23
Conclusions and Next Steps

Our goal for this book was to enable you to apply **machine learning (ML)** to a variety of data sources and extract signals that add value to a trading strategy. To this end, we took a more comprehensive view of the investment process, from idea generation to strategy evaluation, and introduced ML as an important element of this process in the form of the **ML4T workflow**.

While demonstrating the use of a broad range of ML algorithms, from the fundamental to the advanced, we saw how ML can add value at multiple steps in the process of designing, testing, and executing a strategy. For the most part, however, we focused on the **core ML value proposition**, which consists of the ability to extract actionable information from much larger amounts of data more systematically than human experts would ever be able to.

This value proposition has really gained currency with the explosion of digital data that made it both more promising and necessary to leverage computing power to extract value from ever more diverse sets of information. However, the application of ML still requires significant human intervention and domain expertise to define objectives, select and curate data, design and optimize a model, and make appropriate use of the results.

Domain-specific aspects of using ML for trading include the nature of financial data and the environment of financial markets. The use of powerful models with a high capacity to learn patterns requires particular care to avoid overfitting when the signal-to-noise ratio is as low

as is often the case with financial data. Furthermore, the competitive nature of trading implies that patterns evolve quickly as signals decay, requiring additional attention to performance monitoring and model maintenance.

In this concluding chapter, we will briefly summarize the key tools, applications, and lessons learned throughout the book to avoid losing sight of the big picture after so much detail. We will then identify areas that we did not cover but would be worthwhile to focus on as you expand on the many ML techniques we introduced and become productive in their daily use.

In sum, in this chapter, we will:

- Review key takeaways and lessons learned
- Point out the next steps to build on the techniques in this book
- Suggest ways to incorporate ML into your investment process

Key takeaways and lessons learned

A central goal of the book was to demonstrate the workflow of extracting signals from data using ML to inform a trading strategy. *Figure 23.1* outlines this ML-for-trading workflow. The key takeaways summarized in this section relate to specific challenges we encounter when building sophisticated predictive models for large datasets in the context of financial markets:

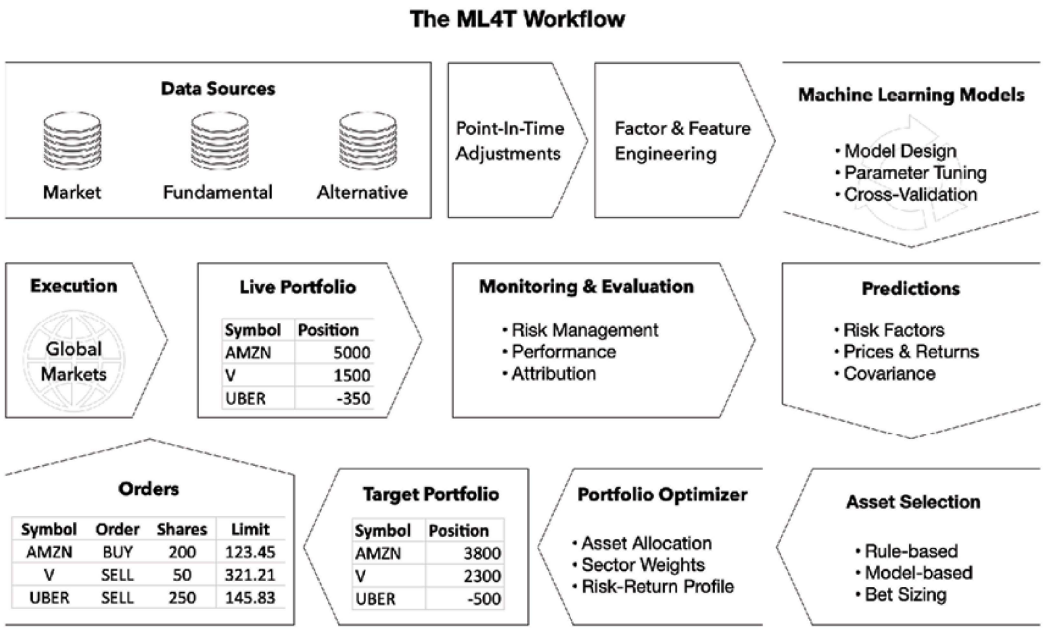

Figure 23.1: Key elements of using ML for trading

Important insights to keep in mind as you proceed to the practice of ML for trading include the following:

- **Data** is the single most important ingredient that requires careful sourcing and handling.
- **Domain expertise** is key to realizing the value contained in data and avoiding some of the pitfalls of using ML.
- ML offers **tools** that you can adapt and combine to create solutions for your use case.
- The choices of **model objectives and performance diagnostics** are key to productive iterations toward an optimal system.
- **Backtest overfitting** is a huge challenge that requires significant attention.
- **Transparency of black-box models** can help build confidence and facilitate the adoption of ML by skeptics.

We will elaborate a bit more on each of these ideas.

Data is the single most important ingredient

The rise of ML in trading and everywhere else largely complements the data explosion that we covered in great detail. We illustrated in *Chapter 2*, *Market and Fundamental Data – Sources and Techniques*, how to access and work with these data sources, historically the mainstay of quantitative investment. In *Chapter 3*, *Alternative Data for Finance – Categories and Use Cases*, we laid out a framework with criteria to assess the potential value of alternative datasets.

A key insight is that state-of-the-art ML techniques like deep neural networks are successful because their predictive performance continues to improve with more data. On the flip side, model and data complexity need to match to balance the bias-variance trade-off, which becomes more challenging the higher the noise-to-signal ratio of the data is. Managing data quality and integrating datasets are key steps in realizing the potential value.

The new oil? Quality control for raw and intermediate data

Just like oil, a popular comparison these days, data passes through a **pipeline with several stages** from its raw form to a refined product that can fuel a trading strategy. Careful attention to the quality of the final product is critical to getting the desired mileage out of it.

Sometimes, you get **data in its raw form** and control the numerous transformations required for your purposes. More often, you deal with an **intermediate product** and should get clarity about what exactly the data measures at this point.

Different from oil, there is often **no objective quality standard** as data sources continue to proliferate. Instead, the quality depends on its signal content, which in turn depends on your investment objectives. The cost-effective evaluation of new datasets requires a productive workflow, including appropriate infrastructure that we will address later in this chapter.

Data integration – the whole exceeds the sum of its parts

The value of data for an investment strategy often depends on combining complementary sources of market, fundamental, and alternative data. We saw that the predictive power of ML algorithms, like tree-based ensembles or neural networks, is in part due to their ability to detect nonlinear relationships, in particular **interaction effects among variables**.

The ability to modulate the impact of a variable as a function of other model features thrives on data inputs that capture different aspects of a target outcome. The combination of asset prices with macro fundamentals, social sentiment, credit card payment, and satellite data will likely yield significantly more reliable predictions throughout different economic and market regimes than each source on its own (provided the amount of data is large enough to learn the hidden relationships).

Working with data from multiple sources increases the **challenges of proper labeling**. It is vital to assign accurate timestamps that accurately reflect historical publication. Otherwise, we introduce look-ahead bias by testing an algorithm with data before it actually becomes available. For example, third-party data may have timestamps that require adjustments to reflect the point in time when the information would have been available for a live algorithm.

Domain expertise – telling the signal from the noise

We emphasized that informative data is a necessary condition for successful ML applications. However, domain expertise is equally essential to define the strategic direction, select relevant data, engineer informative features, and design robust models.

In any domain, practitioners have theories about the drivers of key outcomes and relationships among them. Finance is characterized by a **large amount of available quantitative research**, both theoretical and empirical. However, Marcos López de Prado and others (Cochrane 2011) criticize most empirical results: claims of predictive signals found in hundreds of variables are often based on pervasive data mining and are not robust to changes in the experimental setup. In other words, statistical significance often results from large-scale trial-and-error rather than a true systematic relationship, along the lines of "if you torture the data long enough, it will confess."

On the one hand, there exists a robust understanding of how financial markets work. This should inform the selection and use of data as well as the justification of strategies that rely on ML. An important reason is to prioritize ideas that are more likely to be successful and avoid the multiple testing trap that leads to unreliable results. We outlined key ideas in *Chapter 4*, *Financial Feature Engineering – How to Research Alpha Factors*, and *Chapter 5*, *Portfolio Optimization and Performance Evaluation*.

On the other hand, novel ML techniques will likely uncover new hypotheses about drivers of financial outcomes that will inform theory and should then be independently tested.

More than the raw data, feature engineering is often the key to making signals useful for an algorithm. Leveraging decades of research into risk factors that drive returns on theoretical and empirical grounds is a good starting point to prioritize data transformations that are more likely to reflect relevant information.

However, only creative feature engineering will lead to innovative strategies that can compete in the market over time. Even for new alpha factors, a compelling narrative that explains how they work given established ideas on market dynamics and investor behavior will provide more confidence to allocate capital.

The risks of false discoveries and overfitting to historical data make it even more necessary to prioritize strategies prior to testing rather than "letting the data speak." We covered how to deflate the Sharpe ratio in *Chapter 7*, *Linear Models – From Risk Factors to Return Forecasts*, to account for the number of experiments.

ML is a toolkit for solving problems with data

ML offers algorithmic solutions and techniques that can be applied to many use cases. *Parts 2*, *3*, and *4* of this book have presented ML as a diverse set of tools that can add value to various steps of the strategy process, including:

- Idea generation and alpha factor research
- Signal aggregation and portfolio optimization
- Strategy testing
- Trade execution
- Strategy evaluation

Moreover, ML algorithms are designed to be further developed, adapted, and combined to solve new problems in different contexts. For these reasons, it is important to understand key concepts and ideas underlying these algorithms, in addition to being able to apply them to data for productive experimentation and research as outlined in *Chapter 6*, *The Machine Learning Process*, and summarized in *Figure 23.2*:

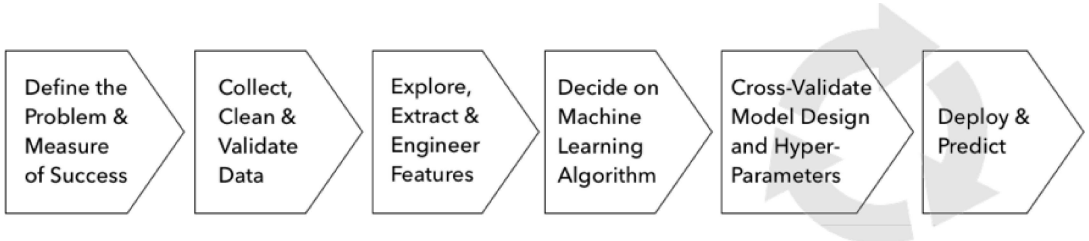

Figure 23.2: The ML workflow

Furthermore, the best results are often achieved by **human-in-the-loop solutions** that combine humans with ML tools. In *Chapter 1*, *Machine Learning for Trading – From Idea to Execution*, we covered the quantamental investment style where discretionary and algorithmic trading converge. This approach will likely further grow in importance and depends on the flexible and creative application of the fundamental tools that we covered and their extensions to a variety of datasets.

Model diagnostics help speed up optimization

In *Chapter 6*, *The Machine Learning Process*, we outlined the most important ML-specific concepts. ML algorithms learn relationships between input data and a target by making assumptions about the functional form. If the learning is based on noise rather than signal, predictive performance will suffer.

Of course, we do not know today how to separate signal and noise from the perspective of tomorrow's outcomes. Careful cross-validation that avoids lookahead bias and robust model diagnostics, such as learning curves and the optimization verification test, can help alleviate this fundamental challenge and calibrate the choice or configuration of an algorithm. This task can be made easier by defining focused model objectives and, for complex models, distinguishing between performance shortcomings due to issues with the optimization algorithm and those with the objective itself.

Making do without a free lunch

No system, whether a computer program or a human, can reliably predict outcomes for new examples beyond those it has observed during training. The only way out is to have some additional prior knowledge or make assumptions that go beyond the training examples. We covered a broad range of algorithms from linear models in *Chapter 7*, *Linear Models – From Risk Factors to Return Forecasts*, to nonlinear

ensembles in *Chapter 11, Random Forests – A Long-Short Strategy for Japanese Stocks*, and *Chapter 12, Boosting Your Trading Strategy*, as well as neural networks in various chapters of *Part 4* of this book.

We saw that a linear model makes the strong assumption that the relationship between inputs and outputs has a very simple form, whereas nonlinear models like gradient boosting or neural networks aim to learn more complex functions. While it's probably obvious that a simple model will fail in most circumstances, a complex model is not always better. If the true relationship is linear but the data is noisy, the complex model will learn the noise as part of the complex relationship that it assumes to exist. This is the basic idea behind the **"no free lunch" theorem**, which states that no algorithm is universally superior for all tasks. Good fit in some instances comes at the cost of poor performance elsewhere.

The key tools to tailor the choice of the algorithm to the data are data exploration and experiments based on an understanding of the assumptions the model makes.

Managing the bias-variance trade-off

A key challenge in adapting an algorithm to data is the trade-off between bias and variance, which both increase prediction errors beyond the natural noisiness of the data. A simple model that does not adequately capture the relationships in the data will underfit and exhibit bias, that is, make systematically wrong predictions. A model that is too complex will overfit and learn the noise in addition to any signal so that the result will show a lot of variance for different samples.

The key tool to diagnose this trade-off at any given iteration of the model selection and optimization process is the **learning curve**. It shows how training and validation errors depend on the sample size. This allows us to decide between different options to improve performance: adjust the complexity of the model or get more data points.

The closer the training error is to human performance or another benchmark, the more likely the model will overfit. A low validation error tells us that we are lucky and found a good model. If the validation error is high, we are not. If it continues to decline with the training size, however, more data may help. If the training error is high, more data is unlikely to help, and we should instead add features or use a more flexible algorithm.

Defining targeted model objectives

One of the first steps in the ML process is the definition of an objective for the algorithm to optimize. Sometimes, the choice is simple, such as in a regression problem. A classification task can be more difficult, for example, when we care about precision and recall. Consolidating conflicting objectives into a single metric like the F1 score helps to focus optimization efforts. We can also include conditions that need to be met rather than optimized for. We also saw that reinforcement learning is all about defining the right reward function to guide the agent's learning process.

The optimization verification test

Andrew Ng emphasizes the distinction between performance shortcomings due to a problem with the learning algorithm or the optimization algorithm. Complex models like neural networks assume nonlinear relationships, and the search process of the optimization algorithm may end up in a local rather than a global optimum.

If a model fails to correctly translate a phrase, for example, the test compares the scores for the correct prediction and the solution discovered by the search algorithm. If the learning algorithm scores the correct solution higher, the search algorithm requires improvements. Otherwise, the learning algorithm is optimizing for the wrong objective.

Beware of backtest overfitting

We covered the risks of false discoveries due to overfitting to historical data repeatedly throughout the book. *Chapter 5*, *Portfolio Optimization and Performance Evaluation*, on strategy evaluation, lays out the main drivers and potential remedies. The low noise-to-signal ratio and relatively small datasets (compared to web-scale image or text data) make this challenge particularly serious in the trading domain. Awareness is critical since the ease of access to data and tools to apply ML increases the risks significantly.

There are no easy answers because the risks are inevitable. However, we presented methods to adjust backtest metrics to account for repeated trials, such as the deflated Sharpe ratio. When working toward a live trading strategy, staged paper-trading and closely monitored performance during execution in the market need to be part of the implementation process.

How to gain insights from black-box models

Deep neural networks and complex ensembles can raise suspicion when they are considered impenetrable black-box models, particularly in light of the risks of backtest overfitting. We introduced several methods to gain insights into how these models make predictions in *Chapter 12*, *Boosting Your Trading Strategy*.

In addition to conventional measures of feature importance, the recent game-theoretic innovation of **SHapley Additive exPlanations (SHAP)** is a significant step toward understanding the mechanics of complex models. SHAP values allow the exact attribution of features and their values to predictions so that it becomes easier to validate the logic of a model in the light of specific theories about market behavior for a given investment target. Besides justification, exact feature importance scores and attribution of predictions allow deeper insights into the drivers of the investment outcome of interest.

On the other hand, there is some controversy over how important transparency around model predictions should be. Geoffrey Hinton, one of the inventors of deep learning, argues that the reasons for human decisions are often obscure. Perhaps machines should be evaluated by their results, just as we do with investment managers.

ML for trading in practice

As you proceed to integrate the numerous tools and techniques into your investment and trading process, there are numerous things you can focus your efforts on. If your goal is to make better decisions, you should select projects that are realistic yet ambitious given your current skill set. This will help you to develop an efficient workflow underpinned by productive tools and gain practical experience.

We will briefly list some of the tools that are useful to expand on the Python ecosystem covered in this book. They include big data technologies that will eventually be necessary to implement ML-driven trading strategies at scale. We will also list some of the platforms that allow you to implement trading strategies using Python, possibly with access to data sources, and ML algorithms and libraries. Finally, we will point out good practices for adopting ML as an organization.

Data management technologies

The central role of data in the ML4T process requires familiarity with a range of technologies to store, transform, and analyze data at scale, including the use of cloud-based services like Amazon Web Services, Microsoft Azure, and Google Cloud.

Database systems

Data storage implies the use of databases. Historically, these have typically been **relational database management systems (RDBMSes)** that use SQL to store and retrieve data in a well-defined table format.

These have included databases from commercial providers like Oracle and Microsoft and open-source implementations like PostgreSQL and MySQL. More recently, non-relational alternatives have emerged that are often collectively labeled NoSQL but are quite diverse, namely:

- **Key-value storage**: Fast read/write access to objects. We covered the HDF5 format in *Chapter 2*, *Market and Fundamental Data – Sources and Techniques*, which facilitates fast access to a pandas DataFrame.
- **Columnar storage**: Capitalizes on the homogeneity of data in a column to facilitate compression and faster column-based operations like aggregation. This is used in the popular Amazon Redshift data warehouse solution, Apache Parquet, Cassandra, and Google's Big Table.
- **Document store**: Designed to store data that defies the rigid schema definition required by an RDBMS. This has been popularized by web applications that use JSON or XML format, which we encountered in *Chapter 4*, *Financial Feature Engineering – How to Research Alpha Factors*. It is used, for example, in MongoDB.
- **Graph database**: Designed to store networks that have nodes and edges and specializes in queries about network metrics and relationships. It is used in Neo4J and Apache Giraph.

There has been some convergence toward the conventions established by the relational database systems. The Python ecosystem facilitates the interaction with many standard data sources and provides the fast HDF5 and Parquet formats, as demonstrated throughout the book.

Big data technologies – from Hadoop to Spark

Data management at scale for hundreds of gigabytes and beyond requires the use of multiple machines that form a cluster to conduct read, write, and compute operations in parallel. In other words, you need a distributed system that operates on multiple machines in an integrated way.

The **Hadoop ecosystem** has emerged as an open-source software framework for distributed storage and processing of big data using the MapReduce programming model developed by Google. The ecosystem has diversified under the roof of the Apache Foundation and today includes numerous projects that cover different aspects of data management at scale.

Key tools within Hadoop include:

- **Apache Pig**: A data processing language, developed at Yahoo, for implementing large-scale **extract-transform-load** (**ETL**) pipelines using MapReduce.
- **Apache Hive**: The de facto standard for interactive SQL queries over petabytes of data. It was developed at Facebook.
- **Apache HBASE**: A NoSQL database for real-time read/write access that scales linearly to billions of rows and millions of columns. It can combine data sources using a variety of different schemas.

Apache Spark has become the most popular platform for interactive analytics on a cluster. The MapReduce framework allowed parallel computation but required repeated read/write operations from disk to ensure data redundancy. Spark has dramatically accelerated computation at scale due to the **resilient distributed data** (**RDD**) structure, which allows highly optimized in-memory computation. This includes iterative computation as required for optimization, for example, gradient descent for numerous ML algorithms. Fortunately, the Spark DataFrame interface has been designed with pandas in mind so that your skills transfer relatively smoothly.

ML tools

We covered many libraries of the Python ecosystem in this book. Python has evolved to become the language of choice for data science and ML. The set of open-source libraries continues to both diversify

and mature, and is built on the robust core of scientific computing libraries NumPy and SciPy.

The popular pandas library has contributed significantly to popularizing the use of Python for data science and has matured with its 1.0 release in January 2020. The scikit-learn interface has become the standard for modern, specialized ML libraries like XGBoost or LightGBM that often interface with the workflow automation tools like GridSearchCV and Pipeline that we have used repeatedly throughout the book.

There are several providers that aim to facilitate the ML workflow:

- **H2O.ai** offers the H2O platform, which integrates cloud computing with ML automation. It allows users to fit thousands of potential models to their data to explore patterns in the data. It has interfaces in Python as well as R and Java.
- **Datarobot** aims to automate the model development process by providing a platform to rapidly build and deploy predictive models in the cloud or on-premises.
- **Dataiku** is a collaborative data science platform designed to help analysts and engineers explore, prototype, build, and deliver their own data products.

There are also several open-source initiatives led by companies that build on and expand the Python ecosystem:

- The quantitative hedge fund **TwoSigma** contributes quantitative analysis tools to the Jupyter Notebook environment under the BeakerX project.
- **Bloomberg** has integrated the Jupyter Notebook into its terminal to facilitate the interactive analysis of its financial data.

Online trading platforms

The main options to develop trading strategies that use ML are online platforms, which often look for and allocate capital to successful trading strategies. Popular solutions include Quantopian, Quantconnect, and QuantRocket. The more recent Alpha Trading Labs focuses on high-frequency trading. In addition, **Interactive Brokers (IB)** offers a Python API that you can use to develop your own trading solution.

Quantopian

We introduced the Quantopian platform and demonstrated the use of its research and trading environment to analyze and test trading strategies against historical data. Quantopian uses Python and offers a lot of educational material.

Quantopian hosts competitions to recruit algorithms for its crowd-sourced hedge fund portfolio. It provides capital to the winning algorithm. Live trading was discontinued in September 2017, but the platform still provides a large range of historical data and attracts an active community of developers and traders. It is a good starting point to discuss ideas and learn from others.

QuantConnect

QuantConnect is another open-source, community-driven algorithmic trading platform that competes with Quantopian. It also provides an IDE to backtest and live trade algorithmic strategies using Python and other languages.

QuantConnect also has a dynamic, global community from all over the world, and provides access to numerous asset classes, including equities, futures, FOREX, and cryptocurrency. It offers live trading integration with various brokers, such as IB, OANDA, and GDAX.

QuantRocket

QuantRocket is a Python-based platform for researching, backtesting, and running automated quantitative trading strategies. It provides data collection tools, multiple data vendors, a research environment, multiple backtest engines, and live and paper trading through IB. It prides itself on support for international equity trading and sets itself apart with its flexibility (but Quantopian is working toward this as well).

QuantRocket supports multiple engines — its own Moonshot, as well as third-party engines as chosen by the user. While QuantRocket doesn't have a traditional IDE, it is integrated well with Jupyter to produce something similar. QuantRocket offers a free version with access to sample data, but access to a wider set of capabilities starts at $29 per month at the time of writing in early 2020.

Conclusion

We started by highlighting the explosion of digital data and the emergence of ML as a strategic capability for investment and trading strategies. This dynamic reflects global business and technology trends beyond finance and is much more likely to continue than to stall or reverse. Many investment firms are just getting started to leverage the range of artificial intelligence tools, just as individuals are acquiring the relevant skills and business processes are adapting to these new opportunities for value creation, as outlined in the introductory chapter.

There are also numerous exciting developments for the application of ML to trading on the horizon that are likely to propel the current momentum. They are likely to become relevant in the coming years and include the automation of the ML process, the generation of synthetic training data, and the emergence of quantum computing. The extraordinary vibrancy of the field implies that this alone could fill a book and the journey will continue to remain exciting.