few or no overloads are expected so that the forecasting system can be coarse. However, if the grid is operated close to its limits, the system has to be carefully crafted.

Implementation

Once data is collected, either by the SCADA system or by the metering system, it has to be historized. In addition to storing the raw data, some processing is required:

- Temporal aggregations, for example over a five-minute period: Either the average value or a high quantile value is kept. The average is representative of the energy consumed over the period, and the high quantile is useful to assess if constraints occurred.
- Disaggregations: When only the withdrawal is measured, the production and the consumption have to be separately estimated. Usually, consumption is what remains after removing the best possible estimation of distributed generation (wind, PV, etc.). Machine learning can be useful to perform these estimations.
- Spatial aggregations: As the system is balanced, it is possible to compute the consumption of a region by computing the difference between the local production and the exchanges with the neighboring regions. This was historically very useful because the production was easy to monitor because there were only a few very large generation units and a few lines with neighboring countries. Nowadays, it tends to be more complex as distributed generation is more widespread.
- Missing value imputation: A measurement may be missing. In the SCADA system, rules exist to replace a missing value with an older or a typical value in real time. In the metering system, the imputation is a heavy impact process as it will be reflected directly on the customer's invoice.

Data is then stored in different databases. Data used in short-term critical processes is stored in high-availability systems in which redundancy allows rapid recovery from the loss of a data center. Data used in longer-term processes (invoicing, reports, ML model training) is stored in ordinary IT databases. Overall, the number of monitored grid elements will range between 1,000 and 100,000. This means that they generate a reasonable volume of data by today's standards. Scalability is not such an issue either, as bulk power grids do not grow anymore in developed countries.

Modeling

Once the data preparation has been finished, the data scientist typically has access to a few hundred time series of production and consumption at various withdrawal points of the grid. They have to develop methods to predict some of them at various horizons. Their focus is usually on wind, PV, and sometimes run-of-the river hydroelectricity production potential and on demand. While wind and PV mainly depend on meteorological factors, the demand is mainly driven by economic activity, but part of it is also dependent on meteorology (for example heating and cooling).

Depending on the horizon, the modeling might look very different:

- Short-term: Up to a few days ahead, the last known values are very important to make predictions. In addition, for the same reasons, meteorological forecasts are available. Therefore, methods will leverage this information. In this case, deterministic forecasts make sense.
- Mid-term: Between a few days and a few years, the meteorology is not known, but the climate is. Statistical extrapolation of past year tendencies make sense, except if an economic crisis occurs. As a result, it is possible to draw scenarios to obtain statistical indicators (mean, confidence intervals, quantiles, etc.) about the future consumptions.
- Long-term: Investment decisions require forecasts over several decades. On this horizon, statistical extrapolations of the current trend are not enough, neither on the socio-economic side nor on the climatic side given global warming. As a result, statistical approaches have to be completed with bottom-up usage-based approaches and expert-made diversified scenarios about the future.

ML and MLOps mainly concern short-term and mid-term forecasts. Of the two, in this case, mid-term models are easier to start with: given a few years of data, the goal is to predict consumption based on:

- The calendar, with a superposition of daily, weekly, and annual cycles. Bank holidays and school vacations also have a big impact, in addition to daylight saving time.
- The meteorological variables (temperature, wind, sun). As buildings have very large thermal inertia, at least two days and up to three weeks of past temperatures may be needed.

While any kind of ML algorithm can be used, the smoothness of the predicted curve is important because the predictions are not used individually, but as daily, weekly, or annual scenarios. Many algorithms do not consider smoothness in their metrics because they rely on the hypothesis that the data is independent and identically distributed, which in our case is incorrect, since the consumption of a given day is usually correlated with the one of the previous day and the one of the previous week.

Generalized additive models (GAM) are often a good starting point: they are based on splines, so that the smoothness is guaranteed. In fact, consumption forecasting was one of the use cases for which they were developed. Combined with climatic scenarios, the ML model is then able to yield yearly consumption scenarios.

Short-term forecasts are more complex. The simplest way to proceed is to remove the mid-term forecast from the recent historical data and use standard time series techniques, such as ARIMA (autoregressive integrated moving average) or exponential

smoothing, on the residuals. This allows the generation of forecasts over several days. An integrated short-term model trained on several years of data has potential advantages over this simple approach.

For example, the mid-term model is trained on realized meteorological data and not on meteorological forecasts. As a result, it gives too much importance to meteorological forecasts even though they may be wrong. A short-term model trained on meteorological forecasts would address this issue. However, although new algorithms, such as long short-term memory (LSTM) neural networks, are promising, it is hard to find a method that allows for forecasting at any time of the day for several time horizons at once in a consistent way.

When the resolution is such that the stochasticity is too large to make meaningful predictions, it is better to aggregate time series spatially or temporally and then use non-ML heuristics to split the aggregated forecasts:

- A sharing key based on past observations in the case of spatial aggregation
- An average profile based on past observations in the case of temporal aggregation

Because the grid is under constant evolution, it is likely that new injections and withdrawals appear for which no historical data is available and that ruptures occur in consumption patterns, so that past data is not relevant anymore. The forecast method has to take into account these edge cases. Ruptures could be spotted using anomaly detection methods. As soon as a rupture is identified, a simplified model could be used for as long as necessary until enough historical data is available.

Once again, neural networks could become an appealing alternative with the promise that only one model could be trained for all the consumptions instead of one model per consumption with standard methods. Indeed, with only one model, the forecast of a consumption with shallow historical data would be possible provided that its pattern looks similar to an existing pattern.

Deployment

Nowadays, the models are likely to be prototyped by a data scientist in R, Python, or MATLAB scripts. The prototype is able to prepare the data, train the model on one dataset, and score it on another. The operationalization could follow several paths:

- The prototype is fully rewritten. This is costly and not flexible but may be necessary if embedding in an operational technology (OT) system is needed.
- Only the data preparation and the scoring are rewritten, which allows for training on a different schedule. It makes sense if the training occurs once a year or so because it is good practice to regularly perform a model review to ensure that it works well and that the skills to maintain it are in place.

- Data science and machine learning platforms can be used to operationalize the prototype. These platforms are flexible and allow the transfer of prototypes to production environments in which security and scalability are guaranteed. Most consumption forecast models will be run periodically in batch mode. For more specific use cases, these platforms are able to export trained models as JAR files, SQL, PMML, PFA, and ONNX so that they can be flexibly integrated into any kind of application.

Monitoring

This section mainly discusses short-term forecasts. Indeed, mid-term and long-term forecasts are systematically impacted by drift, as the past never looks like the future, so they are almost systematically trained again before being used to make predictions. For short-term forecasts, besides IT monitoring to raise alarms if forecasts are not produced on time and warnings for events that may result in missing deadlines, the models themselves should be monitored.

The first kind of monitoring is drift monitoring. For electricity consumption, it is critical that drift monitoring is deployed together with the model. Anomaly detection and rupture detection allow teams to make sure that the trained model can be used. If not, fallback models based on shallow historical data or normative disaggregation of multiple consumption forecasts should be used. This first layer will detect drastic drifts online.

Though the data scientist will try to design models that are adaptive to the consumption level (like ARIMA), it can be useful to detect that some consumption levels are higher or lower than in the training period. This may have happened slowly, so that it was not detected online. The offline analysis of the forecasts, for example once a month if the forecasts are computed every day for the next day, offers the possibility to detect these slow drifts. In these cases, if no additional ground truth is available, it would make sense to shift to a fallback model for these consumptions.

Lastly, after the operations, it is possible to assess the performance of the prediction through various metrics like mean absolute percentage error (MAPE). If a performance drop is detected during a significant amount of time (for example, one month), retraining the corresponding models is an option as new data is available, and the retrained models may increase performance.

This requires a tight integration of the design and the production environment with CI/CD processes (as discussed at length in Chapter 6). If it is possible to handle manually the deployment of new models once a year, it is usually too costly to do so once a month. With an advanced data science and machine learning platform, it is also possible to perform shadow scoring with the new model for a few days before using it for the forecasts.

Closing Thoughts

In this chapter, we have seen how to make the data speak to assist the operation of a transmission power grid. Various ML and non-ML techniques can be used to provide forecasts for up to thousands of consumptions on timescales ranging from minutes to decades.

Thanks to MLOps, design, deployment, and monitoring processes have been standardized across several industries, and data science and machine learning platforms have been developed to support this process. Designers of consumption forecast systems can leverage these standard processes and platforms to improve the efficiency of these systems from the cost, quality, or time to value perspective.

Taking a larger step back, it's clear that different industries have a wide range of machine learning use cases, all of which have their own intricacies when it comes to defining the problem, building models, pushing to production—everything we've covered in this book. But no matter what the industry or use case, MLOps processes are consistently the thread that allows data teams (and more widely, entire organizations) to scale their machine learning efforts.

Index

C

D

E