Monograph

AI in Asset Management: Tools, Applications, and Frontiers

Joseph Simonian, PhD
Editor

CFA Institute Research Foundation

CFA Institute Research & Policy Center

AI in Asset Management: Tools, Applications, and Frontiers

Joseph Simonian, PhD
Editor

CFA Institute Research Foundation

CFA Institute Research & Policy Center

Statement of Purpose

CFA Institute Research Foundation is a not-for-profit organization established to promote the development and dissemination of relevant research for investment practitioners worldwide.

Photo credit: Loomis Creative

Print ISBN: 978-1-952927-64-5

Ebook ISBN: 978-1-952927-65-2

AUTHOR BIOS

Paul Bilokon, PhD, is the Chief Executive Officer of Thalesians Ltd. and a Visiting Professor at Imperial College London. He has over two decades of experience as a quantitative analyst, strategist, trader, and portfolio manager, holding senior roles at Morgan Stanley, Citigroup, and Deutsche Bank. He is the co-author of *Machine Learning in Finance: From Theory to Practice* and has developed cross-asset trading and risk systems in multiple programming languages. Dr. Bilokon has published research across fields such as artificial intelligence, high-performance computing, and finance and has mentored numerous startups at London's Level39 incubator. He earned his education from Oxford University's Christ Church College and Imperial College London.

Francesco A. Fabozzi, PhD, is research director at Yale School of Management's International Center for Finance and director of data science at CFA Institute Research Foundation. He has extensive experience in quantitative investing, data science, and financial econometrics. His research explores applications of large language models, natural language processing, and machine learning in investment management. Dr. Fabozzi has published widely in leading journals such as *The Journal of Financial Econometrics* and *The Journal of Portfolio Management* and serves as managing editor of *The Journal of Financial Data Science*. Through his editorial and leadership roles, he advances the integration of artificial intelligence and data-driven methods in asset management.

Maxim Golts, PhD, is vice president of multi-asset solutions at PGIM, and a senior advisor at Boston University's Questrom School of Business. He has previously held investment and research roles at Acadian Asset Management, State Street Global Advisors, Fidelity Investments, and GMO. Dr. Golts has contributed to the advancement of quantitative strategies and portfolio management across global markets. Before entering finance, he taught mathematics and statistics at Boston University, MIT, and Duke University. He earned his PhD in mathematics from Yale University.

Tony Guida is a quantitative portfolio manager and researcher at Atonra, where he leads efforts to integrate large language models and knowledge graphs into investment processes. He has built and managed systematic equity and macro strategies at major financial institutions, including Unigestion, EDHEC-Risk Scientific Beta, and a UK pension fund. Mr. Guida has authored several influential books, such as *Big Data and Machine Learning in Quantitative Investment* and *Machine Learning for Factor Investing*. He serves on the advisory board of the Financial Data Professional Institute and regularly reviews academic journals in finance. He holds bachelor's and master's degrees in econometrics and a master's degree in economics and finance.

Igor Halperin, PhD, is group data science leader for GenAI Asset Management Technology at Fidelity Investments, where he applies artificial intelligence, reinforcement learning, and information theory to financial decision-making. He has previously served as executive director of quantitative research at JPMorgan, a quantitative researcher at Bloomberg LP, and a research professor of financial machine learning at NYU. Dr. Halperin has published widely and co-authored *Machine Learning in Finance: From Theory to Practice*. In February 2022, he was

named "Buy-Side Quant of the Year" by *Risk* magazine. Dr. Halperin holds a PhD in theoretical high energy physics from Tel Aviv University and an MSc in nuclear physics from St. Petersburg State Technical University.

Petter N. Kolm, PhD, is a clinical professor at New York University's Courant Institute, where he directs the Mathematics in Finance master's program. He received the 2021 Quant of the Year award from *Portfolio Management Research* for his contributions to quantitative portfolio theory. Mr. Kolm previously worked in quantitative strategies at Goldman Sachs Asset Management and currently serves on several corporate and editorial boards. His expertise spans machine learning, portfolio theory, and systematic trading. He holds a PhD in mathematics from Yale University, an MPhil in applied mathematics from the Royal Institute of Technology (KTH), and an MS in mathematics from ETH Zurich.

Gueorgui S. Konstantinov, PhD, is senior portfolio manager in fixed income and currencies at DekaBank, where he oversees research, strategy development, and portfolio management. He has nearly two decades of experience in quantitative investing, risk management, and multi-asset strategies. Dr. Konstantinov serves on the editorial advisory boards of *The Journal of Portfolio Management* and *The Journal of Financial Data Science* and is executive director for the German chapter of CAIA. He holds a doctorate in international finance and an MSc in economics from Vienna University of Economics and Business, along with CAIA and FDP designations.

Anna Martirosyan is strategy and transactions manager at EY Parthenon, where she leads M&A and financial due diligence projects with a focus on banking and capital markets. She began her career at EY Armenia and has since developed expertise in strategic advisory and valuation. Ms. Martirosyan teaches finance and economics courses and conducts research on ethical AI and quantitative methods in finance. She has been published in *The Journal of Portfolio Management* and coordinates EU policy programs supporting economic development in Eastern Europe. She holds an MBA from Hult International Business School and an MS in economics and finance.

Mona Naqvi is managing director, research, advocacy, and standards at CFA Institute, where she oversees the Research and Policy Center, advancing independent research, policy, and standards for the investment profession. Previously, she held senior leadership roles at S&P Global, including global head of sustainable capital markets and investment research. Earlier in her career, she advised the Obama Administration on climate policy, led research at the 2° Investing Initiative, and served as an economist at the Bank of England. Mona graduated *summa cum laude* in government and economics from the London School of Economics and has received multiple industry honors for her leadership in sustainable finance.

Gordon Ritter, PhD, is a partner at Ritter Alpha LP and an adjunct professor at NYU's Courant Institute and Tandon School of Engineering. He is a leader in systematic trading and portfolio optimization, having founded Ritter Alpha to apply advanced statistical and scientific methods to investment management. Dr. Ritter has published extensively in leading finance journals and was named "Buy-Side Quant of the Year" in 2019 and "Quant Educator of the Year" in 2024. Previously, he held senior quantitative roles at GSA Capital and Highbridge Capital Management. He earned his PhD in physics and a master's degree from Harvard University.

Agathe Sadeghi, PhD, is a quant research fellow at Uniswap Labs, where she applies network theory and causal inference to decentralized finance. Her research explores risk factors, market dynamics, and the interaction of participants in digital asset ecosystems. Dr. Sadeghi previously completed a quant analytics internship at Bloomberg, where her work on causal network discovery gained recognition in both academia and industry. She earned her PhD in financial engineering from Stevens Institute of Technology and continues to advance responsible innovation in financial data science.

Joseph Simonian, PhD, is founder and chief investment officer of Autonomous Investment Technologies and senior affiliate researcher at CFA Institute. He has more than 20 years of experience in portfolio management, machine learning, and quantitative investing. Dr. Simonian serves on the editorial boards of *The Journal of Portfolio Management* and *The Journal of Financial Data Science* and has authored over 40 publications, including *Computational Global Macro*. He frequently lectures on quantitative methods and investment strategy. Dr. Simonian earned his PhD from the University of California, Santa Barbara.

Alireza Yazdani, PhD, is senior vice president of applied artificial intelligence at Citi Issuer & Investor Services. He has nearly two decades of experience in quantitative research, fintech, and applied machine learning for credit and risk management. Dr. Yazdani has held academic and professional appointments in data science and fintech education and is a frequent conference speaker and author in applied mathematics and finance. He holds a PhD in applied mathematics and additional executive certifications in data science and AI strategy.

Oswaldo Zapata, PhD, is cofounder of The Quantum Finance Boardroom, an online community advancing the use of quantum computing in finance. He is dedicated to bridging theory and practice through publications, professional outreach, and educational e-books on quantum finance. Dr. Zapata actively collaborates with global finance and technology experts to promote innovation in the field. He holds a PhD in theoretical physics.

CONTENTS

PROFESSIONAL LEARNING QUALIFIED ACTIVITY
This publication qualifies for 5.5 PL credits, inclusive of 0.25 SER credits, under the guidelines of the CFA Institute Professional Learning Program.

FOREWORD

Mona Naqvi
Managing Director, Research, Advocacy, and Standards
CFA Institute

Artificial intelligence (AI) continues to reshape the very foundations of our financial system. From portfolio construction to client engagement, its reach seems to expand daily across every facet of investment practice, challenging long-held assumptions about how to create, measure, and deliver value. The pace of this transformation demands not only technical adaptation but also ethical clarity and professional leadership.

At CFA Institute, our mission is to lead the investment profession globally by promoting the highest standards of ethics, education, and professional excellence. In the age of AI, this mission takes on new urgency. The transformative power of this technology must be unlocked in ways that strengthen—not supplant—human judgment, trust, and fiduciary responsibility. We stand at a pivotal moment, when innovation and integrity must advance hand in hand.

This publication, *AI in Asset Management: Tools, Applications, and Frontiers*, represents a collective effort to illuminate that path forward. Produced in collaboration between CFA Institute Research Foundation and CFA Institute Research and Policy Center, it draws on the insights of leading scholars, technologists, and investment practitioners who are shaping the frontiers of the field. Their contributions reflect the dynamism of the ecosystem already at work: pushing boundaries, testing models, and redefining what it means to make investment decisions in a world of intelligent machines.

Building on the 2023 *Handbook of Artificial Intelligence and Big Data Applications in Investments*, this compendium advances our exploration of how machine learning, natural language processing, and data analytics continue to transform the craft of investing. More than a technical guide, it represents an invitation to think critically, to experiment responsibly, and to ensure that innovation serves the enduring goals of financial security and societal progress.

We believe that the future of finance depends not on the technology itself but on how wisely we apply it. Through this and related work, we aim to empower practitioners to navigate the digital transition with competence, confidence, and conscience.

This publication would not have been possible without the deep engagement of the experts, authors, and reviewers who generously shared their knowledge and perspective. Their work exemplifies the spirit of collaboration that will be essential as our profession enters this next frontier. My thanks to all of them for their insightful and learned contributions.

As we collectively shape the future of finance in the age of AI, may this volume serve as both a guide and a catalyst for understanding, for innovation, and for leadership grounded in purpose.

PREFACE

Joseph Simonian, PhD
Senior Affiliate Researcher
CFA Institute

The integration of artificial intelligence (AI) and machine learning has become increasingly prevalent across the investment industry. Although we remain in the nascent stages of widespread adoption, these technologies are undeniably revolutionizing how investors analyze and interpret financial markets. Therefore, investment professionals must stay abreast of the latest developments in AI, machine learning, and data science.

CFA Institute continues to lead in educating investors and investment professionals on emerging topics, particularly the evolving applications of AI within investment management. In 2023, CFA Institute Research Foundation published an AI handbook that provided a comprehensive overview of the field's core areas.[1] This current volume not only updates that foundational work but also presents fresh insights from distinguished investment practitioners. Although this publication addresses all major branches of AI, each chapter offers a unique perspective on the technology's relevance and practical applications across diverse investment contexts.

We encourage you to use this guide as a navigational compass, helping you chart a course through the often turbulent waters of technological innovation and its intersection with finance. It examines both the achievements realized and the future trajectory of AI applications in portfolio management, risk analysis, and trading.

The unifying theme throughout these chapters is, of course, how AI applications can deepen our understanding of market phenomena. This theme serves as the cohesive thread binding the handbook together. However, we have afforded the authors considerable stylistic latitude to articulate their ideas and structure their chapters in ways they deem most impactful. Similar to an ensemble of jazz musicians performing together, they must harmonize with one another while remaining free to express their perspectives in the manner they consider most effective. With that in mind, note that each chapter prioritizes the practical dimensions of the frameworks discussed. Although pertinent mathematical details are not omitted, the emphasis clearly rests on demonstrating how AI can enhance investors' daily practice.

Ultimately, this volume represents not the conclusion of the financial data science story but, rather, a chapter written during a pivotal moment in technological development and market innovation. We hope this handbook will serve as both inspiration and an informative roadmap for practitioners, enriching and expanding their quantitative arsenals. This is indeed an exciting time to be an investment professional. With the overwhelming amount of information being generated, however, distinguishing signal from noise has become increasingly challenging. If this work can help readers cut through the "data fog" by illuminating the various ways AI tools can assist their day-to-day work, we will consider our mission accomplished.

---

[1] Larry Cao, *Handbook of Artificial Intelligence and Big Data Applications in Investments* (Charlottesville, VA: CFA Institute Research Foundation, 2023).

UNSUPERVISED LEARNING I: OVERVIEW OF TECHNIQUES

Joseph Simonian, PhD
Senior Affiliate Researcher
CFA Institute

Unsupervised learning is a branch of machine learning that encompasses algorithms used to discover hidden patterns and structures in data without labeled examples from which to learn. Unlike supervised learning, there is no "ground truth" to guide the learning process, which means that the algorithm must discover hidden patterns and relationships in data without any explicit guidance from real-word observations regarding what constitutes the correct answer. Without ground truths, unsupervised learning algorithms must rely on mathematical principles, such as maximizing likelihood or minimizing error, to capture the essence of the data. This makes unsupervised learning both an art and a science, requiring careful consideration of what constitute meaningful patterns versus mere noise.

In financial contexts, unsupervised learning can be particularly useful because financial markets are often opaque, labeled data are often scarce or expensive to obtain, or such data quickly become obsolete. In other words, the "correct" answer is often elusive to varying degrees. Financial markets are also dynamic, and as market regimes change, new patterns emerge and traditional relationships often break down. In such cases, unsupervised learning methods can be invaluable in helping practitioners discover structures in financial data that may prove valuable in their portfolio and risk management efforts.

Clustering

Perhaps the most well-known framework for unsupervised learning is clustering. Simpler clustering algorithms, such as *k-means clustering* (Lloyd 1982), operate according to a criterion of compactness, with observations grouped into different clusters based on their distance from designated *centroids*. These centroids are the average (mean) positions of all the data points that belong to a particular cluster. The algorithm for *k*-means clustering is shown in **Figure 1**.

A *k*-means clustering approach makes a good choice when data are numeric, clusters are roughly spherical and similar in size, and a fast, scalable clustering for large datasets is needed. It is mathematically simple, efficient, and easy to interpret. However, *k*-means also assumes that clusters are spherical and equal sized, which is not always the case. Further, it is sensitive to initialization and outliers and requires a specification of the number of clusters, *k*. Finally, *k*-means clustering can detect only clusters that are linearly separable, limiting its usefulness in applications in which nonlinear or otherwise nuanced relationships are present.

With the foregoing in mind, however, note that *k*-means has nevertheless been applied to portfolio construction. For example, Wu, Wang, and Wu (2022) used *k*-means to cluster stocks according to their continuous trend characteristics and then used inverse volatility weighting, risk parity, and mean–variance-type considerations to arrive at final portfolio weights.

Figure 1. Algorithm: *k*-Means Clustering

Input: Dataset X with n points, number of clusters k, maximum iterations

Output: Cluster assignments and centroids

Begin:

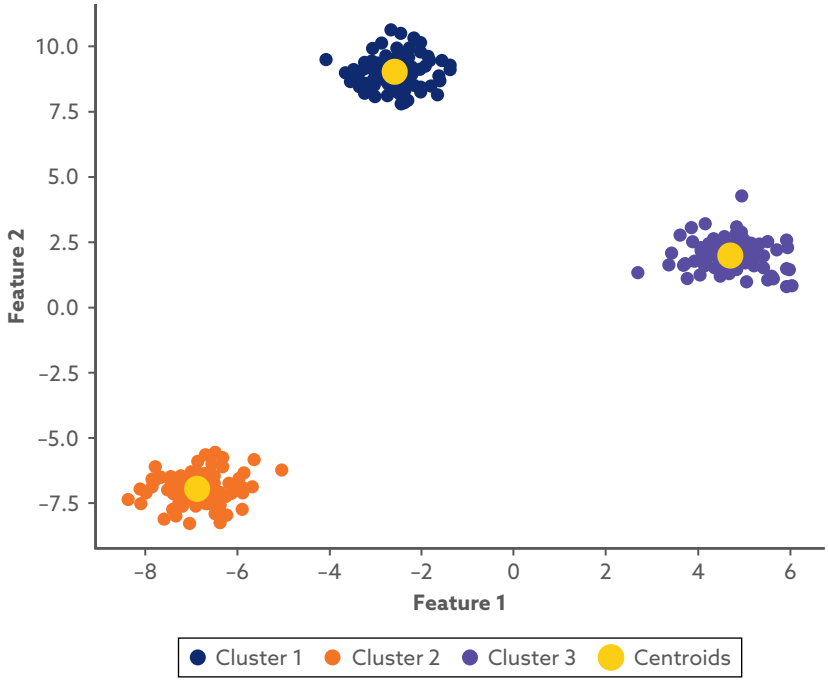
1. **Initialize *k* centroids randomly:** $\mu_1, \mu_2, \ldots, \mu_k$
2. **For iteration = 1 to max iterations:**
   - **a.** For each data point x_i in X:
     - Calculate distance to each centroid: $d(x_i, \mu_j)$ for $j = 1$ to k
     - Assign x_i to closest centroid: $c_i = \text{argmin}_j\, d(x_i, \mu_j)$
   - **b.** For each cluster $j = 1$ to k:
     - Update centroid: μ_j = mean of all points assigned to cluster j
   - **c.** If centroids have not changed significantly:
     - Break (convergence achieved)
3. **Output:** Cluster assignments and final centroids (see **Exhibit 1**)

An alternative clustering algorithm that provides a remedy to the limitations of *k*-means clustering is *spectral clustering*, which involves using matrix representations of finite graphs in order to determine the similarity between observations in a dataset (see **Figure 2**). Indeed, any set of observations or set of vectors of observations may be represented in graphical form. Spectral clustering can thus be viewed as a graph partition problem, in which clusters correspond to connected graph components.

The basic ideas behind spectral clustering were introduced in important papers by Hall (1970), Donath and Hoffman (1973), and Fiedler (1973). For a historical overview of spectral clustering, see Spielman and Teng (2007). Spectral clustering has also been applied in finance. In portfolio management and financial network analysis, spectral clustering excels at identifying groups of correlated assets, detecting market sectors based on complex interdependencies, and analyzing systemic risk by revealing the underlying network structure of financial markets where assets may be connected through indirect relationships that are not apparent in the original feature space. For example, Simonian and Wu (2019) used spectral clustering to build a regime-based trading model. They showed that their framework both produces predictively effective macro signals and classifies regimes in an economically intuitive way. In their framework, graph components are composed of vectors, with each vector consisting of respective values for growth, inflation, and leverage factors.

Hierarchical clustering creates a tree-like structure of clusters by merging smaller clusters into larger ones (agglomerative) or by splitting larger clusters into smaller ones (divisive).

Exhibit 1. Clustering Output (*k*-means)



Source: All synthetic data created by author.

Figure 2. Algorithm: Spectral Clustering

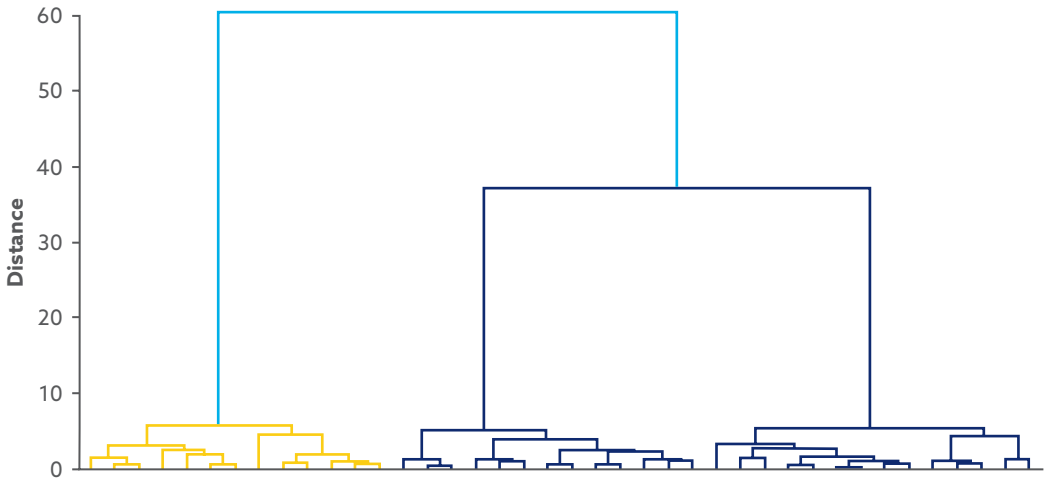
1. **Construct similarity matrix W:**
   - For each pair (i, j): $\mathbf{W}[i, j] = \exp[-||x_i - x_j||^2/(2\sigma^2)]$
2. **Compute degree matrix D:**
   - $\mathbf{D}[i, i] = \Sigma_j \mathbf{W}[i, j]$, $\mathbf{D}[i, j] = 0$ for $i \neq j$
3. **Compute normalized Laplacian:**
   - $L_{norm} = \mathbf{D}^{(-1/2)} \times (\mathbf{D} - \mathbf{W}) \times \mathbf{D}^{(-1/2)}$
4. **Find *k* smallest eigenvectors of *L*:** $v_1, v_2, \ldots, v_k$
5. **Form matrix V** = $[v_1, v_2, \ldots, v_k]$ ($n \times k$ matrix)
6. **Normalize rows of V to unit length**
7. **Apply *k*-means clustering to rows of V**
8. **Output:** Cluster assignments

Agglomerative clustering is more common and works "bottom up" by initially treating each data point as a separate cluster and then iteratively merging the closest pair of clusters until all points belong to a single cluster or a desired number of clusters is reached (see **Figure 3**).

Figure 3. Agglomerative Clustering Algorithm

1. **Initialize:** Each point as its own cluster $C = \{\{x_1\}, \{x_2\}, \ldots, \{x_n\}\}$
2. **Compute distance matrix D between all pairs of points**
3. **While $|C| > 1$:**
   - **a.** Find closest pair of clusters (C_i, C_j) using linkage method:
     - Single: $\min\{d(x, y) : x \in C_i, y \in C_j\}$
     - Complete: $\max\{d(x, y) : x \in C_i, y \in C_j\}$
     - Average: $\text{mean}\{d(x, y) : x \in C_i, y \in C_j\}$
   - **b.** Merge C_i and C_j into new cluster $C_k = C_i \cup C_j$
   - **c.** Update C by removing (C_i, C_j) and adding C_k
   - **d.** Update distance matrix **D** for new cluster C_k
   - **e.** Record merge in dendrogram
4. **Output:** Dendrogram structure (**Exhibit 2**)

Exhibit 2. Hierarchical Clustering Dendrogram



Note: Different colors represent different clusters at different distance cutoffs.

The algorithm requires a distance metric between data points as well as a linkage criterion to measure distances between clusters—for example, single linkage (minimum distance), complete linkage (maximum distance), average linkage (average distance), or Ward linkage (minimizes within-cluster variance).

A well-known investment application of hierarchical clustering is Hierarchical Risk Parity (HRP), introduced by López de Prado (2016). HRP uses hierarchical clustering to infer relationships between assets, which are then used directly for portfolio diversification, addressing three major concerns of quadratic optimizers: instability, concentration, and underperformance. The approach departs from classical mean–variance optimization by using a three-step process that organizes assets into hierarchical clusters based on their correlation structure, reorganizes the correlation matrix according to this tree structure, and then allocates capital recursively through the hierarchy using inverse variance weighting within each cluster.

Another prominent technique is DBSCAN (density-based spatial clustering of applications with noise), a density-based clustering algorithm that groups together points in high-density areas while marking points in low-density regions as noise or outliers, making it particularly effective for discovering clusters of arbitrary shapes and sizes (Ester, Kriegel, Sander, and Xu 1996). The algorithm requires two parameters: epsilon (ε), which defines the neighborhood radius, and minimum points (MinPts), required to form a dense region. Core points have at least MinPts neighbors within ε distance, border points within ε distance of core points, and noise points not meeting either criterion. In contrast to *k*-means, DBSCAN does not require advance specification of the number of clusters and can identify clusters with irregular shapes, making it robust against outliers and noise. In financial applications, DBSCAN excels at fraud detection by identifying unusual transaction patterns, market anomaly detection, and customer behavior analysis where normal clustering algorithms might fail because of the presence of outliers or nonspherical cluster shapes that are common in financial data distributions.

An extension of DBSCAN is OPTICS (ordering points to identify the clustering structure), which creates an ordering of data points that represents the density-based clustering structure, providing more detailed insights into cluster hierarchies and varying density regions within the dataset (Ankerst, Breunig, Kriegel, and Sander 1999). The algorithm computes core distances and reachability distances for each point, creating a reachability plot that visualizes the clustering structure across different density thresholds, allowing analysts to extract clusters at multiple scales without specifying parameters in advance. This hierarchical approach is particularly valuable when dealing with clusters of varying densities or nested clusters or when the optimal clustering parameters are unknown because it provides a comprehensive view of the data's density structure. In high-dimensional financial data analysis, OPTICS proves invaluable for identifying complex market structures, detecting multiscale patterns in trading data, and analyzing portfolio correlations where traditional clustering methods might miss important structural relationships because of varying density patterns for different market conditions or time periods.

Affinity propagation (AP), introduced by Frey and Dueck (2007), belongs to the family of graph-theoretic clustering techniques and is based on the concept of "message passing" (Mézard 2007) between the candidate members of a cluster that continues until each candidate is sufficiently informed to join the appropriate cluster. AP begins by measuring the similarity, $s(\mathbf{i}, \mathbf{k})$, between vectors, which represents the similarity of vector **k** to vector **i**. Similarity is measured by a metric chosen by the model builder (e.g., Euclidean distance).

The basic input to AP is a real-valued number $s(\mathbf{k}, \mathbf{k})$, called a *preference*, for each observation. Observations with larger preference values are more likely to be selected as cluster centers, also known as *exemplars*. However, cluster selection is a function of not only preference size but also the two message-passing operations that are the essence of the AP algorithm. The first—the *responsibility*, $r(\mathbf{i}, \mathbf{k})$—is a message transmitted from observation **i** to a candidate exemplar **k** that expresses the suitability of observation **k** as an exemplar for observation **i** given the suitability of other candidate exemplars. The second—the *availability*, $a(\mathbf{i}, \mathbf{k})$—works in the opposite direction and is sent from a candidate exemplar **k** to an observation **i**. Availability expresses how appropriate it would be for observation **i** to select observation **k** as its exemplar, given the existing support that observation **k** has from other observations to serve as an exemplar. The AP process begins by initializing the availabilities to zero, $a(\mathbf{i}, \mathbf{k}) = 0$, and then proceeds to compute the responsibilities using the following rule:

$$r(\mathbf{i},\mathbf{k}) = s(\mathbf{i},\mathbf{k}) - \max_{\mathbf{k}' \text{ s.t. } \mathbf{k}' \neq \mathbf{k}} \{a(\mathbf{i},\mathbf{k}') + s(\mathbf{i},\mathbf{k}')\}. \tag{1}$$

The following formula then determines whether an observation is a good exemplar:

$$a(\mathbf{i},\mathbf{k}) = \min\left\{0, r(\mathbf{k},\mathbf{k}) + \sum_{\mathbf{i}' \text{ s.t. } \mathbf{i}' \neq \{\mathbf{i},\mathbf{k}\}} \max\{\{0, r(\mathbf{i}',\mathbf{k})\}\}\right\} \tag{2}$$

The "self-availability" of an observation is expressed as follows:

$$a(\mathbf{i},\mathbf{k}) = \sum_{\mathbf{i}' \text{ s.t. } \mathbf{i}' \neq \mathbf{k}} \max\{\{0, r(\mathbf{i}',\mathbf{k})\}\} \tag{3}$$

An investment application of AP is presented by Simonian (2020), who used it to determine the level of diversity within a set of investment signals. To classify signals according to their statistical predictive properties, the author posited a vector consisting of information coefficient (IC) and IC variance values in various regime-specific subsamples as inputs into the clustering algorithm. Using AP in this manner allows us to gain a multidimensional view of investment signal diversity, with each measure providing information on a different aspect of predictive effectiveness.

Cluster Evaluation Techniques

Although many clustering algorithms require positing the number of clusters, techniques have been developed that allow the user to determine the most suitable clustering scheme. Two techniques in particular have become popular. The first is the *silhouette score*, an internal clustering evaluation metric that measures how similar each point is to points in its own cluster compared with points in other clusters, providing both individual point scores and an overall clustering quality measure. For each data point, the silhouette coefficient is calculated as $(b - a)/\max(a, b)$, where a is the mean distance to other points in the same cluster (intracluster distance) and b is the mean distance to points in the nearest neighboring cluster (intercluster distance). The silhouette coefficient ranges from −1 to 1. Values close to 1 indicate that the point is well matched to its cluster and poorly matched to neighboring clusters, values around 0 suggest that the point is on or very close to the decision boundary between clusters, and negative values indicate that the point might have been assigned to the wrong cluster.

The mathematical foundation of the silhouette score relies on distance-based cohesion and separation measures. For a point i in cluster C, the intracluster distance, $a(i)$, represents the

average distance between point i and all other points in the same cluster, measuring cluster cohesion. The intercluster distance, $b(i)$, is the minimum average distance from point i to points in any other cluster, measuring cluster separation. The silhouette coefficient, $s(i) = [b(i) - a(i)]/\max[a(i), b(i)]$, provides a normalized measure that balances cohesion and separation, with higher values indicating better clustering quality.

The second popular cluster evaluation technique, the *Adjusted Rand Index* (ARI), introduced by Hubert and Arabie (1985), builds on the measure introduced by Rand (1971) and is a more explicitly probabilistic measure of cluster uniqueness. Two important characteristics distinguish an ARI value from a silhouette score. The first is that a Rand Index value is relational. Whereas a silhouette score tells us how tight a particular clustering scheme is, an ARI value ranges from 1 to −1 and tells us how similar two clustering schemes are. A value of 0 represents two independent clusters, and a value of 1 represents identical clusters. Negative values indicate worse-than-random clustering. Accordingly, the second distinguishing characteristic of an ARI value is that lower values indicate more unique pairs of clustering schemes. The metric is particularly valuable because it adjusts for the expected similarity that would occur by chance alone, making it more reliable than the basic Rand Index when comparing clusterings with different numbers of clusters or when dealing with imbalanced cluster sizes.

The mathematical foundation of the original Rand Index begins with the contingency table, which cross-tabulates the cluster assignments from two different clusterings. Given two clusterings $U = \{U_1, U_2, \ldots, U_r\}$ and $V = \{V_1, V_2, \ldots, V_s\}$, the contingency table entry n_{ij} represents the number of objects that are in both cluster U_i and cluster V_j. The Rand Index is calculated by counting the number of pairs of objects that are either in the same cluster in both clusterings or in different clusters in both clusterings and dividing by the total number of pairs. This raw measure does not account for the expected agreement that would occur by random chance, however, which is where the adjustment becomes crucial. The mathematical formula for ARI can be expressed as $\text{ARI} = (\text{RI} - \text{Expected}_{RI})/[\max(\text{RI}) - \text{Expected}_{RI}]$, where RI is the Rand Index, Expected_{RI} is the expected value of the Rand Index under the null hypothesis of random clustering, and max(RI) is the maximum possible value of the Rand Index. This adjustment ensures that the expected value of ARI is zero when clusterings are independent, making it a more interpretable measure than the raw Rand Index.

Dimension Reduction Techniques

Finance is a data-driven enterprise. Indeed, the sheer size of data processed and the number of variables considered in financial applications may at times test the limits of mathematical models and information technology infrastructure. Given this fact, reducing the dimensions of a problem when possible is a critical aspect of any investment process.

Principal component analysis (PCA) is a dimension reduction technique that has been used in finance for many years (Pearson 1901; Hotelling 1933). PCA transforms high-dimensional data into a lower-dimensional space while preserving maximum variance. PCA works by finding the principal components, which are orthogonal directions in the feature space that capture the most variance in the data. The algorithm computes the covariance matrix of the data, performs eigendecomposition to find eigenvectors (principal components) and eigenvalues (variance explained), and then projects the original data onto the space spanned by the top k eigenvectors (see **Figure 4**). This linear transformation creates uncorrelated features ordered by the

Figure 4. PCA Algorithm

1. **Center the data:** $X_{centered} = X - \text{mean}(X)$
2. **Compute covariance matrix:** $\mathbf{C} = [1/(n-1)] \times X_{centered}^{T} \times X_{centered}$
3. **Perform eigendecomposition:** $\mathbf{C} = \mathbf{V} \times \Lambda \times \mathbf{V}^{T}$
   - $\mathbf{V}$: eigenvectors (principal components)
   - Λ: eigenvalues (variance explained)
4. **Sort eigenvectors by decreasing eigenvalues**
5. **Select top *k* eigenvectors:** $W = \mathbf{V}[:, 0{:}k]$
6. **Transform data:** $Y = X_{centered} \times W$
7. **Output:** Y, W, explained variance ratio

amount of variance they explain, making PCA particularly useful for data visualization, noise reduction, and feature extraction in preprocessing pipelines.

Perhaps the most well-known example of a financial application of PCA appears in the decomposition of the yield curve by Litterman and Scheinkman (1991). Their application of PCA involves constructing a matrix where each row represents a specific date and each column represents yields at different maturities (such as 3-month, 6-month, 1-year, 2-year, 5-year, 10-year, and 30-year rates). PCA then decomposes the yield curve data into orthogonal components ranked by their explanatory power. Their study (and others that followed) revealed that three principal components explain approximately 95%–99% of yield curve movements:

- *First principal component (level):* This component typically accounts for 80%–90% of the variance and represents parallel shifts in the yield curve. When this factor moves, all yields tend to move up or down together by similar amounts. This behavior reflects broad monetary policy changes, inflation expectations, or general economic conditions.
- *Second principal component (slope):* Explaining roughly 5%–15% of variance, this component captures the steepening or flattening of the yield curve. It represents the spread between long-term and short-term rates, often reflecting expectations about future monetary policy or economic growth.
- *Third principal component (curvature):* Accounting for 1%–5% of variance, this component captures changes in the curve's convexity or "bow" shape. It reflects relative movements in medium-term rates compared with short- and long-term rates, often related to market expectations about intermediate-term economic conditions.

Another popular dimension reduction technique is *t-distributed stochastic neighbor embedding* (*t*-SNE), a powerful dimension reduction technique that can be used to visualize high-dimensional data in a lower-dimensional space, typically 2D or 3D (van der Maaten and Hinton 2008). It is particularly effective for exploring complex datasets and identifying clusters

Figure 5. *t*-SNE Algorithm

1. **Compute high-dimensional similarities:**
   - For each point x_i, calculate conditional probabilities: $p_{\{j|i\}} = \exp(-||x_i - x_j||^2/2\sigma_i^2)/\Sigma k \neq i \exp(-||x_i - x_k||^2/2\sigma_i^2)$
   - Symmetrize: $p_{ij} = (p_{\{j|i\}} + p_{\{i|j\}})/2n$
2. **Initialize low-dimensional embedding:**
   - Randomly place points y_i in target dimensional space
3. **Iterative optimization: For each iteration, compute low-dimensional similarities:**
   - Use *t*-distribution: $q_{ij} = (1 + ||y_i - y_j||^2)^{-1}/\Sigma k \neq l\ (1 + ||y_k - y_l||^2)^{-1}$

   Calculate gradient:
   - Minimize Kullback–Leibler (KL) divergence: $KL(P||Q) = \Sigma_{ij}\ p_{ij} \log(p_{ij}/q_{ij})$
   - Gradient: $\partial C/\partial y_i = 4\Sigma_j\ (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}$

   Update positions:
   - Apply gradient descent with momentum to move points y_i
4. **Output:** Final low-dimensional embedding Y

or patterns that might not be apparent in raw data. In finance, *t*-SNE can be applied to analyze and visualize market segmentation, such as grouping stocks or assets based on their historical performance, risk profiles, or other features. The algorithm for *t*-SNE appears in **Figure 5**.

To provide a supervised counterpoint, we mention *linear discriminant analysis* (LDA), a technique that finds linear combinations of features that best separate different classes, making it particularly useful for classification preprocessing (Fisher 1936). LDA can serve as a powerful classification tool in financial applications, particularly for identifying trading signals. In credit risk modeling, LDA can help separate borrowers into distinct risk categories using financial ratios and other predictive variables, optimizing the linear combination of features that best discriminates between default and nondefault cases. For algorithmic trading, LDA can be used to classify market conditions into bullish, bearish, or neutral regimes based on technical indicators and market microstructure variables.

Another technique, *independent component analysis* (ICA), assumes that data are generated by mixing independent source signals and attempts to recover these original independent components, making it valuable for blind source separation problems such as audio signal processing (Comon 1994; Bell and Sejnowski 1995). ICA is therefore useful in financial applications that require blind source separation, particularly in identifying independent market factors from mixed signals. For example, in multiasset portfolio analysis, ICA separates returns into independent components that may represent different economic factors (inflation, growth, sentiment) that are not directly observable but drive asset performance. For high-frequency trading,

ICA helps isolate genuine price signals from market noise by identifying independent sources of price movement. The technique is particularly valuable in emerging markets, where traditional factor models may not apply, because ICA can discover country-specific or sector-specific independent factors that influence asset returns without requiring prior assumptions about factor structure.

Deep Learning Approaches

Autoencoders are neural networks designed to learn efficient data representations in an unsupervised manner by training the network to reconstruct its input data (Hinton and Salakhutdinov 2006). The architecture consists of an encoder that compresses the input into a lower-dimensional latent representation and a decoder that reconstructs the original input from this compressed representation. The network is trained to minimize reconstruction error, which forces it to learn meaningful features that capture the most important aspects of the data. Applications of encoders include dimensionality reduction, denoising, feature learning, and data compression. The algorithm for autoencoders is shown in **Figure 6**.

Variations of standard encoders include denoising autoencoders that learn to reconstruct clean data from corrupted inputs, variational autoencoders (VAEs) that learn probabilistic latent

Figure 6. Autoencoder Algorithm

1. **Initialize networks:**
   Encoder: $f_{enc}(x; \theta_{enc}) \rightarrow z$ with parameters θ_{enc}
   Decoder: $f_{dec}(z; \theta_{dec}) \rightarrow \hat{x}$ with parameters θ_{dec}
2. **Training loop (for each epoch):**
   For each mini-batch:
   Forward pass:
   Encode: $z = f_{enc}(x; \theta_{enc})$—compress input to latent code
   Decode: $\hat{x} = f_{dec}(z; \theta_{dec})$—reconstruct from latent code
3. **Loss computation:**
   Reconstruction loss: $L = ||x - \hat{x}||^2$ (mean squared error)
4. **Backward pass:**
   Compute gradients with respect to both networks: $\nabla\theta_{enc}L$, $\nabla\theta_{dec}L$
   Update encoder: $\theta_{enc} \leftarrow \theta_{enc} - \alpha\nabla\theta_{enc}L$
   Update decoder: $\theta_{dec} \leftarrow \theta_{dec} - \alpha\nabla\theta_{dec}L$
5. **Output:** Trained encoder and decoder networks

representations, and sparse autoencoders that enforce sparsity constraints on the hidden layer activations to learn more interpretable features (Kingma and Welling 2014). Unlike traditional autoencoders, which compress data into a deterministic latent space, VAEs model the latent space as a probability distribution (typically Gaussian). This probabilistic approach allows VAEs to generate realistic synthetic data by sampling from the learned latent distribution. VAEs consist of two main components:

- *Encoder:* Maps input data to a latent space by learning the parameters (mean and variance) of a probability distribution
- *Decoder:* Reconstructs the original data from samples drawn from the latent distribution

The key innovation of VAEs is the use of variational inference, where the model learns a latent distribution that captures the underlying structure of the data. The training process involves minimizing a loss function that combines reconstruction error (how well the decoder reconstructs the input) and a regularization term (how close the learned latent distribution is to a prior, typically a standard normal distribution). Examples of how VAEs are used in finance include synthetic data generation for testing trading strategies or stress-testing models, uncovering latent factors driving asset prices or market behavior, and identifying unusual patterns in financial data by comparing reconstructed data with the original input.

Another algorithm used to generate synthetic data is known as a *generative adversarial network* (GAN; Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, and Bengio 2014). GANs are a class of generative models that use a game-theoretic framework to learn and generate new data that mimic the distribution of a given dataset. GANs consist of two neural networks:

- *Generator:* Creates synthetic data from random noise, attempting to mimic the real data distribution
- *Discriminator:* Distinguishes between real data (from the dataset) and fake data (produced by the generator)

The generator and discriminator are trained simultaneously in a zero-sum game: The generator attempts to "fool" the discriminator by producing realistic data, and the discriminator tries to correctly identify real versus fake data. Over time, the generator improves its ability to create realistic data, and the discriminator becomes better at distinguishing real from fake. When the GAN converges, the generator produces data that are indistinguishable from the real data. GANs are widely used in finance for tasks that involve generating realistic synthetic data, modeling complex distributions, and simulating market scenarios. Simonian (2024) describes how the synthetic data generated by GANs can be used in the model validation process.

Anomaly Detection

Anomaly detection is an important part of finance because extreme outliers, such as stock market crashes, can have outsized financial and economic implications. However, anomaly detection is important not only for investors but also in such areas as credit card fraud detection, identifying unusual trading patterns, detecting market manipulation, and monitoring portfolio performance for abnormal behavior patterns that could indicate operational risks or model failures. Although other types of machine learning models, such as DBSCAN and support vector

Figure 7. LOF Algorithm

1. **Find *k*-nearest neighbors for each point**
2. **Calculate reachability distance:**
   $r_{dist}(A, B) = \max[k\text{-distance}(B), d(A, B)]$
3. **Compute local reachability density:**
   $Local_{rd}(A) = 1/(\Sigma Reach_{dist}(A, B)/|N_k(A)|)$
4. **Output:** $\text{LOF}(A) = [\Sigma I_{rd}(B)/I_{rd}(A)]/|N_k(A)|$ for B in $N_k(A)$

machines, can perform anomaly detection, dedicated algorithms have also been developed for outlier detection. *Isolation Forest* uses decision trees to isolate anomalies by randomly selecting features and splitting the data according to threshold values, operating under the principle that anomalies are few and different—hence, easier to isolate (Liu, Ting, and Zhou 2008). The algorithm constructs an ensemble of isolation trees by randomly selecting features and split values, with anomalies requiring fewer splits to be isolated and thus having shorter average path lengths in the trees, whereas normal points require more splits and have longer paths. This approach is particularly effective because it directly targets anomalies rather than profiling normal instances, making it computationally efficient with linear time complexity and effective in high-dimensional spaces.

Another popular anomaly detection technique is the *local outlier factor* (LOF), which detects anomalies by comparing how densely packed each point is relative to its local neighborhood (Breunig, Kriegel, Ng, and Sander 2000). The key insight is that outliers exist in sparser regions compared with their neighbors. As a rough analogy, think of data points as houses in a city: In dense neighborhoods, houses are close together, and in less populated areas, houses are spread out. An outlier is like a house that is unusually isolated compared with the density of its surrounding neighborhood. The LOF algorithm is shown in **Figure 7**.

Conclusion

In this chapter, I have provided an overview of some major unsupervised learning algorithms along with examples of how they are applied in various areas of finance. I have shown that unsupervised learning plays a major role in classification, anomaly detection, and synthetic data generation—all major application areas in finance. The next chapter will delve deeper into an area of unsupervised learning—network theory—that has become popular in recent years because of its flexibility and power to illuminate various types of connections that exist between financial and economic entities and actors.

References

Ankerst, Mihael, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. "OPTICS: Ordering Points to Identify the Clustering Structure." *SIGMOD Record* 28 (2): 49–60. doi:10.1145/304181.304187.

Bell, Anthony J., and Terrence J. Sejnowski. 1995. "An Information-Maximization Approach to Blind Separation and Blind Deconvolution." *Neural Computation* 7 (6): 1129–59. doi:10.1162/neco.1995.7.6.1129.

Breunig, Markus M., Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. "LOF: Identifying Density-Based Local Outliers." *Proceedings of the 2000 ACM SIGMOD Record*: 93–104. doi:10.1145/342009.335388.

Comon, Pierre. 1994. "Independent Component Analysis, a New Concept?" *Signal Processing* 36 (3): 287–314. doi:10.1016/0165-1684(94)90029-9.

Donath, W. E., and A. J. Hoffman. 1973. "Lower Bounds for the Partitioning of Graphs." *IBM Journal of Research and Development* 17 (5): 420–25.

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." *KDD'96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*: 226–31.

Fiedler, M. 1973. "Algebraic Connectivity of Graphs." *Czechoslovak Mathematical Journal* 23 (98): 298–305.

Fisher, Ronald A. 1936. "The Use of Multiple Measurements in Taxonomic Problems." *Annals of Eugenics* 7 (2): 179–88. doi:10.1111/j.1469-1809.1936.tb02137.x.

Frey, Brendan J., and Delbert Dueck. 2007. "Clustering by Passing Messages Between Data Points." *Science* 315 (5814): 972–76. doi:10.1126/science.1136800.

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. "Generative Adversarial Nets." *Proceedings of the 27th International Conference on Neural Information Processing Systems* 2: 2672–80.

Hall, K. M. 1970. "An *r*-Dimensional Quadratic Placement Algorithm." *Management Science* 17 (3): 219–229.

Hinton, Geoffrey E., and Ruslan Salakhutdinov. 2006. "Reducing the Dimensionality of Data with Neural Networks." *Science* 313 (5786): 504–07. doi:10.1126/science.1127647.

Hotelling, Harold. 1933. "Analysis of a Complex of Statistical Variables into Principal Components." *Journal of Educational Psychology* 24 (6): 417–41. doi:10.1037/h0071325.

Hubert, Lawrence, and Phipps Arabie. 1985. "Comparing Partitions." *Journal of Classification* 2 (1): 193–218. doi:10.1007/BF01908075.

Kingma, D. P., and M. Welling. 2014. "Auto-Encoding Variational Bayes." arXiv: 1312.6114.

Litterman, Robert B., and Josè Scheinkman. 1991. "Common Factors Affecting Bond Returns." *Journal of Fixed Income* 1 (1): 54–61. doi:10.3905/jfi.1991.692347.

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. 2008. "Isolation Forest." *Proceedings of the 8th IEEE International Conference on Data Mining*: 413–22. doi:10.1109/ICDM.2008.17.

Lloyd, S. P. 1982. "Least Squares Quantization in PCM." *IEEE Transactions on Information Theory* 28 (2): 129–137.

López de Prado, Marcos. 2016. "Building Diversified Portfolios That Outperform Out of Sample." *Journal of Portfolio Management* 42 (4): 59–69. doi:10.3905/jpm.2016.42.4.059.

Mézard, Marc. 2007. "Where Are the Exemplars?" *Science* 315 (5814): 949–51. doi:10.1126/science.1139678.

Pearson, Karl. 1901. "On Lines and Planes of Closest Fit to Systems of Points in Space." *London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2 (11): 559–72. doi:10.1080/14786440109462720.

Rand, William M. 1971. "Objective Criteria for the Evaluation of Clustering Methods." *Journal of the American Statistical Association* 66 (336): 846–50. doi:10.1080/01621459.1971.10482356.

Simonian, Joseph. 2020. "Modular Machine Learning for Model Validation: An Application to the Fundamental Law of Active Management." *Journal of Financial Data Science* 2 (2): 41–50. doi:10.3905/jfds.2020.1.027.

Simonian, Joseph. 2024. *Investment Model Validation: A Guide for Practitioners*. Charlottesville, VA: CFA Institute Research Foundation. doi:10.56227/24.1.15.

Simonian, Joseph, and Chenwei Wu. 2019. "Minsky vs. Machine: New Foundations for Quant-Macro Investing." *Journal of Financial Data Science* 1 (2): 94–110. doi:10.3905/jfds.2019.1.004.

Spielman, D.A. and S.-H. Teng. 2007. "Spectral Partitioning Works: Planar Graphs and Finite Element Meshes." *Linear Algebra and its Applications* 421 (2): 284–305.

van der Maaten, Laurens, and Geoffrey Hinton. 2008. "Visualizing Data Using t-SNE." *Journal of Machine Learning Research* 9 (86): 2579–605.

Wu, Dingming, Xialong Wang, and Shaocong Wu. 2022. "Construction of Stock Portfolios Based on *k*-Means Clustering of Continuous Trend Features." *Knowledge-Based Systems* 252 (27 September). doi:10.1016/j.knosys.2022.109358.

UNSUPERVISED LEARNING II: NETWORK THEORY

Gueorgui S. Konstantinov, PhD
Senior Portfolio Manager, Fixed Income and Currencies, DekaBank

Agathe Sadeghi, PhD
Stevens Institute of Technology, School of Business

Introduction

Decision making and risk assessment have traditionally been grounded in core financial models. These models have encountered challenges as global interdependencies have grown in complexity as financial markets evolve. Many classical models approach the market as either a collective system with uniform interactions or groups of unconnected systems where actors make autonomous decisions. These overly simplified approaches have proved incorrect in many incidents, such as the financial crisis of 2007–2008 and the COVID-19 pandemic. The failure of Lehman Brothers and the near-collapse of AIG highlighted how distress is capable of rapidly spreading from one region of the system to another, revealing hidden links and vulnerabilities. Conventional models did not foresee these vulnerabilities. These events revealed the sobering truth that the interconnectivity of financial institutions and the system is much more intricate than the models of the time could capture and that the interexposures between institutions are crucial to the systemic stability of the structure. This crisis proves that models are blind to emerging risks if relationships between players add up in unpredictable asymmetric ways. To respond to the challenge of managing systemic risk, one needs tools that identify and structure relationships to analyze connections, no matter how unconventional the direction might be.[1]

A powerful and intuitive framework for overcoming these constraints is provided by network theory, a discipline with roots in graph theory. It offers mathematical language for investigating systems made up of distinct entities (represented as nodes or vertices) and the connections or interactions among them (represented as links or edges). This method has shown promise in a wide range of fields, including biology, computer science, transportation, and the social sciences.

Network analysis is a natural fit for financial systems. Markets, assets, financial institutions, and even information flows are all interconnected. These relationships create intricate networks that support the dynamics of the market. Practitioners can examine the true structure of market interactions and go beyond oversimplified assumptions by depicting these varied financial relationships as networks. The following are examples of networks:

- Interbank lending networks: A vital network for distributing liquidity is created when banks lend to and borrow from one another.

[1] Fundamental literature on networks includes Wasserman and Faust (1994), Newman (2010), Barabási (2016), and Borgatti, Everett, Johnson, and Agneessens (2022). Financial networks are extensively analyzed by Diebold and Yilmaz (2015).

- Asset and factor networks: By comparing price movements of stocks, bonds, commodities, currencies, or factors, one can identify community structures, relationships, associations, and interactions that improve diversification.
- Ownership networks: Ownership links are created by businesses that own stock in other businesses.
- Derivative networks: Counterparty exposures in derivative contracts, such as credit default swaps (CDSs), create a web of contingent liabilities.
- Bank–firm networks: The financial sector is connected to the actual economy through lending relationships between banks and nonfinancial firms.
- Supply chain networks: Along supply chains, there are dependencies and financial flows.

By demonstrating how information spreads, how sentiment changes, and how various market segments affect one another, network analysis provides a potent lens through which to view market dynamics. It is a crucial tool for evaluating systemic risk, a viewpoint that central banks and regulators have now widely embraced because it enables modeling of contagion pathways and identification of systemically important institutions. It also supports diversification through cluster detection, feature selection, importance score determination, and unveiling hidden relationships between assets. Network models also provide insight into how news and analyst sentiment affect asset prices by tracking the flow of information through markets.

This chapter introduces key network theory concepts and their practical use in finance. It covers community detection to find hidden asset groups, centrality measures to identify influential actors, and fundamental structures such as nodes and edges. It also investigates network dynamics for modeling contagion and systemic risk. Formal definitions are paired with actual financial examples to discuss applications in investment management, such as portfolio construction, market prediction, and pattern recognition.

Concepts

Knowing the fundamentals of graph theory is necessary to comprehend financial systems from a network perspective. These elements offer the framework for modeling complex dependencies.[2]

Nodes

Nodes or vertices are the basic components or entities that make up a network. They stand in for the actors or objects in the system that is being modeled. The definition and selection of a node are important decisions that depend solely on the particular financial system under study. Numerous entities can be represented by nodes:

- Institutions, such as central banks, investment banks, mutual funds, and hedge funds
- Corporations, such as nonfinancial businesses that participate in supply chains or credit relationships
- Assets and factors, such as individual stocks, bonds, currencies, commodities, or derivatives, such as CDSs

[2] A detailed explanation of these metrics can be found in Newman (2010) and Barabási (2016).

- People, such as market traders, stock analysts, board members, firm employees, and even politicians in policy networks
- Geographic/political entities, such as nations, jurisdictions, communities, or areas involved in global networks of capital flows or trade

Financial network nodes are more than just abstract points; they also carry characteristics that give analysis crucial context. These attributes, such as the total assets of a bank, the sector or volatility of a stock, the credit rating of a company, or the GDP of a nation, aid in differentiating nodes and influence how network relationships are interpreted. A key first step in creating meaningful financial network models is defining nodes and their relevant attributes because this process establishes what relationships can be captured and what insights can be obtained.

Edges

The connections, relationships, interactions, or dependencies between pairs of nodes in a network are represented by edges, also referred to as links or arcs. They represent the connections between the nodes. In finance, they capture the way financial entities interact with one another. The following are examples:

- Lending and borrowing: In an interbank network, an edge can stand in for a loan from Bank A to Bank B or a bank credit line to a business.
- Association: Two stocks whose returns show a high degree of association can be connected by an edge. Association metrics, such as pairwise correlations between asset classes, are the most widely used tool to measure association and to draw edges in financial networks.
- Counterparty: In a derivative contract (such as a CDS), an edge can stand in for the possible loss exposure between two parties.
- Informational: Two stocks may be linked if they are covered by the same financial analyst.
- Ownership: An edge may indicate that Firm A owns a sizable portion of Firm B's stock.
- Affiliation: Edges can connect businesses that share board members.

Edges in financial networks can have attributes that are essential for network analysis. The difference between directed and undirected edges is a crucial one. A relationship with a distinct origin and destination is indicated by directed edges, such as when Bank A lends to Bank B. The relationship's direction is vital in these situations, particularly when simulating influence or contagion. Undirected edges, in contrast, show symmetrical or reciprocal relationships in which order is irrelevant, such as the correlation between two stocks.

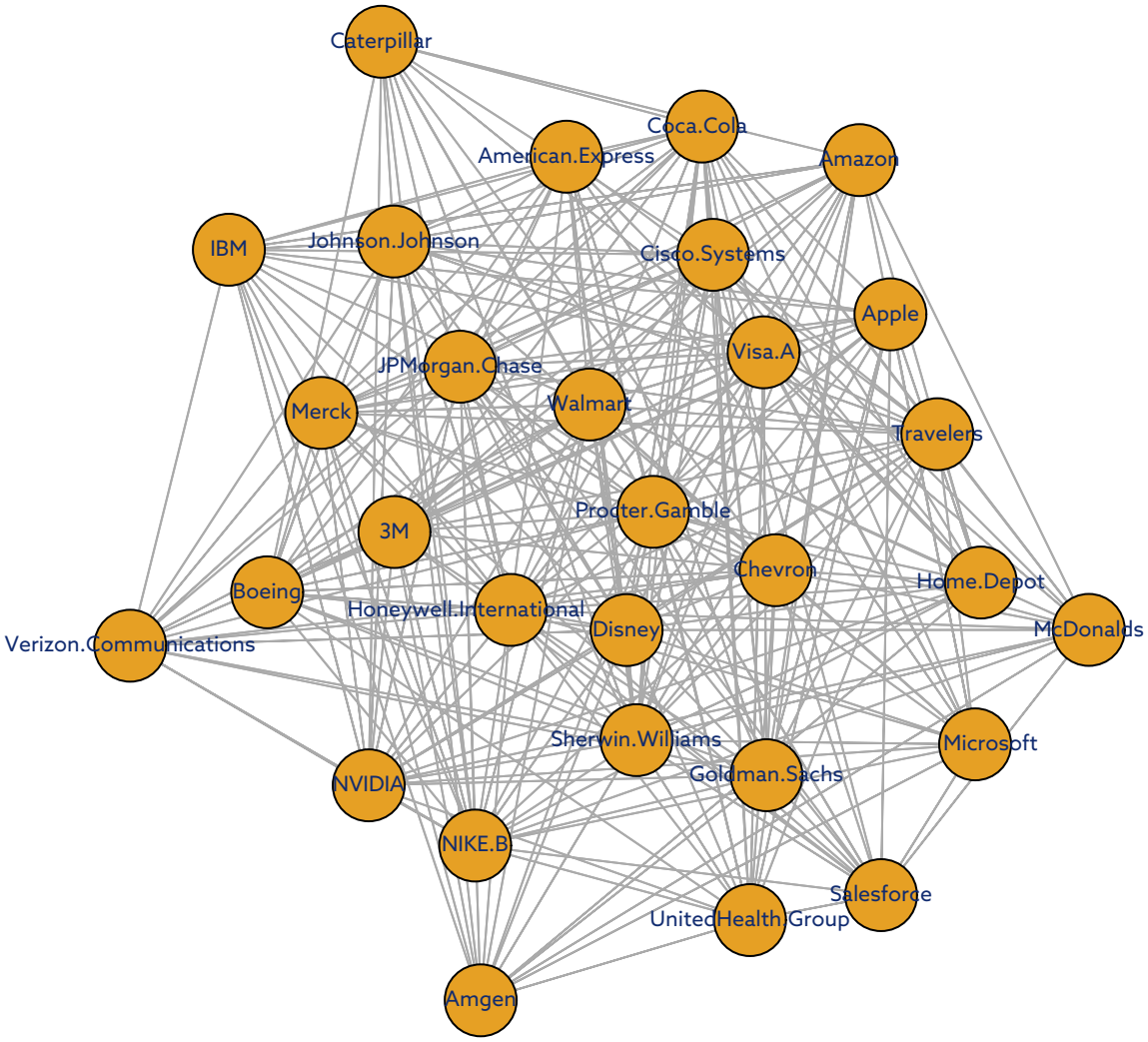
Additionally, edges can be weighted or unweighted. Weighted edges, such as trading volume, correlation coefficients, or loan size, show how strong a relationship is. Ignoring these weights can result in a substantial loss of information because they provide important detail. Conversely, unweighted edges do not quantify the strength of a relationship; they show only whether one exists. They are helpful in situations where a connection's existence is more significant than its size. The adjacency matrix of a network is the one that carries the information about the pairwise weight structure.[3]

[3] See the next section for detailed information about the adjacency matrix.

To create precise and instructive financial network models, edges must be carefully defined, including their directionality and weighting. The outcomes of network analysis, including contagion simulations and centrality calculations, are directly affected by these characteristics.

Exhibit 1 provides a simple example of the correlation-based network of the Dow Jones Industrial Average Index. The network consists of 30 nodes (stocks in the index), and according to the simple algorithm, there are 570 links between the nodes. Given the 30 nodes in the networks, the total number of possible links between all 30 nodes is 30 × 29 = 870. In other words, the algorithm's output of 570 links between the nodes shows that not all nodes share a pairwise link between them. For example, American Express might not have a link to Amazon. Similarly, the edge between, say, Boeing and Visa might not exist.

Exhibit 1. A Simple Correlation-Based Network of the Dow Jones Industrial Average Index



Source: Bloomberg, LLC.

Path, Trails, Walks, Geodesic Distances, and Network Structure

Higher-level insights can be gained from the general arrangement and connectivity patterns of a network, which go beyond individual nodes and edges. A crucial distinction lies in the definitions of paths, trails, walks, and geodesic distances.

A path is a route in a network consisting of a sequence of nodes connected by edges. The terms *geodesic*, *trail*, *path*, and *walk* describe different types of node–edge sequences in a network. They differ in their constraints and the connectivity characteristics they represent.

A geodesic is the shortest path between two nodes in a network. It represents the most direct route, taking into account the network's structure and any weights assigned to its edges (if applicable). Geodesic distances are used to calculate various network metrics, such as average path length, network diameter, and centrality measures. The length of a path is determined by the number of edges it traverses; a path with a length of 1 consists of a single edge. For centrality metrics, such as closeness and betweenness, the shortest path between nodes is critical because it minimizes either total weight (in weighted graphs) or the number of edges (in unweighted graphs).

Paths are useful in finance for modeling complex transaction sequences, analyzing the propagation of shocks through counterparties, and tracking the flow of information from analysts to investors. Konstantinov, Aldridge, and Kazemi (2023) extensively discussed the flow properties of financial networks. A walk is a sequence of nodes and edges in which each edge is adjacent to the previous one and both nodes and edges can be revisited. Walks can include loops and repeated edges. This property is typical in financial networks. A trail is similar to a walk, but no edge may be repeated, although nodes can still be revisited. For instance, a stock can be bought multiple times from the same brokerage house, but the exact transaction (edge) cannot be repeated. A path further restricts the sequence such that no node or edge is repeated. This type of sequence can be quite limiting in financial networks.

According to Newman (2010), some directed networks have a special structure called cycles. A cycle is a walk in which no node is repeated (except for the starting/ending node), forming a closed loop where all edges point in the same direction. Networks without such loops are referred to as *acyclic directed networks*, which are widely used in causal inference and prediction.

The primary differences between these sequences are the rules about revisiting nodes and edges. When analyzing flow processes in networks, each type of sequence can represent different aspects of connectivity, traversal, or the spread of information. For example, geodesics can indicate the most efficient information flow between nodes. Thus, applying geodesic metrics to financial networks provides a generalizable framework for modeling relationships. The simplest way to model such relationships is through pairwise correlation metrics.

An important consideration in flow processes is whether money, information, or services reach a node only once, simultaneously, or multiple times. Financial flows often arrive at nodes repeatedly over time. For example, a portfolio's exposure to certain assets may change frequently depending on the economic environment. In this context, *systematic* financial risk can become *systemic*, simultaneously impacting many nodes via multiple links, not necessarily along a single geodesic path.

Similarly, information can reach multiple points at once, spreading by replication rather than linear transfer and influencing many nodes simultaneously.

In contrast, goods and packages are often transferred along geodesic paths—a model that may not always be appropriate for finance, where assets, factors, or institutions may be disconnected. Consider a hedge fund using a sophisticated quantitative strategy with minimal market exposure. In a hedge fund network, this fund might not be linked to others based on correlation coefficients, meaning no directed path exists. Directionality in networks implies causality, a key concept in financial markets, although it is often difficult to detect. The direction of money, information, and technological impacts (e.g., high-frequency trading) reflects this causality. The processes of paths, trails, and walks help define how relationships form and evolve between nodes.

The underlying flow process—whether involving information, rumors, money, goods, services, orders, transactions, or financial contagion—is critical for network modeling. A major distinction lies in whether flows follow geodesics, trails, paths, or walks. In financial markets, money typically follows walks rather than trails, which is a defining characteristic of financial flow and essential for network analysis.

Based on mathematical graph theory, a network (or graph) is a collection of nodes (vertices) connected by edges (links). A network G is defined as a set of nodes n and edges m, formally expressed as $G = (n, m)$.

There are two primary ways to construct a financial network:

- by modeling the data at the edge level, keeping the nodes fixed, or
- by modeling the network at the node level, with edges defined between them.

Mathematically, relationships between nodes are often represented using a matrix. Financial networks can be modeled by an adjacency matrix, where rows and columns represent nodes and each cell indicates the presence or strength of a connection.

In unweighted networks, entries in the adjacency matrix are binary (0 or 1), whereas in weighted networks, these entries are real numbers reflecting the strength or weight of the connection. Generally, in an unweighted network, the adjacency matrix has cell values $[i, j]$ equal to 1 if there is a link from node i to node j and 0 otherwise. Using this notation, the result is a simple network whose main diagonal reflects self-edges. An example of self-edge (main diagonal of nonzeros) indicates that the node has a self-connectedness. Mathematically for a graph, $G = (n, m)$, where n is the number of nodes and m is the number of edges, the $n \times n$ adjacency matrix is denoted by $[A]_{ij}$ and is estimated using following rule:

$$a_{ij} = \begin{cases} 1 \text{ if node } i \text{ and } j \text{ are connected, or } \{i, j\} \in m \\ 0 \text{ otherwise} \end{cases}$$

The adjacency matrix for an undirected network consisting of 30 nodes for the Dow Jones Industrial Index is shown in **Exhibit 2**.

Konstantinov and Fabozzi (2025) provided an extensive review of the useful and practical ways to construct financial networks. The approaches used to estimate the links between the assets include correlation coefficients, regression models, econometric models, probabilistic models that derive probabilities, evaluation of transaction and money flow volumes, assets under management, and all possible information that applies to exchange in financial markets.

Exhibit 2. Adjacency Matrix

Goldman.Sachs	0 0 0 0 0 0 1 1 1 0 1 1 1 0 1 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1
UnitedHealth.Group	0 0 0 0 0 1 1 1 1 0 1 1 1 0 1 0 1 1 1 0 1 1 0 1 0 0 1 1 1 0
Microsoft	0 0 0 0 0 1 1 1 1 0 1 1 1 0 1 0 1 0 1 0 1 1 0 1 0 1 1 1 1 0
Home.Depot	0 0 0 0 0 1 0 1 1 0 1 1 1 0 1 0 1 0 1 0 1 1 0 1 0 1 1 1 1 0
Caterpillar	0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0
Sherwin.Williams	0 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1
Salesforce	1 1 1 0 0 1 0 0 0 1 0 1 0 1 1 0 0 0 1 1 1 0 1 0 1 1 1 0 1 0
Visa.A	1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 0 1 0 0
American.Express	1 1 1 1 0 1 0 0 0 0 0 0 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 0 1 0 0
McDonalds	0 0 0 0 0 1 1 1 0 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 1 1 1 0
Amgen	1 1 1 1 0 1 0 0 0 1 0 0 0 1 1 1 0 0 1 1 0 0 1 1 1 1 1 0 0 0
Apple	1 1 1 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 0 0 1 1 1 1 1 0 1 0 0
Travelers	1 1 1 1 0 1 0 1 1 1 0 0 0 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 1 1 0
JPMorgan.Chase	0 0 0 0 0 1 1 1 1 0 1 1 1 0 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1
Honeywell.International	1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1
IBM	0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1
Amazon	1 1 1 1 1 1 0 0 0 1 0 0 0 1 1 1 0 0 1 1 1 0 1 1 1 1 1 0 0 0 0
Boeing	1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 0 0 0 1 1 1 0 1 1 1 1 1 0 1 0
Procter.Gamble	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 0 1 1 1 1
Johnson.Johnson	0 0 0 0 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0 1 1 0 1 1 0 1 1 1 1
Chevron	1 1 1 1 1 1 1 0 0 1 0 0 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 0 1
NVIDIA	1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 0 0 1
3M	0 0 0 0 0 0 1 1 1 0 1 1 1 0 1 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1
Disney	1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1
Merck	0 0 0 0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1
Walmart	0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 1
NIKE.B	1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 0 1
Coca.Cola	1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 1 1 1 0 1 1 1 1 1 0 0 1 1
Cisco.Systems	1 1 1 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 0 1 0 0
Verizon.Communications	1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 0

Some structural features are frequently seen in real-world financial networks:

- A core–periphery structure displays a densely connected core (mainly large players) surrounded by a sparsely connected periphery (relatively smaller players). For instance, large international banks with extensive cross-border ties typically form the core of the global banking system, with national or regional banks that have fewer direct international ties encircling the core.
- Scale-free networks have a power-law degree distribution, which means that although most nodes have few connections, a select few (hubs) have a lot of connections. These hubs frequently have a significant impact on the stability and operation of networks. A few big, highly linked banks serve as hubs in the interbank lending market, which frequently has a scale-free structure. Such a hub's failure may have systemic repercussions that are disproportionately severe.
- Small-world networks are characterized by short average path lengths between any two nodes and high clustering. Fast transmission throughout the network is made possible by this structure. Financial markets might frequently act like small-world networks, enabling information, sentiment, or shocks to spread remarkably quickly.

Understanding these structural patterns aids in understanding the general behavior, resilience, and shock susceptibility of the network. The particular topology has a big impact on how information moves, how fast contagion spreads, and where systemic risks could be concentrated.

How to evaluate networks is an important question, and a graph theoretic toolkit allows us to measure the specific properties of a network. A classical financial network has specific properties, and the average degree, density, centrality scores, reciprocity, and clustering coefficient are the most important. The average degree k_i of a node i and k_j of node j at time t is the most basic structural property representation of the connections of a node in a network:

$$k_{i,t} = k_{j,t} = \sum_i a_{ij,t} = \sum_j a_{ji,t} = \frac{m}{n}.$$

The *density*, or *completeness*, of a network, ς, is given by the ratio between the numbers of current links relative to the number of all possible links between the nodes in the network:

$$\varsigma_t = \frac{m_t}{n_t(n_t - 1)}.$$

Here, m represents the number of links between nodes in the network and n represents the number of nodes. The density values range from 0 (no ties are present) to 1 (a completed network). A density value of 1 means all possible links are used.

The relation between the pairs of nodes in a directed network (a network with direction or flow, indicated by arrows) at time t is given by the value of *reciprocity*, ϱ_t. The reciprocity measures whether a node i is linked to a node j and if a node j is also linked to a node i. The values range between 0 and 1 (with 1 for fully reciprocal network). Given the entries in the adjacency matrices, the reciprocity is given by

$$\varrho_t = \frac{\sum_{ij} a_{ij,t}\, a_{ji,t}}{m_t}.$$

A node pair (i, j) is called *reciprocal* if ties exist between both nodes in both directions.

Centrality Measures

After the network is defined, finding the most important nodes is a crucial task. Although influence can take many different forms, centrality measures assign scores based on a node's position and connections. Selecting the appropriate measure depends on the information flow process because different measures quantify different aspects of the nodes. We discuss the most important measures of network properties.

Degree Centrality

The simplest measure, degree centrality, counts the number of direct connections a node has. It is denoted as deg(v), where v is the node. A high degree measure denotes a popular or active entity—for instance, a bank with many trading relationships or a frequently traded stock. To compare networks of varying sizes, a normalized version (by number of nodes) is frequently used.

If the network is directed, the number of incoming edges is called *in-degree*, and the number of outgoing edges is referred to as *out-degree*. A node with a high in-degree is one that receives a lot of connections or attention, such as a bank that borrows a lot and is therefore at risk of lender distress or a stock with a lot of buy recommendations. In contrast, a node with a high out-degree is a source of influence or risk, such as a bank lending to numerous people, which could spread distress if it fails.

Degree is often used as a measure of a node's connectedness in networks. In real-world networks, the average degree typically exceeds 1, indicating that a node is connected to more than one other node. A higher average degree or greater overall connectedness increases the likelihood of forming communities within the network.

One limitation of degree centrality is its exclusive focus on direct, immediate connections. This approach does not account for a node's position within the broader network or its indirect influence via the connections of its neighbors. A node may have a high degree but be linked only to minor, peripheral nodes, limiting its actual influence. Furthermore, standard degree centrality ignores the weights of the edges, although this shortcoming can be addressed by defining a weighted degree centrality.

As noted by Konstantinov and Fabozzi (2025), the key advantage of degree centrality—and metrics that assess the degree of all nodes in a network—is its ability to measure the immediate impact of risk at the node level. In contrast, eigenvector centrality captures both direct and indirect long-term influence across the network.

Importantly, many mathematical models in network science place a central focus on degree distribution, which plays a foundational role in understanding the structure and dynamics of complex networks.

Betweenness Centrality

The extent to which a node is situated on the shortest routes between any other pair of nodes in the network is measured by betweenness centrality. Developed by Linton C. Freeman, it gauges a node's function as a bridge or middleman. The fraction of shortest paths between each pair of nodes (u,v) that pass through node s is added up using the following formula:

$$C(b) = \sum_{u \neq v \neq s} \frac{\sigma_{u,v}(s)}{\sigma_{u,v}},$$

where $\sigma_{u,v}(s)$ is the number of shortest paths that go through s and $\sigma_{u,v}$ is the total number of shortest paths between u and v.

The application of betweenness centrality in financial networks depends on the underlying flow processes within the network. Unlike such metrics as closeness centrality, which assume random path selection, betweenness centrality is particularly appropriate for networks in which the flow of resources or information follows specific, predetermined paths.

A key assumption when applying betweenness centrality is that transactions or flows tend to follow the shortest paths between nodes. This assumption does not always hold in financial markets, however, where flows may follow more complex, indirect routes. As such, the application of closeness centrality in financial contexts requires careful consideration and may not be universally appropriate.

Nodes with high betweenness centrality are identified as critical intermediaries that control or facilitate the movement of capital, information, or risk across the network. These may include banks that bridge market segments, clearinghouses, or major dealers in the financial industry. Betweenness centrality can also highlight bottlenecks or key links between otherwise disconnected institutions or investor groups.

A node with low betweenness is typically not essential for linking other nodes in the network. Although high betweenness scores often indicate influence, they may also reflect peripheral connectors between clusters. Therefore, interpreting these scores depends heavily on the context and structure of the specific financial network being analyzed.

Closeness Centrality

Closeness centrality measures the average distance between a node and every other reachable node in the network. This metric, based on the foundational work of Linton C. Freeman, is grounded in the idea that a node's importance is inversely related to its geodesic distance from other nodes. It is calculated as the reciprocal of the sum of the shortest path distances from the node to all other nodes in the network.

A node with a higher closeness score (i.e., a lower average distance) can reach other nodes more quickly, indicating greater efficiency in communication or influence across the network.

In the context of financial networks, assets or institutions with high closeness centrality are assumed to respond more rapidly to information, shocks, or risk events compared with those that have lower scores. For example, a financial institution that is well positioned to quickly disseminate or receive contagion would likely exhibit a high closeness centrality value.

The conventional formula for closeness is the inverse of the sum of distances:

$$C(v) = \frac{N-1}{\sum_{u \neq u'} d(u,u')},$$

where N is the number of nodes and $d(u,u')$ is the shortest path distance between the two nodes.

Differentiation can be challenging in highly connected networks because many nodes may have similar, high closeness scores. This measure is also sensitive to the overall structure of the network. Finding all-pairs shortest paths is necessary for calculations, and for very large networks, this process can be computationally demanding.

Eigenvector Centrality

The idea behind eigenvector centrality is that a node's significance is determined by the significance of the nodes it is connected to, not just by the number of connections it has. A node's eigenvector centrality will be high if it is connected to numerous highly central nodes. The components of the principal eigenvector of the network's adjacency matrix (**A**), which corresponds to the largest eigenvalue (λ), are mathematically known as the centrality scores (x):

$$\mathbf{A}x = \lambda x.$$

The key to identifying systemically significant institutions lies in eigenvector centrality, which captures nodes that are influential not only through their direct connections but also through links to other highly connected and influential nodes. Conversely, nodes with low eigenvector centrality are typically connected only to peripheral or less significant nodes.

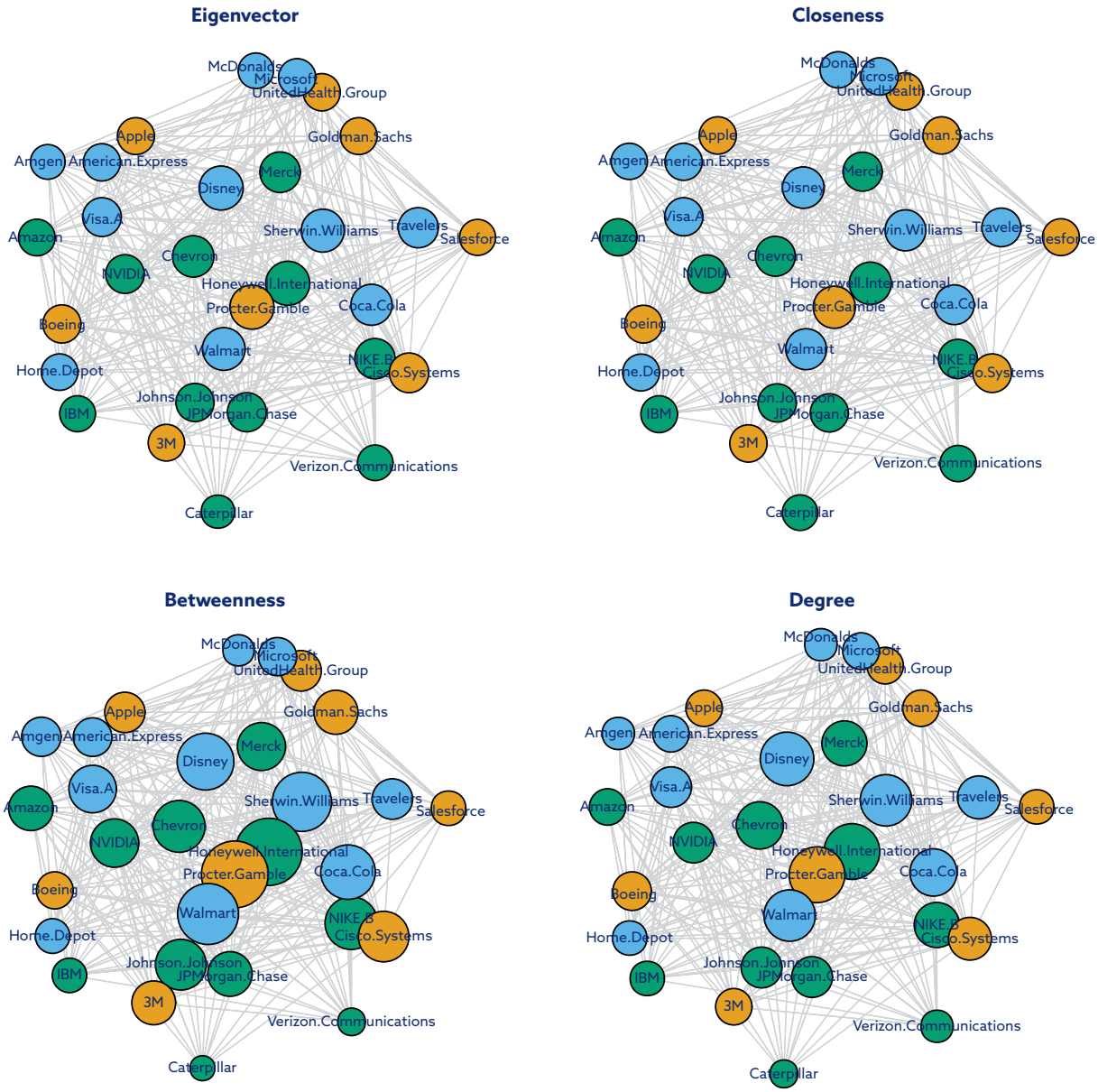
An important property of eigenvector centrality is that it is only appropriate for analyzing nodes that are connected to at least one other node. If a node is completely disconnected from the rest of the network, eigenvector centrality assigns it a score of zero. This scoring may not accurately reflect the node's importance, however, because it ignores exogenous factors that can affect a node's role within the broader system.

A simple example illustrates this limitation. Consider an asset such as the risk-free rate, which may be temporarily disconnected from other assets in a financial network. In this case, eigenvector centrality would assign it a score of zero, implying no influence. This implication is misleading, however, because the risk-free rate is determined by exogenous factors—for instance, central bank policy—that have significant system-wide effects and influence the connectivity and behavior of other assets in the network. Thus, although eigenvector centrality is a powerful tool for identifying influential nodes based on endogenous network structure, it should be interpreted carefully, especially in contexts where external forces play a key role in shaping network dynamics.

A direct comparison of four widely used centrality metrics provides insight into how importance scores vary among algorithms. Each centrality measure captures a distinct aspect of a node's role within the network, resulting in different scores for the same node. In the corresponding network visualizations in **Exhibit 3**, larger node sizes represent higher importance scores based on the respective centrality metric. **Exhibit 4** presents the numerical values associated with

each node under the four centrality measures. Obviously, a node's centrality depends on the specific centrality metric used. For instance, a node with a high betweenness centrality score may not necessarily have a high eigenvector centrality score, because each metric captures different aspects of a node's role within the network.

Exhibit 3. Overview of the Betweenness, Closeness, Eigenvector, and Degree Centrality of a Dow Jones Industrial Average Correlation-Based Network



Source: Bloomberg, LLC.

Exhibit 4. Centrality Scores for Betweenness, Closeness, Eigenvector, and Degree Centrality of a Dow Jones Industrial Average Correlation-Based Network

	Eigenvector	Closeness	Betweenness	Degree
Goldman.Sachs	0.69	0.71	0.01	34
UnitedHealth.Group	0.71	0.71	0.01	34
Microsoft	0.72	0.71	0.01	34
Home.Depot	0.69	0.69	0.01	32
Caterpillar	0.56	0.64	0.00	26
Sherwin.Williams	0.95	0.85	0.02	48
Salesforce	0.66	0.69	0.01	32
Visa.A	0.77	0.74	0.01	38
American.Express	0.71	0.71	0.01	34
McDonalds	0.65	0.67	0.01	30
Amgen	0.62	0.67	0.01	30
Apple	0.70	0.71	0.01	34
Travelers	0.83	0.76	0.01	40
JPMorgan.Chase	0.79	0.74	0.01	38
Honeywell.International	0.99	0.91	0.03	52
IBM	0.66	0.69	0.01	32
Amazon	0.69	0.71	0.01	34
Boeing	0.75	0.72	0.01	36
Procter.Gamble	1.00	0.91	0.03	52
Johnson.Johnson	0.75	0.74	0.01	38
Chevron	0.88	0.81	0.02	44
NVIDIA	0.77	0.74	0.01	38
3M	0.69	0.71	0.01	34
Disney	1.00	0.88	0.02	50
Merck	0.85	0.78	0.01	42
Walmart	0.94	0.85	0.02	48
NIKE.B	0.84	0.78	0.02	42
Coca.Cola	0.87	0.81	0.02	44
Cisco.Systems	0.81	0.76	0.01	40
Verizon.Communications	0.65	0.67	0.00	30

A simple comparison illustrates the point. Consider the eigenvector centrality of Goldman Sachs (GS), and compare it with the degree and closeness centrality that GS has in the network. Recall that eigenvector centrality assigns a higher score to a node that is itself connected to other highly connected nodes. Degree centrality measures the total number of links that a specific node has to other nodes. In this respect, GS has a relatively high eigenvector centrality (0.69) and is highly connected to other highly connected nodes, but its degree of 34 is in the lower percentile of degree connectedness. That is, GS has 34 edges to other nodes. The eigenvector centrality, however, indicates that GS is not connected to highly connected nodes.

Community Detection

Network analysis offers powerful tools not only for identifying individually significant nodes but also for uncovering structural patterns, such as clusters of nodes that are more closely connected to each other than to the rest of the network. These groups are commonly referred to as modules, communities, or components. Identifying such communities helps reveal the underlying structure of a network, providing insights into shared characteristics, functional relationships, and potential vulnerabilities within financial systems.

A community is typically defined as a subset of nodes within a network where the density of internal connections is significantly higher than the density of connections between that subset and the rest of the network. The emergence of community structure depends on the network regime. A network is said to be in a connected regime when it forms cohesive components or communities. The average degree of nodes in the network plays a critical role in this dynamic: Once it exceeds a certain threshold, the network begins to form a connected component.

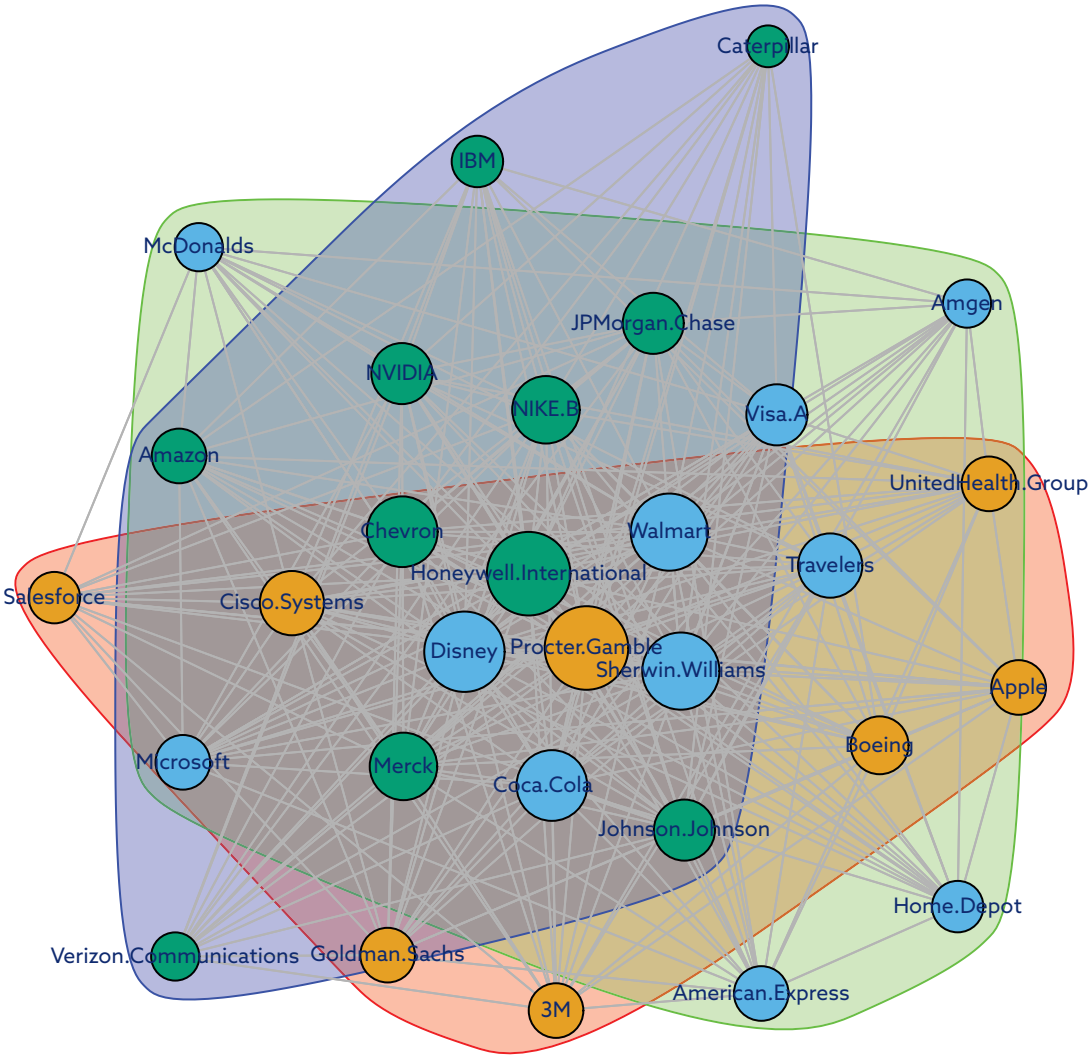
To identify these communities, community detection algorithms aim to maximize the number of intracommunity links while minimizing intercommunity links. These algorithms divide the network into distinct (and sometimes overlapping) communities, and at a coarser resolution, this process effectively reveals the network's global structural organization.

Finding communities in financial networks enables the discovery of significant clusters that may not be visible through node-level analysis alone. The following list provides examples:

- Banking groups: Groups of banks with comparable balance sheets or a high level of interbank activity indicate shared vulnerabilities or concentrated risk.
- Asset communities (clusters): Diversification is aided by identifying sectors or risk exposures by classifying assets based on strong return correlations. Unlike static classifications, such as GICS, these clusters can capture dynamic market structures.
- Functional modules: Communities in supply chains or financial systems frequently represent groups carrying out related tasks.
- Shared interest groups: Communities in financial information networks identify groups that share information sources, tactics, or interests.

Exhibit 5 provides an example for the community detection of the Dow Jones Industrial Average Index according to a correlation-based network and the Cluster Spinglass algorithm according to the highest modularity score.

Exhibit 5. A Correlation-Based Network for the Dow Jones Industrial Average Index



Source: Bloomberg, LLC.

Measuring Community Structure

Assessing the quality of a particular network partition is a major challenge in community detection. The most popular metric for this assessment is modularity.

By comparing the number of edges that actually fall within the suggested communities with the number that would be predicted if edges were dispersed randomly throughout the network while maintaining each node's degree, modularity quantifies how well a network is divided into

communities. Strong community structure is indicated by a high modularity score, which means that the identified communities are much more densely connected internally and sparsely connected externally than would be predicted by chance. Usually, modularity values fall between −0.5 and 1. In real-world networks, values higher than roughly 0.3 are frequently regarded as suggestive of substantial community structure.

The following formula suggested by Clauset, Newman, and Moore (2004) is used to determine the modularity, Q, for a network's partition C with adjacency matrix A:[4]

$$Q = \frac{1}{2M}\sum_{i,j}\left[A_{ij} - \frac{k_i k_j}{2M}\right]\delta(C_i, C_j),$$

where A_{ij} is the weight of the edge connecting nodes i and j (1 if connected and unweighted and 0 otherwise). $k_i = \sum_j A_{ij}$ is the weighted degree of node i. $M = \frac{1}{2}\sum_{i,j} A_{ij}$ is the network's total number of edges (or weight sum). The community that node i is assigned to is called C_i. $\delta(C_i, C_j)$ is the Kronecker delta, which is 0 if nodes i and j are not in the same community and 1 otherwise. In this formula, the difference between the expected and actual edge weights for every pair of nodes in the same community is added up.

For many community detection algorithms, modularity serves as both an objective function and an evaluation metric. One widely used algorithm, the Louvain method, specifically seeks the network partition that maximizes the modularity score. The choice and performance of community detection algorithms depend significantly on the type of graph (e.g., directed vs. undirected) and its characteristics.

According to Yang, Algesheimer, and Tessone (2016), algorithm selection should consider network size (number of nodes), number of edges, and the mixing parameter—a measure of how well defined the communities are. Networks with fewer than 1,000 nodes are generally considered small, and most standard algorithms, including Louvain, Spinglass, and fast greedy, are well suited for these. For larger networks, such algorithms as Multilevel, Walktrap, and Infomap may be more appropriate, particularly as computational efficiency becomes critical.

Community Detection Models

A variety of algorithms have been developed to identify communities in networks. These can be broadly categorized by their methodology:

- Bottom-up approaches begin with each node as its own community and merge communities based on such criteria as modularity gain.
- Top-down approaches start with the full network and iteratively split it, often by removing edges with high betweenness centrality.
- Optimization-based methods aim to maximize a quality function (e.g., modularity) using heuristics.

[4] See Newman (2010) and Bech and Atalay (2010).

Other techniques include random walks, spectral clustering, and statistical inference. Commonly used algorithms in financial applications include Louvain, Girvan–Newman, hierarchical clustering, and *k*-means.

Louvain Method

The Louvain algorithm is a greedy optimization technique that operates in two iterative stages to maximize modularity:

- Local optimization phase: Each node begins in its own community. Nodes are moved to neighboring communities if such moves increase modularity. This process continues until no further improvement is possible.
- Aggregation phase: Identified communities are collapsed into "super-nodes," forming a new network with weighted edges reflecting the total connections between groups. The two-phase process repeats on the new network until modularity no longer increases.

The Louvain method is widely applied in finance to group large sets of stocks based on correlation matrices, identifying risk groups or sectors for portfolio construction.

Girvan–Newman Algorithm

This algorithm identifies communities by progressively removing edges with the highest edge betweenness centrality—those that frequently appear in the shortest paths between nodes. In finance, this approach can be useful for detecting correlated asset clusters or uncovering cohesive subgroups within interbank networks.

Hierarchical Clustering

Hierarchical clustering creates a tree-like structure (dendrogram) of clusters. Variants include the following:

- Single linkage—based on the minimum distance
- Average linkage—based on the average distance
- Complete linkage—based on the maximum distance

Typically, correlation is used as the similarity metric. This method is popular for asset clustering.

k-Means Clustering

In *k*-means, nodes are grouped into a predefined number (k) of clusters by minimizing the distance between each node and its assigned cluster center. Although simple, this process requires prior specification of k and is commonly used for clustering stocks or clients in the financial sector.

The optimal choice of algorithm depends on the network size, network type (weighted/unweighted, directed/undirected), computational resources, and analytical objectives. Because different algorithms may produce distinct community structures from the same dataset, it is crucial to understand their assumptions and test the robustness of the results.

Case Study: Asset Clustering

A well-known application of community detection in finance is grouping financial assets (usually stocks) for the purpose of portfolio diversification. The central idea is that assets in different clusters behave more independently, whereas those in the same cluster exhibit similar behavior (e.g., high return correlation). This strategy can be implemented as follows:

1. Compute pairwise correlations between asset returns over a specified period.
2. Convert the correlation coefficients (ρ) into a distance metric (**d**) using a transformation such as $\mathbf{d}_{ij} = \sqrt{2(1-\rho_{ij})}$. This step is necessary because correlation coefficients should be converted to Euclidean distance metrics.
3. Apply a community detection algorithm (e.g., Louvain, Spinglass, fast greedy, leading eigenvector, Walktrap, or Multilevel) on the resulting network.
4. Select representative assets from each cluster. Selection criteria may include node centrality, cluster size, or other financial metrics.

The effectiveness of a cluster-based investment strategy can be assessed by comparing portfolio performance with benchmarks or peer strategies. Standard evaluation tools include the Sharpe ratio, the information ratio, and factor exposure analysis. These metrics help determine whether the identified clusters and community structures contribute meaningfully to risk-adjusted returns.

Network Dynamics

Financial systems are inherently dynamic and evolving. Modeling these dynamics—especially the spread of shocks and the emergence of systemic risk—is a core application of network theory. Mathematical models from network analysis provide essential tools for evaluating these network dynamics across different market cycles. Key structural metrics used to capture and assess financial network behavior include the following:

- Edge density
- Reciprocity (in directed networks)
- Assortativity degree
- Transitivity
- Mean distance
- Diameter
- Mean degree

These measures help reveal patterns in connectivity and vulnerability, offering insights into how financial systems adapt and destabilize under stress.

Systemic Risk

Following Konstantinov and Fabozzi (2025), the focus in assessing systemic risk lies not solely on its origins—often stemming from external factors, such as macroeconomic shifts, financial distress among key institutions, or sovereign crises—but also on how it spreads and permeates through the network. Thus, systemic risk analysis focuses on the potential failure of individual nodes within the system because their risk exposures can precipitate broader systemic repercussions. Nodes play a pivotal role here because this analysis indicates which nodes are most susceptible to being affected, either simultaneously or in a specific sequence.

Systemic risk is the possibility that a single shock or failure will set off a series of events that could cause the financial system to collapse or be severely disrupted, affecting its functionality and potentially impacting the real economy. It is an emergent characteristic resulting from the interdependence of financial institutions and their group dynamics.

To properly understand the meaning of systemic risk, financial contagion, and spillover, it is necessary to understand the metrics that describe networks and capture interconnectedness. Following Konstantinov and Fabozzi (2025), financial networks are characterized by specific metrics. A thorough understanding of these metrics is critical for understanding network dynamics and interconnectedness. Some common network properties include size, degree, density, and community structure, among others. According to Konstantinov and Fabozzi (2025), "These metrics refer to the overall properties of a network and aim to describe its underlying structure. The structure of network connectedness depends on the underlying information flow process, or how nodes interact over the edges." The metrics that describe the concentration of interconnectedness and the degree of nodal connectedness deserve special attention.

- *Role of density:* Network connectivity and financial stability have a complicated relationship that is frequently characterized as "robust yet fragile." Although fully connected networks offer maximum diversification, moderate connectivity can increase resilience by distributing minor shocks across more institutions. In other words, a network with few links is more fragile than a network with a large number of links. However, higher connectivity has the potential to magnify significant shocks beyond a certain threshold, facilitating quicker contagion and enhancing systemic fragility. As a result, there is a tipping point at which additional links cause the system to become unstable rather than stable.
- *Role of concentration:* Das (2016) showed that even when the overall level of connectivity is the same, financial systems with highly concentrated exposures are typically more susceptible to contagion and systemic risk than systems with more dispersed exposures. In a concentrated system, partners may suffer catastrophic losses if a major counterparty fails.

High-centrality nodes often play a critical role in systemic risk. Hubs—measured by degree, or eigenvector—are systemically important financial institutions whose failure can trigger widespread contagion, making their identification essential for regulation. Bridges, characterized by high betweenness centrality, act as key intermediaries, and their failure can disrupt the flow of capital or information across the network.

Because of both confidentiality and complexity, a major barrier to financial network modeling is limited data. Regulators frequently have access to only a portion of the data, such as aggregate exposures. Although such methods as sparse reconstruction and maximum entropy aid in estimating network structure, they may understate systemic risk if they fail to account for network sparsity or assume uniform link weights.

Financial Contagion and Spillover

Risk propagation in financial networks has been studied using mathematical epidemiology models because of the similarities between financial contagion and disease spread. Financial contagion occurs when a shock or distress event originating in one part of the financial system (e.g., an institution, market segment, or asset class) spreads across the network, potentially causing systemic failures or widespread instability. This process is often compared with epidemic spread or a domino effect, driven by the intricate interconnectedness of financial entities. There are two broad channels of contagion:

- *Direct linkages:* These arise from explicit contractual obligations between financial institutions. For example, if Institution A defaults on a loan owed to Institution B (as in interbank lending), Institution B may suffer direct financial losses. These losses can impair its ability to meet obligations, possibly triggering further defaults.
- *Indirect linkages:* These occur without direct contractual ties, operating through market-wide mechanisms, such as liquidity herding, fire sales of assets, information contagion, or common asset exposures.

Indirect and direct channels often interact. For instance, a direct counterparty loss may induce fire sales, which depress asset values and spark funding runs, escalating the contagion. Network models aim to capture and simulate these complex interactions, offering regulators and analysts a way to assess vulnerability propagation paths.

Case Study: Global Financial Crisis in 2008

Financial shocks can be amplified by network structure, as the 2008 crisis showed. Through a highly interconnected system that included repos, interbank lending, and complex derivatives, such as mortgage-backed securities, collateralized debt obligations, and CDSs, what started as losses in the US subprime mortgage market swiftly spread throughout the world. The September 2008 collapse of Lehman Brothers, a major market node, sparked widespread concerns about counterparty risk. This heightened concern resulted in repos and interbank lending being frozen, demonstrating how the collapse of a major institution can cause systemic instability. The risk of direct contagion was also made evident by AIG's near default. Because AIG had sold significant amounts of CDS protection to big banks, its failure would have resulted in massive losses all at once, necessitating a government bailout to stop further collapse. The crisis was made worse by fire sales.

The global financial crisis emphasized how important it is to adopt a regulatory approach that goes beyond the solvency of individual institutions and specifically takes into account both network interconnectedness and systemic risk. Tools for network analysis became crucial for comprehending these weaknesses. The crisis brought to light the system's "robust-yet-fragile" nature: Interconnectedness that could withstand minor shocks turned into a pathway for catastrophic failure when those shocks grew too big.

Investment Management

Network theory provides useful tools and perspectives for investment management practitioners engaged in portfolio construction, market prediction, and pattern recognition, in addition to risk management and regulatory oversight.

Diversification in Portfolio Construction: Modern Portfolio Theory vs. Network Theory

Expected returns, variances, and the pairwise covariance matrix of assets are the main components of traditional portfolio construction, which is dominated by Markowitz's mean–variance optimization (MVO). MVO has real-world drawbacks, however, such as sensitivity to errors in input estimation (particularly expected returns) and potentially unstable or unduly concentrated allocations. Through the explicit incorporation of the richer structure of asset relationships uncovered by network analysis, network-based approaches seek to improve or offer alternatives. A direct relationship exists between the MVO framework and network centrality scores, which is extensively discussed in Zareei (2019), Ciciretti and Pallotta (2024), and Konstantinov and Fabozzi (2025), among others.

For a minimization of portfolio risk, the mathematical algorithm searches for all possible weights w that minimize the portfolio risk captured by the portfolio variance σ^2_{PF}, which is a weighted product of the individual weights and the variance–covariance matrix, $\Sigma = [\sigma_{ij}]$ (see Konstantinov, Fabozzi, and Simonian 2023). That is,

$$\min_w \sigma^2_{PF} = w'\Sigma w$$
$$\text{s.t. } x \geq L,$$

where w is a transposed n-dimensional vector of portfolio weights and Σ is the n-dimensional variance–covariance matrix of the portfolio assets. The variable x represents the portfolio return, while L represents the minimum return value that the portfolio must satisfy. The weights that satisfy the optimizations are

$$w_{min} = \frac{1}{1'\Sigma^{-1}1}\Sigma^{-1}1,$$

where Σ^{-1} is the inverse covariance matrix and 1 represents vectors of ones.

Using the notion of the covariance matrix, Σ, that it is a product of the correlation matrix, Ω, and the diagonal matrix, Δ, whose entries are the variances with $\sigma_i = \sqrt{\sigma_{ij}}$, then the relationship is $\Sigma = \Delta\Omega\Delta$ and the weights subject to the correlation matrix are

$$\frac{1}{1'\Sigma^{-1}1}\Sigma^{-1}1\Omega^{-1}.$$

Considering the mean–variance framework, the diversification, or minimum risk given the expected level of return as measured by the portfolio variance and return $E(r)$, is computed as follows:

$$\min_w \sigma^2_{PF} = w'\Sigma w \text{ with } R_{PF} = w'E(r).$$

As a result, the weights of the mean–variance framework are a function of the expected portfolio returns and the covariance matrix of the assets, which gauges risk:

$$w_{mv} = \frac{R_{PF}}{E(r)'\Sigma^{-1}E(r)}\Sigma^{-1}E(r),$$

where Σ^{-1} is the inverse covariance matrix, $E(r)$ represents vectors of expected asset returns, and R_{PF} is the portfolio return. Similarly, we can obtain the weights in the mean–variance framework using the correlation matrix.

In contrast, in network theory, diversification in a network context is identified by weaker relationships between nodes in a graph. The shorter the distances between nodes, the greater the potential impact of risk transmission. This limitation, as noted by Peralta and Zareei (2016) and Zareei (2019), has been addressed and relaxed by Konstantinov (2022), whose approach has found broader application in portfolio allocation, as discussed later in this chapter. Fundamentally, the transfer of risk between nodes is central when using networks for portfolio allocation.

It is important to highlight that centrality scores—such as degree, eigenvector, or alpha centralities—are critical here because they help identify how risk might flow and affect nodes. Degree centrality is not the preferred score, however, because it measures only the immediate connectedness of a node, whereas other centrality metrics capture the more nuanced interconnectedness and influence of a node within the network.

When network theory is applied to portfolio construction, several methods can be used to heighten the diversification of asset portfolios, including the following:

- *Network-based asset selection:* These techniques seek to combine related assets to ensure diversified portfolio exposure across various sources of risk and return.
- *Community detection on association, probabilistic, or statistical networks:* To find clusters of co-moving assets, such algorithms as Louvain, Spinglass, leading eigenvector, or fast greedy provide efficient ways to detect community affiliations with differing factor exposures. Hierarchical clustering or *k*-means methods are applied to asset correlation or distance matrices. Instead of relying on static classifications, selecting representative assets from each cluster can improve diversification and better capture changing market structures. This approach does not require a specific network formulation and is widely used in finance.
- *Graph filtering:* The minimum spanning tree (MST) method highlights important relationships and hierarchies by extracting simplified structures from dense correlation matrices. MST connects all assets with the smallest possible total edge distances. Asset selection can be guided by examining these structures (e.g., central versus peripheral nodes). Some research has shown that portfolios constructed from peripheral MST nodes exhibit strong performance. However, a major drawback of using MST to visualize graphs is that important links might be omitted. Therefore, MST should be used with great caution.
- *Network-based weighting schemes:* Network principles can also guide capital distribution among assets after selection.

- *Centrality-based weighting:* The position of an asset in the network is directly used for weighting. There is an inverse relationship between centrality scores and the weights assigned in the minimum-variance framework. Consequently, peripheral assets are favored, potentially lowering contagion risk. This decision, however, may vary depending on market conditions. Expected returns also play an essential role because centrality-based algorithms focus on risk and interconnectedness but do not account for expected return impacts. Other centrality metrics can be applied to better integrate expected returns.
- *Fundamental networks:* These link assets not only by price movements but also by financial fundamentals (such as profits). To enhance diversification, more capital is allocated to assets that are fundamentally distinct from others. Financial research offers examples of leveraging such information with centrality metrics and risk management tools.
- *Hierarchical risk parity of a community structure:* By using clustering to group similar assets, this model distributes capital among groups to achieve a more balanced risk allocation (see Raffinot 2018).
- *Network risk parity:* A variant of hierarchical risk parity, this method uses an asset's degree of network connectivity based on risk. To reduce exposure to systematic risks that might become systemic, assets less central in the network are given more weight.

As the number of assets increases, these network-based techniques aim to achieve more robust diversification and potentially better risk-adjusted performance than traditional methods relying solely on pairwise statistics. They do so by explicitly leveraging the topological structure of asset relationships.

Market Prediction

Network theory offers frameworks and inputs for forecasting market movements and creating innovative trading tactics. Profitable opportunities or information about future price behavior can be found in the structure of connections between entities. Predictive signals can also be derived from network properties, such as reciprocity, edge density, clustering coefficients, and change in degree centralities.[5] Variations in network density or structure may indicate changes in market volatility or regimes. Spillover effects, in which the actions of related entities affect the performance of a target entity in the future, can be captured by network analysis and applying appropriate network metrics, such as network nodal entropy, Ricci curvature, fragility, or criticality indicators.[6]

One area of quantitative finance that is expanding quickly is the fusion of network science and artificial intelligence, specifically machine learning and deep learning. Although more straightforward techniques can still be competitive, deep neural networks (DNNs) frequently outperform conventional models in forecasting stock returns. Adjusting to shifting market conditions is a major challenge in these models; by modifying regularization according to recent performance, however, such strategies as online early stopping are helpful.

[5] See Das (2016) and Konstantinov, Chorus, and Rebmann (2020) for several network-based indicators to predict market behavior.

[6] See Konstantinov and Fabozzi (2025) for extensive discussion and estimation of risk indicators in portfolio management.

Graph neural networks (GNNs) are a category of DNNs that are perfect for modeling financial relationships such as stock interdependencies because they can learn from both node features and network structure, modeling the dependencies between entities. A trading GNN, for instance, has been proposed as a way to calculate the impact of assets, dealers, and their relationships on prices (Wu 2025).

By using attention scores to weight neighbors differently, graph attention networks outperform GNNs in identifying the most pertinent connections. This ability helps with stock prediction and portfolio optimization in financial networks where relationships are complex and have different levels of importance.

Conclusion

As this chapter has shown, applying network theory to financial markets and institutions offers finance professionals useful tools and insightful perspectives. The intricate reality of interconnectedness is not captured by the conventional perspective of isolated actors or homogeneous systems. Network theory provides a strong framework for explicitly modeling these connections.

The integration of AI, the use of multiplex and multilayer networks, developments in explainable AI (XAI) for improved interpretability, and the inclusion of alternative data sources are some new trends in the application of network theory to finance.

Although network analysis provides insightful information, some obstacles exist for applying it in finance. Because data on exposures such as loans or derivatives are often confidential, practitioners are forced to work with partial or proxy data, which introduces potential biases. Complexity, speed, and interpretability must all be balanced in advanced models, which can be computationally taxing, particularly for dynamic or AI-driven models. Additionally, there is no one-size-fits-all tool. The data, research question, and objectives must all be considered when selecting a network structure, metrics, or algorithms. Drawing reliable conclusions requires an understanding of each tool's limitations.

References

Barabási, Albert-László. 2016. *Network Science*. Cambridge, UK: Cambridge University Press.

Bech, Morten L., and Enghin Atalay. 2010. "The Topology of the Federal Funds Market." *Physica A: Statistical Mechanics and Its Applications* 389 (22): 5223–46. doi:10.1016/j.physa.2010.05.058.

Borgatti, Stephen P., Martin G. Everett, Jeffrey C. Johnson, and Filip Agneessens. 2022. *Analyzing Social Networks Using R*. Los Angeles: SAGE Publications.

Ciciretti, Vito, and Alberto Pallotta. 2024. "Network Risk Parity: Graph Theory-Based Portfolio Construction." *Journal of Asset Management* 25 (2): 136–46. doi:10.1057/s41260-023-00347-8.

Clauset, Aaron, Mark E. J. Newman, and Cristopher Moore. 2004. "Finding Community Structure in Very Large Networks." *Physical Review E* 70 (6): 6–11. doi:10.1103/PhysRevE.70.066111.

Das, Sanjiv Ranjan. 2016. "Matrix Metrics: Network-Based Systemic Risk Scoring." *Journal of Alternative Investments* 18 (4): 33–51. doi:10.3905/jai.2016.18.4.033.

Diebold, Francis X., and Kamil Yilmaz. 2015. *Financial and Macroeconomic Connectedness: A Network Approach to Measurement and Monitoring*. New York: Oxford University Press. doi:10.1093/acprof:oso/9780199338290.001.0001.

Konstantinov, Gueorgui S. 2022. "Emerging Market Bonds: Expected Returns and Currency Impact." *Journal of Portfolio Management* 48 (8): 139–58. doi:10.3905/jpm.2022.1.385.

Konstantinov, Gueorgui S., Irene Aldridge, and Hossein B. Kazemi. 2023. "Financial Networks and Portfolio Management." *Journal of Portfolio Management* 49 (9): 190–216. doi:10.3905/jpm.2023.1.525.

Konstantinov, Gueorgui, Andreas Chorus, and Jonas Rebmann. 2020. "A Network and Machine Learning Approach to Factor, Asset, and Blended Allocation." *Journal of Portfolio Management* 46 (6): 54–71. doi:10.3905/jpm.2020.1.147.

Konstantinov, Gueorgui S., and Frank J. Fabozzi. 2025. *Network Models in Finance: Expanding the Tools for Portfolio and Risk Management*. Hoboken, NJ: John Wiley & Sons.

Konstantinov, Gueorgui S., Frank J. Fabozzi, and Joseph Simonian. 2023. *Quantitative Global Bond Portfolio Management*. Singapore: World Scientific Publishing. doi:10.1142/13313.

Newman, Mark. 2010. *Networks: An Introduction*, 1st ed. Oxford, UK: Oxford University Press. doi:10.1093/acprof:oso/9780199206650.001.0001.

Peralta, Gustavo, and Abalfazl Zareei. 2016. "A Network Approach to Portfolio Selection." *Journal of Empirical Finance* 38 (Part A, September): 157–80. doi:10.1016/j.jempfin.2016.06.003.

Raffinot, Thomas. 2018. "Hierarchical Clustering-Based Asset Allocation." *Journal of Portfolio Management* 44 (2): 89–99. doi:10.3905/jpm.2018.44.2.089.

Wasserman, Stanley, and Katherine Faust. 1994. *Social Network Analysis: Methods and Applications*. New York: Cambridge University Press. doi:10.1017/CBO9780511815478.

Wu, Xian. 2025. "Trading Graph Neural Network." Working paper (10 April). doi:10.48550/arXiv.2504.07923.

Yang, Zhao, René Algesheimer, and Claudio J. Tessone. 2016. "A Comparative Analysis of Community Detection Algorithms on Artificial Networks." *Scientific Reports* 6. doi:10.1038/srep30750.

Zareei, Abalfazl. 2019. "Network Origins of Portfolio Risk." *Journal of Banking & Finance* 109 (December). doi:10.1016/j.jbankfin.2019.105663.

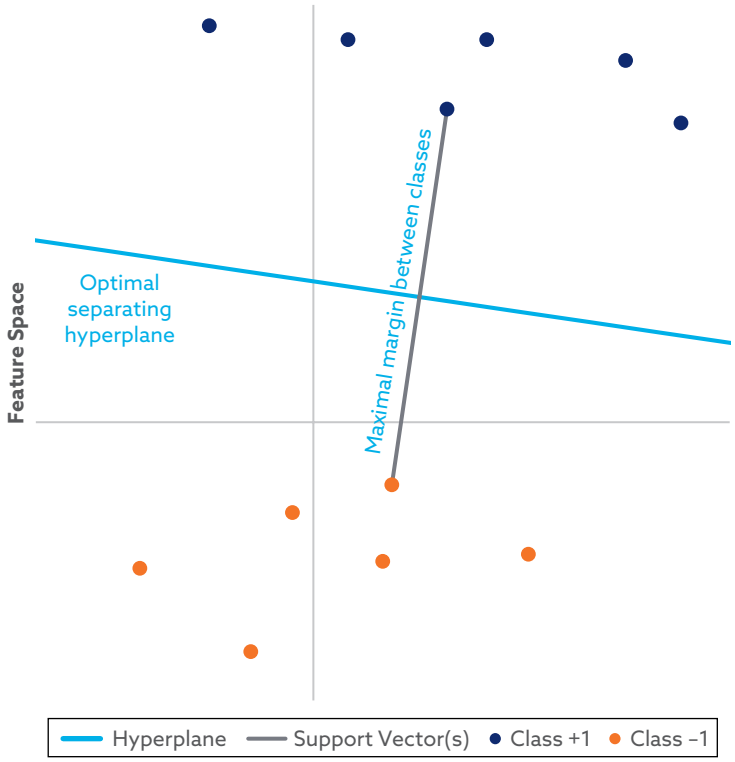
SUPPORT VECTOR MACHINES

Maxim Golts, PhD
VP, Multi-Asset Solutions, PGIM
Senior Advisor, Questrom School of Business, Boston University

Introduction

Support vector machines (SVMs) are a powerful machine learning tool developed by Vladimir Vapnik and Corinna Cortes in the 1990s as an alternative to neural networks (see Cortes and Vapnik 1995; Vapnik 1999, 2000). They were developed to address classification problems. The basic idea is to separate the training data points corresponding to two different classes, often labeled +1 and –1, by an *affine hyperplane* in the feature space (see **Exhibit 1**). According to Vapnik (1999, p. 996), "We say that this set of vectors is separated by the *optimal hyperplane (or the maximal margin hyperplane)* if it is separated without error and the distance between the closest vector and the hyperplane is maximal."

Exhibit 1. SVM Setup: Data in the Feature Space, Support Vectors, and the Optimal Separating Hyperplane



Note: This is a generic SVM illustration with random data generated by the author.

 •

Technical Concepts

Let us start by considering a classification problem with training data of the following form:

$$(x_1,y_1),\ldots,(x_l,y_l),\ x\in R^n,\ y\in\{+1,-1\}.$$

In other words, vectors $x \in R^n$, often called *inputs* or *features*, in the training data belong to two classes: those paired with $y = +1$ and those paired with $y = -1$. The goal of any learning machine is to find a functional form for this classification, fitting the training data as well as possible but not overfitting.

One simple case is when the two classes of $x \in R^n$ can be separated by a *hyperplane:* an affine subspace of R^n of codimension 1. Then the class of $x \in R^n$ corresponding to $y = +1$ will simply be on one side of the hyperplane, and the class corresponding to $y = -1$ will be on the other side.

In practice, such a separating hyperplane does not exist for many real-life problems. Following Cortes and Vapnik (1995) and Vapnik (1999, 2000), there are two major approaches to constructing learning machines for the classification problems:

- Use functional approximations by nonlinear (e.g., *sigmoid*) functions. This approach leads to neural networks, which are beyond the scope of this chapter.
- Map (nonlinearly) the input vectors $x \in R^n$ into a *higher-dimensional feature space*, and construct a separating hyperplane in this higher-dimensional space. This approach leads to SVMs, which we will consider here in more detail.

An SVM linear model developed in such a higher-dimensional feature space corresponds to a nonlinear class separation model in the original feature space. Support vector machines (and neural networks) can be used in machine learning beyond classification problems, such as regression and density estimation problems (see, e.g., Vapnik, Golowich, and Smola 1997; Smola and Schölkopf 2004; Awad and Khanna 2015).

Figure 1. Hyperplane-Separable Data

Support vector machines were introduced by Vladimir Vapnik and Corinna Cortes (see Cortes and Vapnik 1995; Vapnik 1999, 2000). Suppose the training data are of the form

$$(x_1,y_1),\ldots,(x_l,y_l),\ x\in R^n,\ y\in\{+1,-1\}$$

and the two classes corresponding to $y = +1$ and $y = -1$ *can be separated* by an (affine) hyperplane, called an "optimal separating hyperplane" or a "maximal margin hyperplane":

$$(w\cdot x)-b=0,$$

where w is the vector orthogonal to the optimal hyperplane and b is a scalar. Vapnik (1999) describes the optimal hyperplane using the following inequalities:

$$(w \cdot x_i) - b \geq 1 \text{ if } y_i = 1.$$

$$(w \cdot x_i) - b \leq -1 \text{ if } y_i = -1.$$

These inequalities can be expressed compactly as follows:

$$y_i\left[(w \cdot x_i) - b\right] \geq 1,\ i = 1, \ldots, I.$$

The maximal margin hyperplane is obtained by solving the following optimization problem:

$$\min_{w,b}(w,w)$$

$$\text{subject to } y_i\left[(w \cdot x_i) - b\right] \geq 1,\ i = 1, \ldots, I.$$

Following Vapnik (1999), this optimization problem can be set up with Lagrange multipliers α_i:

$$\min_{\alpha} \sum_{i=1}^{I} \alpha_i - \frac{1}{2} \sum_{i,j}^{I} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$

$$\text{subject to} \sum_{i=1}^{I} \alpha_i y_i = 0 \text{ and } 0 \leq \alpha_i,\ i = 1, \ldots, I.$$

The optimal solution vector w_0 is a linear combination of the vectors in the training set:

$$w_0 = \sum_{i=1}^{I} y_i \alpha_i^0 x_i,\ \alpha_i^0 \geq 0,\ i = 1, \ldots, I.$$

Moreover, Vapnik shows that only some training vectors x_i, called the *support vectors*, have nonzero coefficients in the expansion of w_0. In other words, we have

$$w_0 = \sum_{\text{support vectors}} y_i \alpha_i^0 x_i,\ \alpha_i^0 \geq 0.$$

Figure 2. Generalization for Linearly Nonseparable Data

Dealing with the case when the data are linearly nonseparable, following Vapnik (1999), we set

$$\min_{w,b}(w,w)+C\sum_{i=1}^{I}\xi_i$$

$$\text{subject to } y_i\left[(w\cdot x_i)-b\right]\geq 1-\xi_i,\ i=1,\ldots,I.$$

The idea is to allow some points to be on the "wrong" side of the optimally separating hyperplane. The optimization problem can be formulated as follows:

$$\min_{\alpha}\sum_{i=1}^{I}\alpha_i-\frac{1}{2}\sum_{i,j}^{I}\alpha_i\alpha_j y_i y_j(x_i\cdot x_j)$$

$$\text{subject to } \sum_{i=1}^{I}\alpha_i y_i=0 \text{ and } 0\leq\alpha_i\leq C,\ i=1,\ldots,I.$$

More generally, we can map the input vectors $x\in R^n$ into a very higher-dimensional feature space through some nonlinear mapping chosen a priori and construct an optimal separating hyperplane in this high-dimensional feature space. It turns out we do not need to know the explicit form of this nonlinear map; all we need is the inner product in the higher-dimensional space, represented by a two-variable function $K(x_i, x_j)$ for $x_i, x_j\in R^n$, called the *kernel*.

Then, the most general optimization problem can be formulated as follows:

$$\min_{\alpha}\sum_{i=1}^{I}\alpha_i-\frac{1}{2}\sum_{i,j}^{I}\alpha_i\alpha_j y_i y_j K(x_i\cdot x_j)$$

$$\text{subject to } \sum_{i=1}^{I}\alpha_i y_i=0 \text{ and } 0\leq\alpha_i\leq C,\ i=1,\ldots,I.$$

One of the most important concepts in learning theory is the *trade-off* between the quality of the model fit on the training data and the complexity of the model. In fact, support vector machines were inspired by the so-called structural risk minimization principle, arising in statistical learning theory to address this kind of trade-off.

Intuitively, one can achieve a great model fit in the training data, but the confidence interval will be large, resulting in *overfitting*—fitting the spurious patterns in the training data too well, at the expense of poor performance on the data the model has not seen

in training. Vapnik (1999) points out that SVMs allow the control of both model fit and the confidence interval. In the most general case, the unique optimization solution is obtained when one chooses the value of the *trade-off* parameter, *C*.

Simple Example: Classifying Stocks vs. Bonds

In this section, I illustrate the support vector machine concepts via a simple example. For a set of equity and bond indexes, we want to separate the set in a two-dimensional feature space: the correlation between an index return and inflation (Consumer Price Index, or CPI, change) and the correlation between the index return and real GDP growth. We calculate quarterly correlations using year-over-year inflation, year-over-year GDP growth rates, and (the prior quarter's) equity and bond index year-over-year (YoY) returns. We use quarterly data between 1998 and 2025.

For example, for US large caps (represented by the MSCI USA Gross Total Return USD Index), the correlation with year-over-year inflation is 20% and the correlation with GDP growth is 60%. Thus, US large caps are represented in our two-dimensional feature space by a vector:

$$x = (0.20, 0.60) \in R^2.$$

Similarly, US Treasuries (represented by the Bloomberg US Treasury Total Return Index), are −39% correlated with the subsequent quarter's inflation and −42% correlated with the subsequent quarter's GDP growth. Thus, US large caps are represented in our two-dimensional feature space by a vector:

$$x = (-0.39, -0.42) \in R^2.$$

We train the model on three equity indexes (the MSCI USA Gross Total Return USD Index, MSCI USA Small Cap Gross Total Return USD Index, and MSCI USA IMI High Dividend Yield Gross Total Return USD Index), setting y = +1, and three bond indexes (the Bloomberg US Treasury Total Return Index, Bloomberg US Corporate Investment Grade Total Return Index, and Bloomberg US Treasury Inflation-Linked Total Return Index), setting y = –1. Not surprisingly, all equity indexes are quite significantly positively correlated with real GDP growth, and US Treasuries are negatively correlated with real GDP growth. See **Exhibit 2** for more data details.

According to Vapnik (1999), optimizing the SVM's Lagrangian on our six training data points reduces to the following problem:

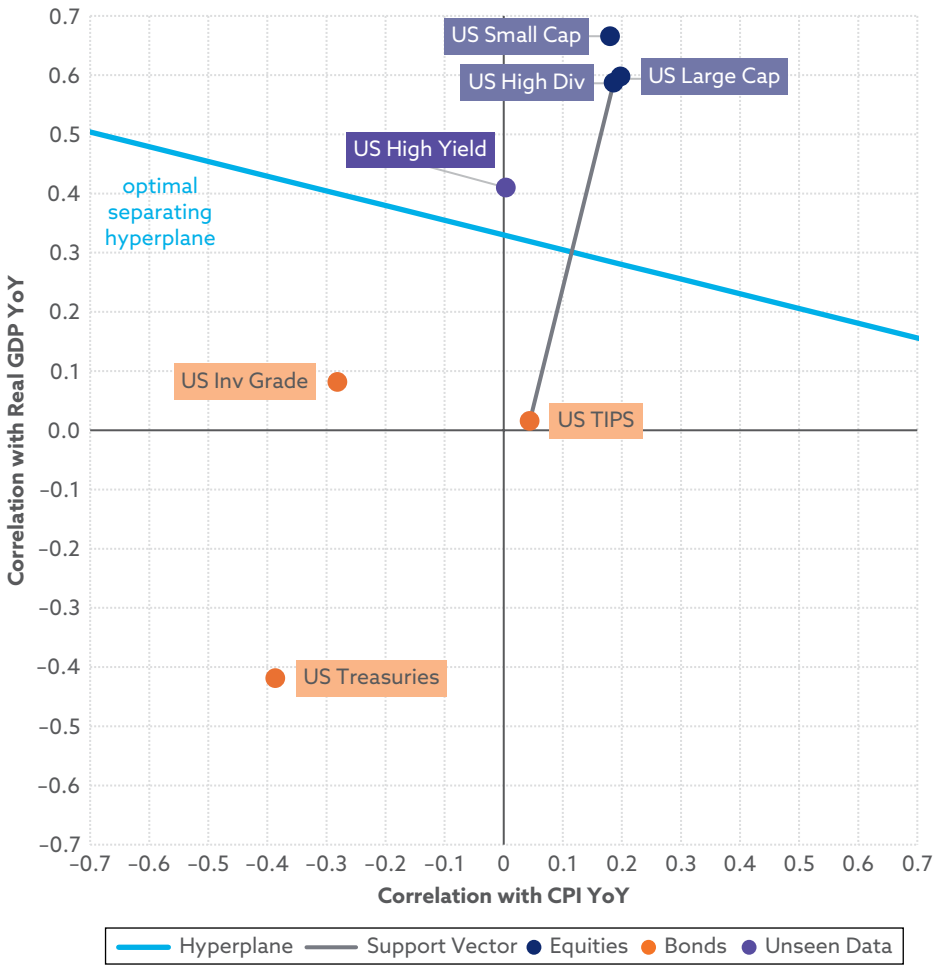
$$\min_{\alpha} \sum_{i=1}^{6} \alpha_i - \frac{1}{2} \sum_{i,j}^{6} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$

$$\text{subject to } \sum_{i=1}^{6} \alpha_i y_i = 0 \text{ and } 0 \le \alpha_i,\ i = 1, \ldots, 6,$$

where α_i are Lagrange multipliers. Refer back to Figure 1 for the general optimization setup.

Exhibit 2. Optimal Hyperplane Separating Stock and Bond Indexes in a Two-Dimensional Feature Space of Correlations with Real GDP and CPI Year over Year, 1998–2025

		Training Data						Unseen Data
	Correlations	US Large Cap	US Small Cap	US High Div	US Treasuries	US Inv Grade	US TIPS	US High Yield
Features	CPI YoY	0.20	0.18	0.19	−0.39	−0.28	0.04	0.00
	Real GDP YoY	0.60	0.67	0.59	−0.42	0.08	0.02	0.41
	y: stock (+1) or bond (−1)	1	1	1	−1	−1	−1	
	α^0: Lagrange multipliers	0	0	5.7639	0	0	5.7639	
	$(w \cdot x) - b$	1.045	1.253	1.000	−2.784	−1.048	−1.000	0.267



Notes: Quarterly correlations of CPI year-over-year change and real GDP year-over-year growth with the prior quarter's year-over-year return of equity and bond indexes. Author's calculations for illustration and educational purposes only.

Sources: Data are from Bloomberg (US Large Cap: MSCI USA Gross Total Return USD Index; US Small Cap: MSCI USA Small Cap Gross Total Return USD Index; US High Div: MSCI USA IMI High Dividend Yield Gross Total Return USD Index; US Treasuries: Bloomberg US Treasury Total Return Index; US Inv Grade: Bloomberg US Corp Investment Grade Total Return Index; US TIPS: Bloomberg US Treasury Inflation-Linked Total Return Index; US High Yield: Bloomberg US Corp High Yield Total Return Index).

After solving this problem, we can see that the optimal separating hyperplane is defined by two support vectors: the US high-dividend index and the US Treasury Inflation-Protected Security index. We have the optimized solution vector:

$$w_0 = (+1)\alpha^0_{US\,High\,Div} x_{US\,High\,Div} + (-1)\alpha^0_{US\,TIPS} x_{US\,TIPS},$$

where the (optimized) Largange multipliers, $\alpha^0_{US\,High\,Div} = \alpha^0_{US\,TIPS} = 5.7639$, are corresponding to the support vectors, $x_{US\,High\,Div}$ and $x_{US\,TIPS}$, and the Lagrange multipliers corresponding to the four nonsupport vectors are zero.

Further, we have

$$b = \frac{1}{2}(w_0 \cdot x_{US\,High\,Div} + w_0 \cdot x_{US\,TIPS}),$$

so that

$$(w_0 \cdot x_{US\,High\,Div}) - b = 1 \text{ and } (w_0 \cdot x_{US\,TIPS}) - b = -1.$$

Also note that the US large-cap index is fairly close to being a support vector:

$$(w_0 \cdot x_{US\,Large\,Cap}) - b = 1.045.$$

We can now use our SVM model to classify a data point the model has never seen: US high-yield bonds. The correlations of the Bloomberg US Corporate High Yield Total Return Index with CPI year-over-year change and real GDP year-over-year growth (features $x \in R^2$) put the US high-yield index on the side of equities, albeit close to the optimal separating hyperplane:

$$(w_0 \cdot x_{US\,High\,Yield}) - b = 0.267.$$

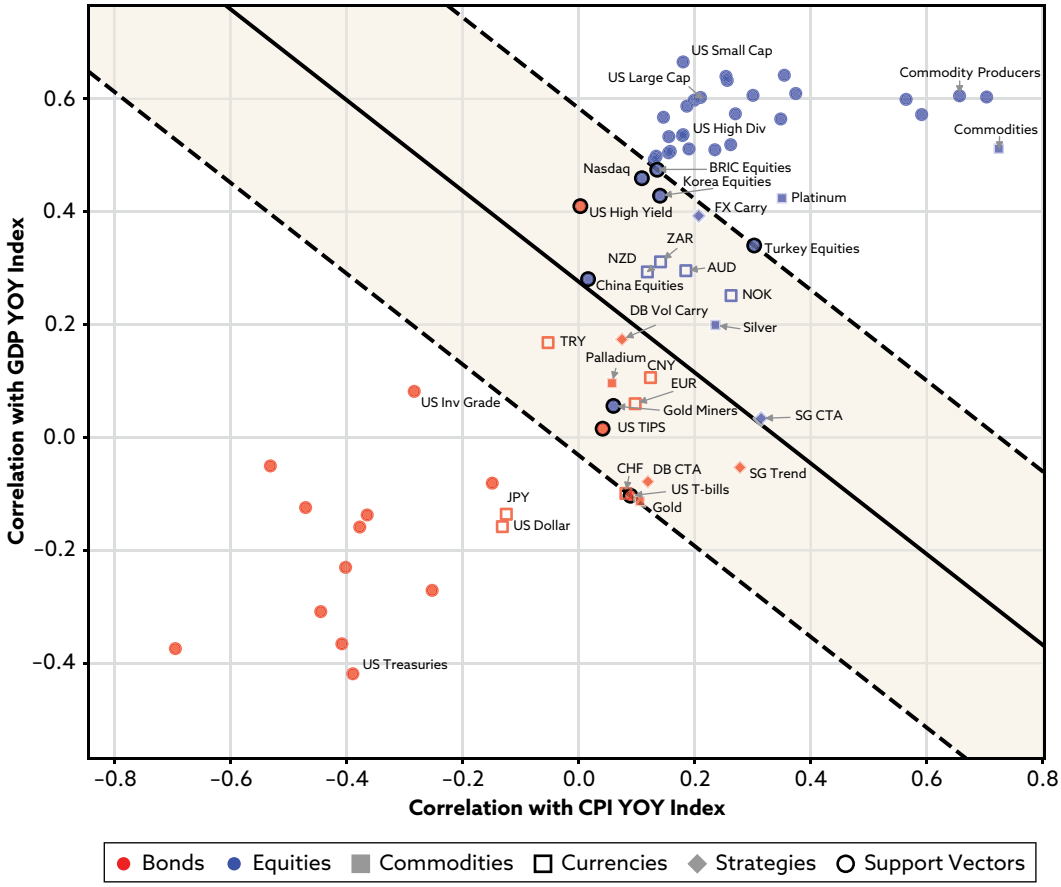
In this sense, the SVM model classifies the US high-yield index as equities rather than bonds.

For **Exhibit 3**, arising from the asset allocation research conducted in collaboration with Jiahui "Joy" Yu, PhD, we trained the data on the correlations of real GDP and CPI YoY with the prior quarter's year-over-year returns of 32 equity indexes and 15 bond indexes. In this case, stocks and bonds are not linearly separable and the trade-off parameter is set as $C = 1$, as described in Figure 2.

The support vectors in this case are six equity indexes (NYSE Arca Gold Miners Index, MSCI China Gross Total Return Local USD Index, Nasdaq-100 Total Return Index, MSCI Emerging Korea Gross Total Return Local Index, MSCI Emerging Turkey Gross Total Return Local Index, and MSCI EM BRIC Gross Total Return Local Index) and three bond indexes (Bloomberg US Treasury Inflation-Linked Total Return Index, Bloomberg US Corporate High Yield Total Return Index, and Bloomberg US Treasury Bill Index).

The trained model is then used to classify 21 other assets. Note that gold miners (NYSE Arca Gold Miners Index), gold, Japanese yen, the US Dollar Index, and CTA strategies (DB Cross Asset CTA Trend Index and SG Trend Index) are classified as bond-like assets, albeit "within the margin." US high-yield bonds (Bloomberg US Corporate High Yield Total Return Index) are classified as equity-like, similarly to the model in Exhibit 2, albeit "within the margin" in this model with regularization penalty.

Exhibit 3. Classifying Assets as Stock- or Bond-Like Using a Support Vector Machine with Regularization Penalty, 1999–2025



Notes: Quarterly correlations of CPI YoY change and real GDP YoY growth with the prior quarter's YoY return assets. The SVM model with trade-off parameter $C = 1$ is trained on 32 equity indexes and 15 bond indexes and is used to classify 21 other assets (4 metals, 1 commodity index, 11 currencies, and 5 strategy indexes) as equity-like and bond-like.

Labeled points: US Large Cap: MSCI USA Gross Total Return USD Index; US Small Cap: MSCI USA Small Cap Gross Total Return USD Index; US High Div: MSCI USA IMI High Dividend Yield Gross Total Return USD Index; US Treasuries: Bloomberg US Treasury Total Return Index; US Inv Grade: Bloomberg US Corporate Investment Grade Total Return Index; US TIPS: Bloomberg US Treasury Inflation-Linked Total Return Index; US High Yield: Bloomberg US Corporate High Yield Total Return Index; Gold: XAUUSD Spot Exchange Rate - Price of 1 XAU in USD (US dollars per troy ounce); Silver: XAGUSD Spot Exchange Rate - Price of 1 XAG in USD (US dollars per troy ounce); Platinum: XPTUSD Spot Exchange Rate - Price of 1 XPT in USD (US dollars per troy ounce); Palladium: XPDUSD Spot Exchange Rate - Price of 1 XPD in USD (US dollars per troy ounce); Commodities: Bloomberg Commodity Index Total Return; Gold Miners: NYSE Arca Gold Miners Index; China Equities: MSCI China Gross Total Return Local USD Index; Nasdaq: Nasdaq-100 Total Return Index; Korea Equities: MSCI Emerging Korea Gross Total Return Local Index; Turkey Equities: MSCI Emerging Turkey Gross Total Return Local Index; US T-bills: Bloomberg US Treasury Bill Index; Commodity Producers: MSCI World Commodity Producers Gross Total Return USD Index; ZAR: ZARUSD Total Return Long - Long ZAR, Short USD; NZD: NZDUSD Total Return Long - Long NZD, Short USD; AUD: AUDUSD Total Return Long - Long AUD, Short USD; NOK: NOKUSD Total Return Long - Long NOK, Short USD; GBP: GBPUSD Total Return Long - Long GBP, Short USD; TRY: TRYUSD Total Return Long - Long TRY, Short USD; CNY: CNYUSD Total Return Long - Long CNY, Short USD; EUR: EURUSD Total Return Long - Long EUR, Short USD; CHF: CHFUSD Total Return Long - Long CHF, Short USD; JPY: JPYUSD Total Return Long - Long JPY, Short USD; US Dollar: US Dollar Index; SG CTA: SG CTA Index; SG Trend: SG Trend Index; FX Carry: Bloomberg GSAM FXCarry Index; DB Vol Carry: Deutsche Bank Volatility Carry (US Large Cap); DB CTA: DB Cross Asset CTA Trend Index.

Sources: Author's calculations for illustration and educational purposes only. Data are from Bloomberg [Bloomberg US Agg Total Return Index; Bloomberg US Corporate High Yield Total Return Index; Bloomberg 1-3 Yr Gov/Credit Total Return Index; Bloomberg US Treasury Total Return Index; Bloomberg 10+ Yr Gov/Credit Total Return Index; Bloomberg US Treasury Inflation-Linked Total Return Index; Bloomberg US Treasury Bill Index; Bloomberg US Mortgage Backed Securities Index; Bloomberg US Corporate Investment Grade Total Return Index; Bloomberg Long US Treasury with 10+Y Total Return Index; Bloomberg US Treasury Inflation Notes 10+Y Total Return Index; Bloomberg Global Agg - Australia Total Return Index Unhedged AUD; Bloomberg Global Agg - Canadian Total Return Index Unhedged CAD; Bloomberg Global Agg - Japanese Total Return Index Unhedged JPY; Bloomberg EuroAgg Index; MSCI USA Value Gross Total Return USD Index; MSCI USA Growth Gross Total Return USD Index; MSCI USA Gross Total Return USD Index; MSCI USA Quality Gross Total Return USD Index; Nasdaq-100 Total Return Index; MSCI USA Minimum Volatility Gross Total Return USD Index; S&P 500 Total Return Index; MSCI USA Small Cap Gross Total Return USD Index; MSCI USA IMI High Dividend Yield Gross Total Return USD Index; MSCI Australia Gross Total Return Local Index; MSCI Canada Gross Total Return Local Index; MSCI UK Gross Total Return Local Index; MSCI EU Gross Total Return Local Index; MSCI Japan Gross Total Return Local Index; MSCI EM Gross Total Return Local Index; MSCI Brazil Gross Total Return Local Index; MSCI EM BRIC Gross Total Return Local Index; MSCI China Gross Total Return Local USD Index; MSCI India Gross Total Return Local Index; MSCI Hong Kong Gross Total Return Local Index; MSCI Emerging Korea Gross Total Return Local Index; MSCI Emerging South Africa Gross Total Return Local Index; MSCI Emerging Turkey Gross Total Return Local Index; MSCI Emerging Taiwan Gross Total Return Local Index; S&P Global Natural Resources Total Return Index; MSCI World Energy Gross Total Return USD Index; MSCI World Commodity Producers Gross Total Return USD Index; MSCI World Commodity Producer Sector Capped Gross Total Return USD Index; NYSE Arca Gold Miners Index; Dow Jones Equity REIT Total Return Index; MSCI US REIT Gross Total Return Index; S&P Global Infrastructure Total Return Index; XAUUSD Spot Exchange Rate - Price of 1 XAU in USD (US Dollars per Troy Ounce); XAGUSD Spot Exchange Rate - Price of 1 XAG in USD (US Dollars per Troy Ounce); XPTUSD Spot Exchange Rate - Price of 1 XPT in USD (US Dollars per Troy Ounce); XPDUSD Spot Exchange Rate - Price of 1 XPD in USD (US Dollars per Troy Ounce); Bloomberg Commodity Index Total Return; ZARUSD Total Return Long - Long ZAR, Short USD; NZDUSD Total Return Long - Long NZD, Short USD; AUDUSD Total Return Long - Long AUD, Short USD; NOKUSD Total Return Long - Long NOK, Short USD; GBPUSD Total Return Long - Long GBP, Short USD; TRYUSD Total Return Long - Long TRY, Short USD; CNYUSD Total Return Long - Long CNY, Short USD; EURUSD Total Return Long - Long EUR, Short USD; CHFUSD Total Return Long - Long CHF, Short USD; JPYUSD Total Return Long - Long JPY, Short USD; US Dollar Index; SG CTA Index; SG Trend Index; Bloomberg GSAM FXCarry Index; Deutsche Bank Volatility Carry (US Large Cap); DB Cross Asset CTA Trend Index].

Support Vector Machine Applications in Investments

Support vector machines can be used for descriptive classification, prediction, and optimization. I showed a classification example of stocks versus bonds in the previous section.

Nazareth and Reddy (2023) conducted a large-scale review of literature on financial applications of machine learning. They found that SVM models are the most frequently studied, especially in insolvency and bankruptcy prediction, mainly because of their effectiveness in dealing with two-group classification problems. There are also many applications of SVMs in stock market and cryptocurrency studies.

Ryll and Seidens (2019) conducted a systematic meta-analysis of existing works on ML-based trading algorithms and found that SVMs significantly "outscore" some neural networks that have similar objectives in classification.

As with any artificial intelligence (AI) or machine learning (ML) applications in investing, a major risk is overfitting the SVM to the training data; see Simonian (2024) for a comprehensive model validation review.

Prediction

In this section, I discuss the application of SVM to predicting future asset returns and credit ratings. For example, an investor may want to use SVMs with the following objectives:

- Given an asset universe, select assets more likely to have higher expected returns or outperform a benchmark over a subsequent time period.
- Reconstruct corporate credit ratings with fundamental accounting variables.

Given an asset universe, SVMs can be used for separating assets into two groups: those likely to have higher expected returns and those likely to have lower expected returns. Portfolios can then be formed by going long assets likely to outperform and/or short assets likely to underperform.

Fan and Palaniswami (2001) used SVMs to predict which stocks trading on the Australian Stock Exchange would be likely to outperform the market. They considered 37 firm accounting indicators, grouped them into eight groups (return on capital, profitability, leverage, investment, growth, short-term liquidity, return on investment, and risk), and used principal component analysis for dimension reduction, obtaining eight-element input (feature) vectors prior to training, with each element representing a single extracted principal component from each group. Then, they applied SVM techniques to this eight-dimensional feature space in order to select the top-returning 25% of the stock returns every year, labeled as exceptional high-return stock (Class +1), while the others were labeled as unexceptional return (Class −1). The equally weighted portfolio of exceptional high-return stock formed by SVM had a total return of 207% over a five-year out-of-sample period, 1995–99, outperforming the benchmark return of 71%.

Kim (2003) used SVMs to predict the direction of change in the daily KOSPI, the South Korean composite stock price index, using 12 technical indicators (including momentum, price oscillator, and relative strength index) as features. The KOSPI data sample has 2,928 trading days, from January 1989 to December 1998. About 20% of the data were used for holdout and 80% for training. The study used both the polynomial kernel and the Gaussian radial basis kernel for SVM. Empirically, SVM outperformed the back-propagation neural network and case-based reasoning by 3.10% and 5.85%, respectively, for the out-of-sample ("holdout") data. Kim (2003, p. 318) concluded that "SVM provides a promising alternative for financial time-series forecasting."

Huerta, Corbacho, and Elkan (2013) used SVM to identify stocks whose volatility-adjusted price change falls within the highest or lowest quantile (e.g., the highest or lowest 25%). The highest-ranked stocks were used for long positions, and the lowest-ranked stocks were used for short sales. The data sample is the US equity universe available from the merged CRSP/Compustat database between 1981 and 2010. The study examined 44 fundamental and 7 technical features. Empirically, the constructed portfolios achieved annual returns of 15% (not counting transaction costs), with volatilities under 8%. The authors also discussed the process of choosing SVM meta-parameters to mitigate overfitting.

An interesting application to credit ratings is studied in Huang, Chen, Hsu, Chen, and Wu (2004). They examined two datasets: a Taiwanese dataset of 74 cases of bank credit ratings and 21 financial variables, which covered 25 financial institutions from 1998 to 2002, and a US dataset with 265 cases of bank credit ratings for 36 commercial banks from 1991 to 2000. Five rating categories appeared in the US dataset: AA, A, BBB, BB, and B. The authors developed several SVM-based models for credit ratings with accounting ratios' feature spaces of dimensions ranging from 7 to 21. SVMs achieved the best performance (compared with benchmark neural networks) in three of the four models tested, and SVM and neural network models outperformed the logistic regression model consistently.

Portfolio Construction and Optimization

In this section, I discuss applications of SVMs to portfolio optimization, such as

- using SVMs to preselect assets before portfolio construction,
- integrating SVMs with portfolio construction and optimization, and
- using SVMs for feature selection in portfolio construction.

Gupta, Mehlawat, and Mittal (2012) used SVMs for classifying financial assets in three predefined classes and then solving a portfolio selection problem incorporating investor preferences.

Silva, Felipe, de Andrade, da Silva, de Melo, and Tonelli (2024) found that preselecting Brazilian stocks using the SVM model and subsequently optimizing them by maximizing the Sharpe ratio resulted in a superior return and faster recovery after drawdown periods compared with the benchmark.

Islip, Kwon, and Kim (2025) proposed integrating SVMs and cardinality-constrained mean–variance optimization into one procedure (rather than preselecting assets before portfolio optimization), called SVM-MVO. Their joint selection of a portfolio and a separating asset-screening hyperplane optimized the trade-off between risk-adjusted returns, hyperplane margin, and classification errors that were made by the hyperplane. The authors observed that "SVM-MVO models are equivalent to regularization that penalizes portfolios with eligibility decisions that cannot be well explained by a low-dimensional hyperplane" (Islip et al. 2025, p. 1056). The study "demonstrates the effectiveness of the SVM-MVO models in constructing portfolios with lower risk, higher returns, and higher Sharpe ratios" (Islip et al. 2025, p. 1057). In particular, they outperformed their cardinality-constrained MVO counterpart during major financial events.

Integration of various ML techniques into the framework of Markowitz's portfolio selection is discussed in López de Prado, Simonian, Fabozzi, and Fabozzi (2025). In particular, the study found that SVMs can be helpful in feature selection by pinpointing variables most relevant to asset price movements.

The next chapter examines another widely used group of supervised learning tools known as *supervised ensemble methods*.

References

Awad, Mariette, and Rahul Khanna. 2015. "Support Vector Regression." In *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, 67–80. Berkeley, CA: Apress. doi:10.1007/978-1-4302-5990-9_4.

Cortes, Corrina, and Vladmir Vapnik. 1995. "Support-Vector Networks." *Machine Learning* 20 (3): 273–97. doi:10.1023/A:1022627411411.

Fan, A., and Marimuthu Palaniswami. 2001. "Stock Selection Using Support Vector Machines." *International Joint Conference on Neural Networks Proceedings (Cat. No.01CH37222)*, vol. 3: 1793–98. doi:10.1109/IJCNN.2001.938434.

Gupta, Pankaj, Mukesh Kumar Mehlawat, and Garima Mittal. 2012. "Asset Portfolio Optimization Using Support Vector Machines and Real-Coded Genetic Algorithm." *Journal of Global Optimization* 53: 297–315. doi:10.1007/s10898-011-9692-3.

Huang, Zan, Hsinchun Chen, Chia-Jung Hsu, Wun-Hwa Chen, and Soushan Wu. 2004. "Credit Rating Analysis with Support Vector Machines and Neural Networks: A Market Comparative Study." *Decision Support Systems* 37 (4): 543–58. doi:10.1016/S0167-9236(03)00086-1.

Huerta, Ramon, Fernando Corbacho, and Charles Elkan. 2013. "Nonlinear Support Vector Machines Can Systematically Identify Stocks with High and Low Future Returns." *Algorithmic Finance* 2 (1): 45–58. https://journals.sagepub.com/doi/abs/10.3233/AF-13016.

Islip, David, Roy H. Kwon, and Seongmoon Kim. 2025. "Integration of Support Vector Machines and Mean–Variance Optimization for Capital Allocation." *European Journal of Operational Research* 322 (3): 1045–58. doi:10.1016/j.ejor.2024.11.022.

Kim, Kyoung-jae. 2003. "Financial Time Series Forecasting Using Support Vector Machines." *Neurocomputing* 55 (1–2): 307–19. doi:10.1016/S0925-2312(03)00372-2.

López de Prado, Marcos, Joseph Simonian, Francesco A. Fabozzi, and Frank J. Fabozzi. 2025. "Enhancing Markowitz's Portfolio Selection Paradigm with Machine Learning." *Annals of Operations Research* 346: 319–40. doi:10.1007/s10479-024-06257-1.

Nazareth, Noella, and Yeruva Venkata Ramana Reddy. 2023. "Financial Applications of Machine Learning: A Literature Review." *Expert Systems with Applications* 219 (1 June). doi:10.1016/j.eswa.2023.119640.

Ryll, Lukas, and Sebastian Seidens. 2019. "Evaluating the Performance of Machine Learning Algorithms in Financial Market Forecasting: A Comprehensive Survey." Working paper (6 July). https://doi.org/10.48550/arXiv.1906.07786.

Silva, Natan Felipe, Lélis Pedro de Andrade, Washington Santos da Silva, Maísa Kely de Melo, and Adriano Olímpio Tonelli. 2024. "Portfolio Optimization Based on the Pre-selection of Stocks by the Support Vector Machine Model." *Finance Research Letters* 61 (March). doi:10.1016/j.frl.2024.105014.

Simonian, Joseph. 2024. *Investment Model Validation: A Guide for Practitioners*. Charlottesville, VA: CFA Institute Research Foundation. doi:10.56227/24.1.15.

Smola, Alex J., and Bernhard Schölkopf. 2004. "A Tutorial on Support Vector Regression." *Statistics and Computing* 14 (3): 199–222. doi:10.1023/B:STCO.0000035301.49549.88.

Vapnik, Vladmir N. 1999. "An Overview of Statistical Learning Theory." *IEEE Transactions on Neural Networks* 10 (5): 988–99. doi:10.1109/72.788640.

Vapnik, Vladmir N. 2000. *The Nature of Statistical Learning Theory*, 2nd ed. New York: Springer-Verlag.

Vapnik, Vladmir N., Steven Golowich, and Alex Smola. 1997. "Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing." In *Advances in Neural Information Processing Systems 9 (NIPS 1996)*, edited by M. Mozer, M. Jordan, and T. Petsche, 281–87. Cambridge, MA: MIT Press.

ENSEMBLE LEARNING IN INVESTMENT: AN OVERVIEW

Alireza Yazdani, PhD
Senior Vice President, Data Science and AI, Investor and Issuer Services, Citi

Introduction

The use of quantitative methods in finance and investment applications is not a new phenomenon, particularly among more sophisticated practitioners. This usage spans a wide range of methods, including but not limited to probabilistic, statistical, and stochastic models; univariate and multivariate time-series techniques; discrete and continuous simulations; linear and nonlinear programming; dynamic programming; and mathematical optimization. Notwithstanding the great success of many such quantitative methods, they have known limitations, including the parametric or prespecified functional forms that may lack flexibility, theoretical data distribution assumptions that are regularly violated, and computational challenges associated with the so-called curse of dimensionality. An ever-growing list of federated datasets, alongside the development of tools that transform such data into useful variables for explanatory and predictive tasks, calls for a paradigm shift in how financial data are used and processed.

In recent years, financial data science and machine learning (ML) methods have been widely adopted as prediction and decision-support means in various investment applications. The following are some of the contributing factors to such widespread adoption:

- Numerical power and flexibility of ML in dealing with unstructured, private, and microlevel data beyond the traditional econometric methods (López de Prado 2019)
- Scalability of ML in distilling large datasets consisting of many variables (factor zoo) into subsets that reliably predict cross-sectional variations in stock returns (Gu, Kelly, and Xiu 2020)
- The efficacy of the ML models in predicting cross-section of stock returns, compared with ordinary least-squares (OLS) regression, attributable to their ability to uncover nonlinear patterns and robustness to multicollinear predictors (Gu et al. 2020)
- The possibility of demystifying (i.e., breaking down and explaining) ML predictions into linear, nonlinear, and interactive components (Li, Turkington, and Yazdani 2020)

Simonian and Fabozzi (2019) argued that financial data science is to be regarded as a discipline in its own right and not a mere application of data science methods to finance or a branch of classical econometrics.

One major branch of machine learning is supervised learning, where available data are used to train a model that is to establish a mapping between input variables (also known as features or predictors) and output variables (also known as the labels, the response, or the dependent variable). If the response variable is continuous numeric, the model is called a regression, and if the response is categorical, the model is called a classification. This mapping is then used for inference and scoring, to explain the relationship between the input and response, or ideally to make predictions about the new values of the response variable given the new input.

Among the widely used supervised learning models are ensembles, which are based on the presumption of the wisdom of crowds. In this view, combining diverse opinions from a committee of several individuals can lead to greater wisdom. In other words, combining multiple predictions obtained from numerous somewhat independent methods often leads to better predictions than identifying a single best prediction. In a regression, this combining can be simple averaging or other sophisticated weighting mechanisms, whereas in a classification, majority voting or other weighting mechanisms may be used. Technically, the question is how effectively an ensemble model can decompose bias and variance terms, to be able to more generally work well on previously unseen test data not used for model training. Recall that the expected model prediction error for a regression model, as measured by the mean squared error (MSE), is formulated as

$$\text{MSE} = E[(\hat{y} - y)^2] = E[(\hat{y} - E[\hat{y}])^2] - (E[\hat{y}] - y)^2.$$

This relationship can be interpreted as:

$$\text{MSE} = \text{Variance} + \text{Bias}^2.$$

Here, bias refers to the error committed by the learning algorithm, where higher values may indicate underfitting as a result of the model failing to adequately establish the relationships between features and output. Variance refers to the error from the sensitivity of the model to another subset of training data, where higher values indicate overfitting. Generally, the more complex the model, the lower the bias and the higher the variance will be. Instances of model complexity include using many features, a greater number of parameters, and complex model design or architecture. The bias–variance trade-off concerns minimizing test error by finding the right model complexity.

Before delving into the various categories and the qualities of ensemble models, it is worth mentioning that the idea of combining predictions is not an entirely new one, and it has previously been studied by statisticians and econometricians, particularly in the context of time-series forecasting. In a seminal work, Bates and Granger (1969, pp. 451–68) observed that "combined forecasts yield lower mean-square error than either of the original forecasts." Over time, several combination methods have been developed, including simple averaging, time-weighted and performance-weighted estimations, nonlinear combination, and combining by learning (Wang, Hyndman, Li, and Kang 2023). Combining is directly related to a special case of ensemble learning known as stacking, which I will discuss later. By aggregating predictions, ensemble models reduce the impact of errors and noise that may exist in individual predictions, enabling them to achieve greater overall accuracy, reliability, and stability.

Ensemble Methods

In principle, ensemble methods share two main components: a diverse set of predictive "base learner" models trained on sampled subsets of training data and a combination mechanism, such as averaging or voting, used to aggregate the predictions from these base learners (Flach 2012). Broadly speaking, ensemble methods fall into one of three categories: bagging, boosting, and meta-learning. In this section, we review these categories, as well as some properties and possible drawbacks of ensemble learning methods.

Bagging

Short for "bootstrapped aggregating," bagging is an instance of ensemble averaging where a diverse group of base models is developed and combined with the end goal of improving

the stability and accuracy of classification and regression algorithms. These base models are built independently through repeatedly bootstrapping (i.e., random sampling with replacement) from the training data (see **Exhibit 1**). Bagging is particularly effective because of the possibility of the calculation of so-called out-of-bag error: the average of the errors from each independently developed model evaluated on the unselected subset of training data during the bootstrapping. Out-of-bag error estimation is an effective mechanism similar to cross-validation, leading to better model calibration and overall improvement of predictions.

Perhaps the best-known example of bagging in machine learning is the *random forests* algorithm (Breiman 2001). Random forests are generated by building a large collection of de-correlated and possibly deep decision trees and then combining them through averaging in regression or majority voting in classification (see **Exhibit 2**). Decision trees learn by recursive partitioning of the feature space and classifying members of a population by separating

Exhibit 1. A Bootstrapped Aggregation Ensemble

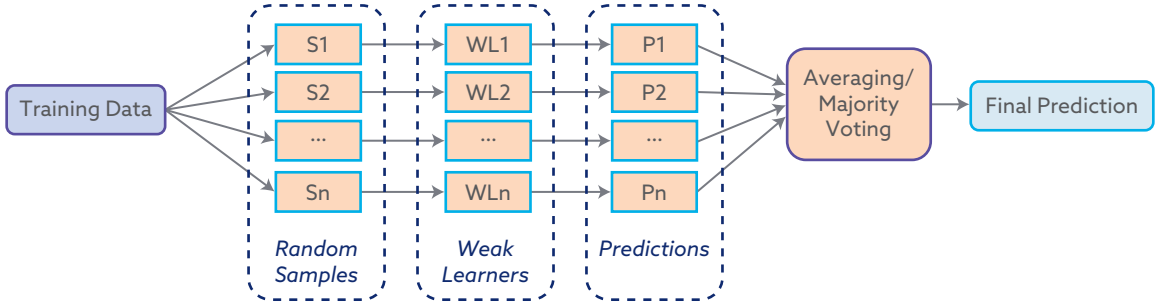
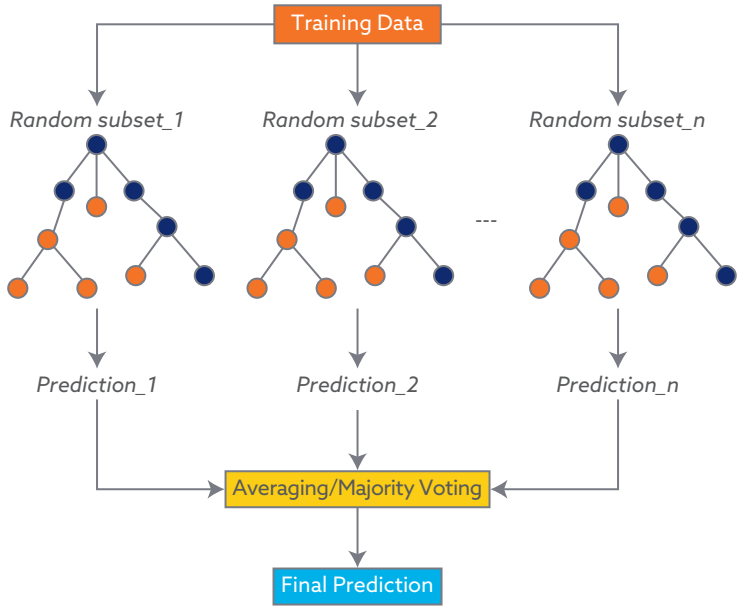


Exhibit 2. A Random Forest Model



it into subpopulations based on various dichotomous independent variables. Decision trees have desirable properties, including automatic variable selection, handling mixed features and missing values, and scalability. The depth and breadth of decision trees enables them to capture local patterns in training data and reduce bias. Because these trees are identically distributed, an ensemble consisting of the average of those trees has the same bias as the individual trees. If these trees are grown on the random subsets of data and input variables, their pairwise correlation is reduced to the extent that the bagged ensemble consisting of a sufficiently large number of trees can achieve an overall lower variance (Hastie, Tibshirani, and Friedman 2017).

Technically, for m independently and identically distributed decision tree predictions with variance σ^2, their average has a variance of $\frac{\sigma^2}{m}$. When predictions are not independent yet their maximum pairwise correlation is bounded by ρ, the variance of their average is bounded by $\rho\sigma^2 + \frac{(1-\rho)\sigma^2}{m}$. In other words, a random selection of input variables will reduce the maximum pairwise correlation, ρ (hence reducing the first term), and an increase in the number of decision trees, m, will reduce the second term, resulting in overall reduction of the variance. For an elegant exposition and technical discussion on several properties of random forests, see Breiman (2001). The implementation of ensemble-based methods for classification, regression, and anomaly detection is straightforward in practice. For example, in Python, many such models can be created using, among others, the scikit-learn library. The following is an example code for creating the random forest classification model in Python:

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split

#create a synthetic dataset

from sklearn.datasets import make_classification

X, y = make_classification(n_samples=1000, n_features=10, n_informative=6,
random_state=123)

#Split the dataset into 80% training and 20% testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)

#train a Random Forest model and predict

model = RandomForestClassifier(random_state=123)

model.fit(X_train, y_train)

model.predict(X_test)

#mean accuracy

model.score(X_test,y_test)
```

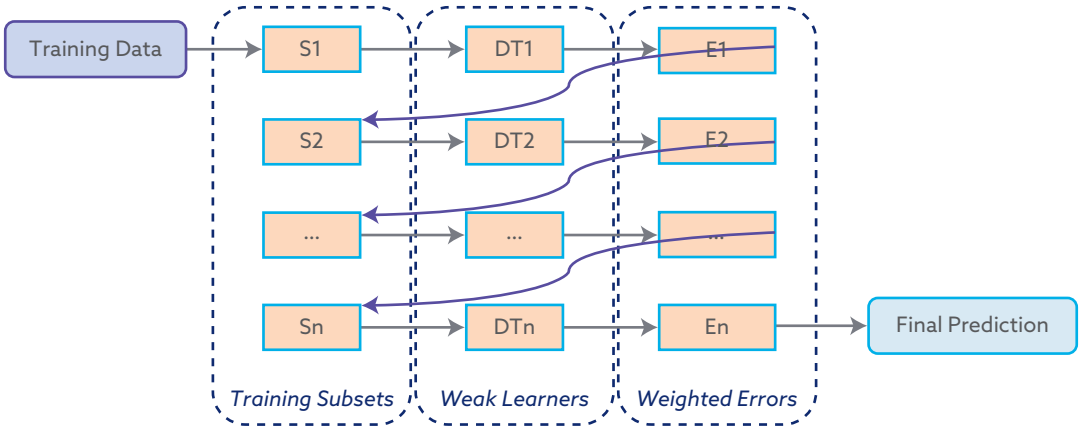
Boosting

Boosting is similar to bagging in that it benefits from data sampling and variable sampling, as well as in the property of combining several base learners. In boosting, however, unlike bagging, the basic mechanism is bias reduction, particularly using shallow trees. In addition, base learners are trained sequentially to correct the errors of the previous learners. The original boosting classifier, AdaBoost, proposed by Freund and Schapire (1996), works by sequentially training weak classifiers that focus on the mistakes of the previous ones by assigning higher weights to misclassified samples. The final prediction is a weighted sum of the individual learners' outputs (see **Exhibit 3**). Notwithstanding its superior predictive accuracy, the AdaBoost method is less readily interpretable than decision trees and requires a longer training time.

The *gradient boosted model* (GBM) proposed by Friedman (2001) has proved to be a popular alternative boosting method because it mitigates some of AdaBoost's problems. The GBM works by sequentially adding new tree predictors and trying to fit the new predictor to the residual errors by the previous tree, which gradually improves predictions as new trees are added (Géron 2019). Notably, recent advancements in boosting algorithms—for example, XGBoost (Chen and Guestrin 2016) and LightGBM (Ke, Meng, Finely, Wang, Chen, Ma, Ye, and Liu 2017)—focus on regularization, adaptability, efficiency, and scalability, while maintaining high accuracy. The following is an example code for creating the gradient boosted regression model in Python:

```
from sklearn.datasets import make_regression

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.model_selection import train_test_split

#create a synthetic dataset

X, y = make_regression(n_samples=1000, n_features=10, n_informative=6,
random_state=123)

#Split the dataset into 80% training and 20% testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)

#train a model, predict and score

model = GradientBoostingRegressor(random_state=123)

model.fit(X_train, y_train)

model.predict(X_test)

#the coefficient of determination R-squared

model.score(X_test, y_test)
```

Exhibit 3. Example Boosting Mechanism Consisting of Decision Tree Base Learners



Meta-Learning

Meta-learning involves training a set of base models and also training a meta-model to combine them in order to achieve the greatest bias–variance trade-offs. For example, using as new input features the predictions from several diverse base learners (e.g., consisting of a linear model, multiple trees, and a neural network), a linear meta-model can be used to learn the weights to best combine these predictions (Flach 2012).

Alternatively, a logistic regression, a decision tree, or an even more complex model can be used as a meta-model by following the same idea that predictions from base learners are input features to the meta-model (as shown in Exhibit 3). This approach is also known as *stacked generalization* (Wolpert 1992) or simply *model stacking*, as shown in **Exhibit 4**. Stacking usually involves estimating weights for each base learner using the least squares method. These weights can be restricted to values greater than zero and with a total sum of one that solve the associated quadratic programming. This approach has the advantage that estimated weights can be interpreted as probability values and also avoids giving too much weight to base models with high complexity. One key consideration while training meta-models is to use a validation sample that is different from the data used in training base learners, so as to avoid propagating any biases in the base learners.

The following is an example Python code for creating a stacked classifier by using a random forest and a support vector classifier as the base models and logistic regression as the meta-learner:

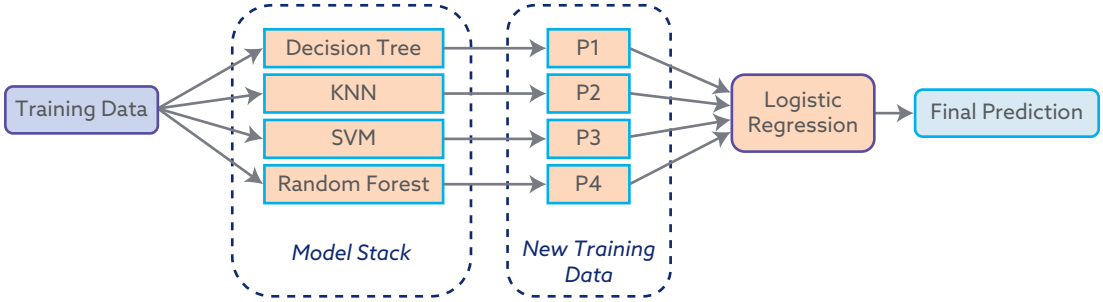
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.ensemble import StackingClassifier
#create a synthetic dataset
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=15, n_informative=6,
random_state=123)
#Split the dataset into 80% training and 20% testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
#train a meta-model and predict
estimators = [('rf', RandomForestClassifier(n_estimators=100, random_state=123)),
('svr', make_pipeline(StandardScaler(),
LinearSVC(random_state=123)))]
clf = StackingClassifier(estimators=estimators, final_estimator=LogisticRegression())
clf.fit(X_train, y_train)
score(X_test, y_test)
```

Properties of Ensemble Learning Methods

Before listing major instances of the financial applications of ensemble models, I want to highlight important features and mechanisms associated with such models that are key to their widespread adoption:

- The bootstrapped aggregation or iterative correction mechanisms described previously imply that ensemble models may achieve optimal bias–variance trade-offs, reduced risk of overfitting, and improved robustness to noise and outliers. These features make ensembles

Exhibit 4. Example Model Stacking, Where Base Learners' Predictions Are Passed Through a Meta-Learner



a powerful choice for predictive tasks that require good out-of-sample generalizations on high-variance, low-signal-to-noise financial datasets.

- Ensemble models scale well with large datasets, making them suitable for big data in finance. In particular, they can handle various independent or interdependent variables, including firm fundamentals, macroeconomic factors, and market sentiments, and focus on the most relevant variables that generate the strongest signals.
- Financial systems are complex, nonlinear, and ruled by different market regimes and economic cycles. They consist of numerous variables that may behave and interact unexpectedly under volatile and abnormal circumstances. Heterogeneous ensemble methods, particularly regression trees, combine models of different types to capture multi-way predictor interactions, allowing flexibility in capturing various patterns in financial data and subsequently adapting to nonlinear and short-lived dynamics.
- Ensemble models can easily accommodate many data preprocessing and adjustment strategies (such as balanced bagging or weighted boosting) to handle rare and imbalanced data, often encountered in fraud detection, credit default prediction, economic recession, and extreme risk events.
- Explainability techniques for tree-based ensembles may be used to make these models more transparent by explaining their predictions. For example, feature importance scores identify influential predictors, and Shapley values (Shapley 1953) provide local attribution analysis of the predictions. These techniques greatly aid in understanding the decision-making processes.

Ensemble Learning Challenges

Despite strong predictive performance, using ensemble learning may come with certain challenges, although arguably not as much as with some other machine learning and deep learning techniques. First, ensemble models may be considered black boxes without using interpretability tools, making it a challenge to explain predictions to nontechnical stakeholders and reducing trust in critical applications. In addition, training and predicting with ensemble models often require significant computational resources, particularly for large datasets or complex models. As such, compared with simpler models, deploying ensemble models in production

environments can be more challenging because of their larger size, more complex pipelines, and extra computational demands.

In particular, gradient boosting algorithms can be slow to train because of their sequential tree-building process and the complexity of the hyperparameter tuning process. Without adequate calibration, either overfitting or underfitting is possible. Finally, following the "no free lunch" theorem (Wolpert 1996), which states that there is no best single machine learning algorithm for all possible prediction problems, one could argue that there are numerous real-world datasets and prediction problems for which ensemble models will not be the top performers. In such circumstances, other models and algorithms are required, depending on the "speed–accuracy–complexity trade-offs" (Wolpert 2002).

Applications of Ensemble Learning in Finance

A quick survey of the literature published in the last few years points to several financial applications of ensemble learning, usually with a considerable level of success compared with classical methods. It is not possible to review or validate all the instances listed in the literature, but I will highlight a small subset of such applications from different categories, which should provide a good perspective on the variety and comparative performance of ensemble models. These highlighted examples can be grouped into the following (loosely defined, somewhat overlapping) three categories: portfolio management, volatility forecasting and option pricing, and miscellaneous applications.

Portfolio Management

Portfolio management encompasses various activities related to investment selection, portfolio construction, and asset allocation optimization in order to meet the short- and long-term financial objectives and risk tolerance of an investor or an institution. This data-intensive process typically involves analyzing factors and historical returns, estimating model parameters, forecasting risk premium or conditional expected stock returns in excess of the risk-free rate, and occasionally estimating portfolio weights.

Cross-Section of Stock Returns

Asset return prediction and the identification of predictors of returns are among the fundamental topics in asset pricing research. Although traditional factor models—such as the CAPM and its extensions, including the three-factor and five-factor Fama–French models—are widely used to explain historical returns of diversified portfolios, practitioners consider them empirically inadequate to predict returns under various market regimes. Subsequently, a plethora of factors (the "factor zoo") have been proposed as candidate predictors of cross-sectional variations in expected returns.

Among the various groups of such factors are financial, macro, microstructure, behavioral, and accounting factors, many of which may be viewed as belonging to the broader categories of common sources of risk or idiosyncratic characteristics that pertain to an individual firm or portfolio (Harvey, Liu, and Zhu 2016). Naturally, one may ask which one of these factors provides effective and independent information about the cross-section of returns. Here, the emphasis is placed on efficacy and independence within the context of the inadequacy of simple linear models in out-of-sample predictions in the presence of multicollinearity and noise.

Literature reviews point to the growing adoption of machine learning methods in empirical asset pricing to address the problem of the factor zoo (Bagnara 2024). These methods primarily include regularization and dimension reduction as processing steps, followed by the use of random forests, neural networks, and comparative analyses that typically involve ensemble methods. Among the important rationales stated for using machine learning in asset pricing are "return prediction is economically meaningful" and "relative to traditional empirical methods in asset pricing, machine learning accommodates a far more expansive list of potential predictor variables and richer specifications of functional form" (Gu et al. 2020, p. 2224).

Ensemble methods, in particular, have been found to provide superior results over traditional linear factor models in reducing the risk of overfitting, thanks to the forecast combination mechanism. Among the various instances of forecast combination mechanisms are combining forecasts from different classes of algorithms, combining forecasts based on different training windows, combining forecasts that use different factor libraries, and combining forecasts for different horizons to increase forecast diversity and reduce the risk of overfitting (Rasekhschaffe and Jones 2019).

It is important to point out that the superiority of a method is evaluated by quantifying its performance over a "testing" subsample that has been held out of the training sample (used for estimation) and the validation sample (used for model tuning). These testing data are truly out of sample and may reflect a method's predictive performance more realistically. The metrics used for quantifying model performance include the following:

- Standard algorithm performance evaluation metrics, such as the coefficient of determination, R^2; root mean square error; and mean absolute percentage error
- Aggregate portfolio performance metrics, such as mean and standard deviation of returns, cumulative return, maximum drawdown, and information ratio for the risk-adjusted return related to a benchmark

The algorithm performance evaluation metrics are estimated on the basis of a direct comparison of the realized versus predicted returns of the testing period. In contrast, aggregate portfolio performance metrics may be estimated on the basis of out-of-sample (e.g., one-month-ahead) stock return predictions of machine learning, going long on the top decile of stocks with the highest predicted return stocks, going short on the bottom decile, and calculating the backtest returns. Following Gu et al. (2020), the next-period expected return $r_{i,t+1}$ of stock i can be modeled as

$$\begin{cases} r_{i,t+1} = E_t(r_{i,t+1}) + \varepsilon_{i,t+1} \\ \text{s.t. } E_t(r_{i,t+1}) = L\left(X_{i,t}; \bar{h}\right), \end{cases}$$

where L is a (machine) learning algorithm acting on the p-dimensional vector, $X_{i,t}$ represents current-period features (factors), and $\bar{h}$ is the algorithm's tunable hyperparameters—for example, the number of trees or the maximum tree depth in a random forest. The OLS regression is a special case of this equality when L is a linear unbiased estimator for which the error terms, $\varepsilon_{i,t+1}$, have zero mean and constant variance and are uncorrelated.

The OLS estimator is an optimal model among linear unbiased estimators because of its lowest variance and is traditionally a popular choice. With a large set of candidate predictors, however, estimating an OLS regression may pose computational challenges and yield unreliable

predictions. First, a linear model is restricted in its functional form and may not detect interactions among features and their nonlinear dynamics. Second, when the number of features increases, there is an elevated risk of in-sample spurious correlation and overfitting. In addition, multicollinearity may be strong in the presence of many features, resulting in unreliable parameter estimates, because of high correlation or linear dependence among regressors.

Machine learning models are desirable when traditional linear models are not feasible, because they can be used for high-dimensional feature sets containing both cross-sectional firm characteristics and macroeconomic factors. Additionally, they can be regularized to reduce overfitting, and they efficiently search and select among many model specifications.

This notion has been demonstrated by Gu et al. (2020). The authors carried out a large-scale study on monthly total individual equity return data spanning 60 years, 1957–2016 (with the latest 30 years as out-of-sample testing). The studied data contain more than 30,000 stocks (6,000+ per month) for firms listed on the NYSE, AMEX, and NASDAQ. The Treasury bill rate was used for the risk-free rate to calculate individual excess returns, and for stock-level predictors, the authors used more than 90 characteristics based on the cross-section of stock returns, as well as industry dummies for Standard Industrial Classification codes and several macroeconomic predictors.

Gu et al. (2020) found that shallow neural networks and ensembles of regression trees are the best-performing methods across a variety of portfolio constructions (see **Exhibit 5** and **Exhibit 6**). They attributed this performance to the detection of nonlinear interactions missed by other methods and effective "shallow" learning (compared with "deep" learning) resulting from the scarcity of data and the low signal-to-noise ratio in asset pricing. The authors also reported greater success from ML in forecasting larger and more liquid stock returns and portfolios and observed agreement among ML models on a small set of dominant predictive signals, including price trends (return reversal and momentum), measures of stock liquidity, stock volatility, and valuation ratios.

Risk Factor Analysis

Machine learning can enrich risk analysis by providing insight into relationships between variables that are unaccounted for in more-traditional factor models. Several categories of "factor model" exist, including macroeconomic, statistical, and fundamental models (Connor 1995). In such models, the security return is the dependent variable, and the risk factors,

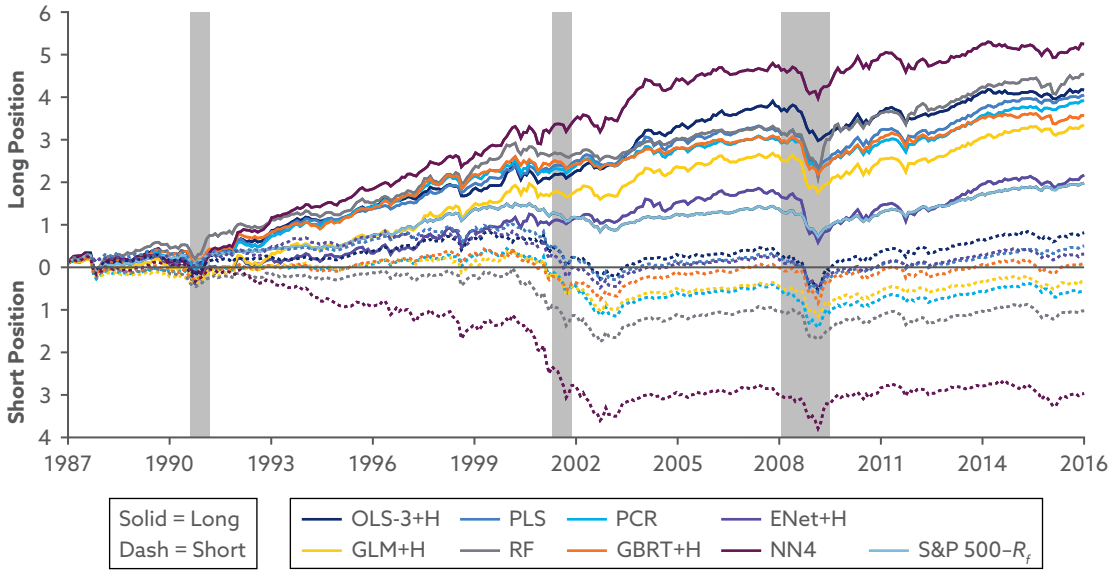
Exhibit 5. Performance of ML Portfolios

	OLS	Elastic Net	RF	GBRT	NN4
Mean return	1.34	2.11	2.38	2.14	3.33
Maximum drawdown percentage	84.74	33.70	46.42	37.19	14.72
Information ratio	1.30	0.93	1.30	1.60	2.40

Note: RF stands for random forest, GBRT stands for gradient boosted regression tree, and NN4 stands for four-hidden-layer neural network.

Source: Excerpts from Table 8 of Gu et al. (2020), used with permission.

Exhibit 6. Cumulative Log Returns of ML Portfolios Sorted on Out-of-Sample Forecasts (National Bureau of Economic Research recessions are shaded)



Source: Figure 9 of Gu et al. (2020), used with permission.

including macroeconomic variables and security characteristics, are the independent variables. Traditionally, estimation methods are based on time-series and/or cross-sectional regression, with OLS regression being the prevalent approach. The main assumptions of these models are that the selected group of independent variables adequately explains portfolio behavior and the factors have a low correlation for the model to be stable and statistically valid.

These assumptions may be easily violated under different market regimes, however, posing challenges to the estimation of the model parameters and explanation of the results. For example, as the dimensionality of the list of candidate factors grows to account for additional exposure, exacerbated by dramatic changes in factor relationships during major market events, both linear and parametric nonlinear models, which often heavily depend on sample data, become hard to estimate and are rendered inadequate and unstable.

As a remedy to these drawbacks, Simonian, Wu, Itano, and Narayanam (2019) developed a machine learning framework to uncover nonlinear relationships, discontinuities (e.g., threshold correlations), and interaction effects between factors in the well-known Fama–French–Carhart (FFC) equity factor model. These authors showed how the RF model provides a higher level of explanatory power relative to more commonly used frameworks, such as CAPM and FFC. They also provided useful information regarding the sensitivity of assets to factors by leveraging the importance of features to derive pseudo-betas. Additionally, the authors combined this approach with association rule learning to build a sector-rotation trading strategy. They showed that their strategy, using the RF-predicted return of a sector and the ratio of

short-term to long-term realized volatility as trading signals, outperforms the "no information" equal-weight portfolio, as measured by various out-of-sample performance metrics. The authors argue that this approach provides an effective way to inform both risk analysis and portfolio management.

Volatility Forecasting and Option Pricing

Quantitative finance, also known as financial mathematics, refers to the advanced mathematical models used for the pricing of derivative securities and portfolio risk management. A subcategory of quantitative finance is financial engineering, which uses computational simulations for pricing, trading, hedging, and other activities. This is a data-intensive process in which machine learning can be readily used.

Volatility Forecasting

An accurate measurement of volatility is of paramount importance in financial risk management, asset pricing, derivative pricing, and estimating expected returns of portfolios. Traditionally, volatility estimation models have relied on a handful of statistical inference methods and a limited group of predictors. In light of the availability of high-frequency price data, growing sets of candidate variables, and scalable computing power, machine learning methods enable researchers and practitioners to move beyond single-threaded forecasting methods typically dominated by a small set of particular features or algorithms and to develop scalable systems of volatility prediction.

Such systems can offer reduced human intervention during the selection of features and models, accommodating an expanded set of features and controlling overfitting through automated calibration. Notably, Li and Tang (2024) recently proposed an automated volatility forecasting system for the S&P 100 universe that leverages more than 100 features and five ML algorithms to predict the response variable of the realized variance (RV), an estimator of the quadratic variation of the log price process over a given period, using a set of features that include multiple realized features based on RV-based models, such as heterogeneous autoregressive (HAR), semivariance HAR (SHAR), and mixed data sampling (MIDAS).

In particular, after evaluating the out-of-sample performance of five learning algorithms—LASSO (least absolute shrinkage and selection operator), principal component regression (PCR), random forest (RF), gradient boosted regression trees (GBRT), and two-hidden-layer neural network (NN2)—the authors also constructed a simple average ensemble to combine all machine learning algorithms. The reported ensemble delivered superior performance across various forecast horizons; the automated volatility forecasting system acts dynamically and becomes increasingly powerful over time (**Exhibit 7**). For example, "in terms of utility gains, the ensemble model delivers 44 more bps per year to the mean–variance investor relative to using OLS for this large stock universe" (Li and Tang 2024, pp. 82–83).

Option Pricing

The motivation to apply machine learning methods for pricing options and derivatives is driven by the computational estimation challenges and limitations of classical parametric models. By considering options as functional mappings between the contracted terms' inputs (e.g., underlying asset and time to maturity) and the premium output (e.g., a fixed deviation

Exhibit 7. Out-of-Sample R^2 Relative to the Historical Mean of Realized Volatilities for OLS-Based Models

Model	Out-of-Sample R^2 Relative to Long-Run Mean			
	Daily	Weekly	Monthly	Quarterly
HAR	0.58	0.69	0.70	0.64
MIDAS	0.58	0.71	0.71	0.64
SHAR	0.58	0.70	0.70	0.64
OLS	0.61	0.73	0.72	0.63
LASSO	0.61	0.73	0.73	0.65
PCR	0.60	0.71	0.72	0.67
RF	0.59	0.71	0.73	0.66
GBRT	0.60	0.73	0.73	0.66
NN2	0.62	0.75	0.74	0.65
Ensemble	0.62	0.74	0.75	0.67

Source: Excerpts from Tables A.3 and A.4 of Li and Tang (2024), used with permission.

from a target price), data-driven and machine learning methods can be applied (Hutchinson, Lo, and Poggio 1994). In a study of European call options with West Texas intermediate (WTI) crude oil futures contracts as an underlying asset (Ivaşcu 2021), machine learning algorithms, particularly ensemble methods, such as XGBoost and LightGBM, largely outperformed classic methods, such as Black–Scholes and Corrado–Su, with both historical and implied volatility parameters.

Bid–Ask Spread

The foreign exchange (FX) market is both the largest and the most liquid financial market in the world. In particular, because of the uneven spread of liquidity across different currencies and different market times and conditions, predicting the likely cost of trading currencies as measured by bid–ask spreads is crucial for profitable trading. As such, the FX market is a fertile playground for data-driven and ML methods seeking to identify and benefit from market microstructures.

Recently, Kirkby and Andrean (2024) identified various predictors of daily bid–ask spreads, including temporal features, such as the day of the week and the hour of day and their interactions; various spreads, lags, and moving averages; and rate volatility, capturing the interplay between market volatility and bid–ask spreads and the spillover of volatility between related markets. They observed that from among supervised machine learning algorithms, subject to a modest level of hyperparameter tuning, the random forest model had the best out-of-sample performance, with an economically meaningful ability to forecast the next day's spread with a relative error of less than 20% on average.

Miscellaneous Applications

In addition to the previous examples, other applications of ensemble learning published in the financial data science literature include the following:

- Default prediction of commercial real estate properties and identification of the ratio of the capitalization rate spread to the average capitalization rate spread of property type as a top driver of defaults (Cowden, Fabozzi, and Nazemi 2019)
- Predictability of stock market bubbles and large near-term drawdowns based on patterns in stock price behavior (Simonian 2022)
- Prediction and estimation of the likelihood of recession in a given month using historical macroeconomic factor time series (Yazdani 2020)
- Comparing long short-term memory and gradient boosting algorithms in predicting the returns of S&P 500 constituents using technical indicators and principal component analysis (Cvetkovic 2024)

This list provides only a sample of the published literature on ensemble modeling, and many more may be identified and explored.

Explainability of Ensemble Methods

Broadly speaking, explainability, also known as interpretability, refers to processes used to understand how a machine learning model makes decisions and what its output means. Explainability is a way to increase reliability in the presence of uncertainty and reduce the risk of overfitting and irrelevant and unfair decision making. For a comprehensive study of explainability methods, see Molnar (2019). The following are some common explainability techniques:

- "Partial dependency plots show the marginal effect of an input feature on the predicted outcome."[1]
- The SHAP (SHapley Additive exPlanations) technique enables estimation and visualization of each input feature's contribution to the output (based on "Shapley values [that] measure the average marginal contribution of a feature across all possible combinations of features)."[2]
- Feature importance can be used to estimate the importance of a feature for the model, based on measuring the performance decrease when randomly shuffling the feature value across all samples.
- The LIME (local interpretable model-agnostic explanations) technique can be used to locally approximate a model's outputs with a simpler, interpretable model.
- Counterfactual explanations describe the smallest change to the feature values that change the prediction to a predefined outcome.
- Other methods—such as conformal prediction intervals (Vovk, Gammerman, and Shafer 2022)—that provide reliable estimates, even in highly unpredictable markets, are crucial for risk management and decision making under uncertainty.

---

[1] https://en.wikipedia.org/wiki?curid=54575571.

[2] https://en.wikipedia.org/wiki?curid=54575571.

In line with applications of ensemble models for financial forecasting and predictive modeling, explainability is used increasingly to demystify the prediction by such models. For example, Li et al. (2020) used ML to predict one-month spot returns of a portfolio of major currencies and subsequently developed a framework for demystifying ML model predictions called the "model fingerprints." This approach draws on partial dependence to identify feature interactions and estimate the average marginal effect when features have low correlations.

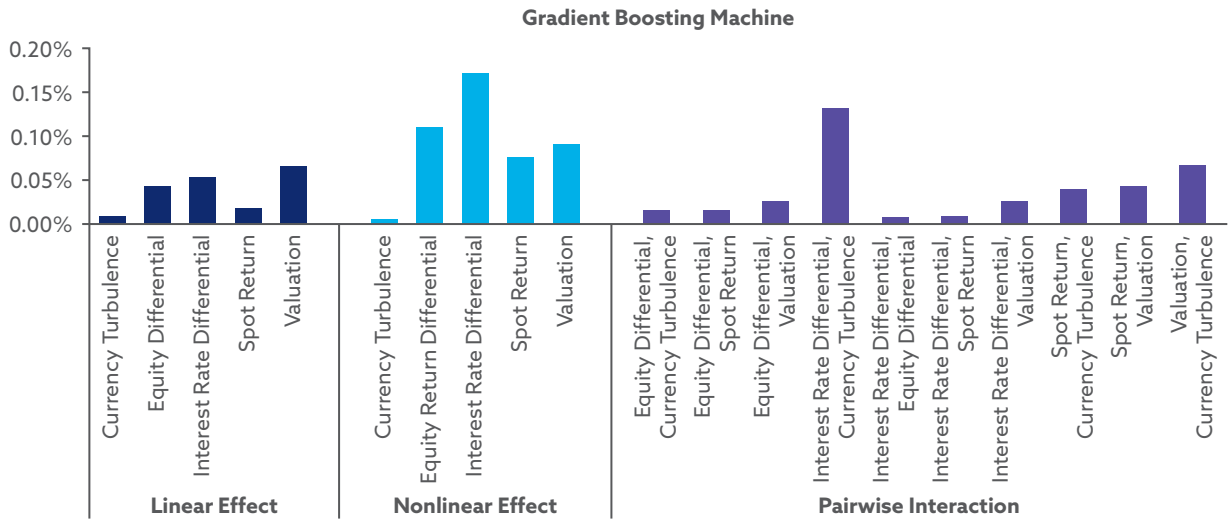
In particular, the model fingerprint decomposes predictions into the contributions from the following four sources: linear effect (*L*) for the time series returns of portfolios formed from linear predictions in isolation; pairwise interactions (*P*) for returns of portfolios formed from the combined linear and pairwise interaction predictions minus the returns from linear effect; nonlinear effect (*N*) for returns of portfolios from the combined linear, nonlinear, and pairwise interaction predictions minus the returns from linear and pairwise effects; and higher-order effects (*H*). Mathematically, for a model prediction function $\hat{y} = \hat{f}(x_1, \ldots, x_m)$ dependent on *m* input variables, this decomposition can be written as

$$\hat{y} = \sum_{i=1}^{m} L(x_i) + \sum_{i=1}^{m} N(x_i) + \sum_{i \neq j}^{m} P(x_i, x_j) + \sum_{i=1}^{m} H(x_i).$$

An example of such decomposition for the gradient boosting model prediction is shown in **Exhibit 8**. The model fingerprint method, when accompanied by other local and global model interpretability tools such as feature importance and SHAP force plots for attribution analysis, yields explanations for the model outcome, which is useful for comparing and interpreting various investment strategies.

Among other notable examples of ML model explainability is the framework developed by Davis, Lo, Mishra, Nourian, Singh, Wu, and Zhang (2023) in which the authors used ML models

Exhibit 8. Decomposition of Gradient Boosting Model Predictions Using the Fingerprint Method



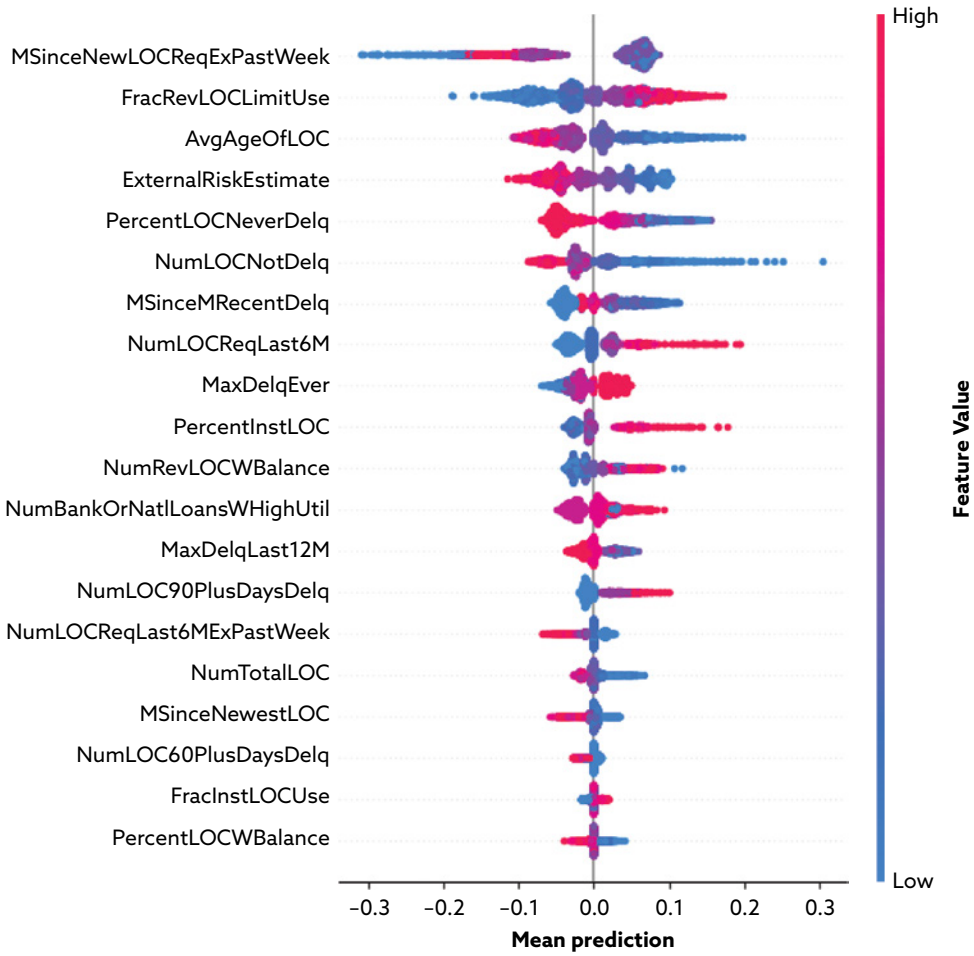
Source: Middle panel in Exhibit 3 of Li et al. (2020), used with permission.

to forecast individuals' home equity credit risk using a real-world dataset. The authors demonstrated methods to explain these ML models' output and to make them more accessible to end users. Notably, the authors developed different means of explainability to accommodate different requirements of the various stakeholders, including explanations for predictions of creditworthiness for loan companies (see **Exhibit 9**), stress tests for regulators, guiding counterfactuals for loan applicants, and explanatory rules for data scientists.

Summary and Future Directions

With the rapid development of artificial intelligence, which has led to the empowerment and automation of machine learning processes, one can expect further growth and adoption of data-driven methods in financial and investment decision-making processes. In particular, the

Exhibit 9. SHAP Feature Importance for the Probability of Default



Note: The larger values (in red) decrease the probability of default, while the smaller values (in blue) increase the probability of default.

Source: Figure 3(a) of Davis et al. (2023), used with permission.

emergence of large language models (LLMs) and generative AI, capable of generating a reasonable first-pass analysis of financial data and accompanying computer codes, is anticipated to further increase the adoption of data-driven methods. In addition, researchers are investigating methods to combine multiple LLMs (LLM ensembles) to enhance overall performance. Despite all of these attempts and because of the black-box perception associated with many ML and AI methods, key considerations remain in some critical applications of such methods—in particular, those involving high-risk decisions, fairness, and transparency. Frameworks that carefully blend both traditional and modern frameworks—including explanatory, explainability, and predictive—while drawing on the domain knowledge, may have a greater chance of success and survival in an environment typically characterized by the existence of competing agents, high noise-to-signal ratios, and rare arbitrage opportunities.

Acknowledgment

I would like to thank Tod McKenna for his support.

Disclaimer

The views expressed are those of the author and do not reflect the views of Citi or any of its affiliates. Any statements made have not been verified by Citi for accuracy and completeness.

References

Bagnara, Matteo. 2024. "Asset Pricing and Machine Learning: A Critical Review." *Journal of Economic Surveys* 38 (1): 27–56. doi:10.1111/joes.12532.

Bates, John M., and Clive W. J. Granger. 1969. "The Combination of Forecasts." *Journal of the Operational Research Society* 20 (4): 451–68. https://doi.org/10.1057/jors.1969.103.

Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32. doi:10.1023/A:1010933404324.

Chen, Tianqi, and Carlos Guestrin. 2016. "XGBoost: A Scalable Tree Boosting System." In *KDD '16: Proceedings of the 22nd SIGKDD Conference on Knowledge Discovery and Data Mining*, 785–94. New York: Association for Computing Machinery. doi:10.1145/2939672.2939785.

Connor, Gregory. 1995. "The Three Types of Factor Models: A Comparison of Their Explanatory Power." *Financial Analysts Journal* 51 (3): 42–46. doi:10.2469/faj.v51.n3.1904.

Cowden, Chad, Frank J. Fabozzi, and Abdolreza Nazemi. 2019. "Default Prediction of Commercial Real Estate Properties Using Machine Learning Techniques." *Journal of Portfolio Management* 45 (7): 55–67. doi:10.3905/jpm.2019.1.104.

Cvetkovic, Miljana. 2024. "LSTM versus Gradient Boosting for Asset Pricing: A Case Study of the S&P 500." *Journal of Financial Data Science* 6 (4): 49–71. doi:10.3905/jfds.2024.1.169.

Davis, Randall, Andrew W. Lo, Sudhanshu Mishra, Arash Nourian, Manish Singh, Nicholas Wu, and Ruixun Zhang. 2023. "Explainable Machine Learning Models of Consumer Credit Risk." *Journal of Financial Data Science* 5 (4): 9–39. doi:10.3905/jfds.2023.1.141.

Flach, Peter. 2012. *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. Cambridge, UK: Cambridge University Press. doi:10.1017/CBO9780511973000.

Freund, Yoav, and Robert E. Schapire. 1996. "Experiments with a New Boosting Algorithm." In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML'96)*, 148–56. San Francisco: Morgan Kauffman Publishers. https://cseweb.ucsd.edu/~yfreund/papers/boostingexperiments.pdf.

Friedman, Jerome H. 2001. "Greedy Function Approximation: A Gradient Boosting Machine." *Annals of Statistics* 29 (5): 1189–1232. https://www.jstor.org/stable/2699986.

Géron, Aurélien. 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow*, 2nd ed. Sebastopol, CA: O'Reilly Media.

Gu, Shihao, Bryan Kelly, and Dacheng Xiu. 2020. "Empirical Asset Pricing via Machine Learning." *Review of Financial Studies* 33 (5): 2223–73. doi:10.1093/rfs/hhaa009.

Harvey, Campbell R., Yan Liu, and Heqing Zhu. 2016. "... and the Cross-Section of Expected Returns." *Review of Financial Studies* 29 (1): 5–68. doi:10.1093/rfs/hhv059.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2017. *The Elements of Statistical Learning*, 2nd ed. New York: Springer.

Hutchinson, James M., Andrew W. Lo, and Tomaso Poggio. 1994. "A Nonparametric Approach to Pricing and Hedging Derivative Securities via Learning Networks." *Journal of Finance* 49 (3): 851–89. doi:10.1111/j.1540-6261.1994.tb00081.x.

Ivașcu, Codruț-Florin. 2021. "Option Pricing Using Machine Learning." *Expert Systems with Applications* 163 (January). doi:10.1016/j.eswa.2020.113799.

Ke, Guolin, Qi Meng, Thomas Finely, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tia-Yan Liu. 2017. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree." In *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, 3149–57.

Kirkby, Justin L., and Victor Andrean. 2024. "Forecasting Bid–Ask Spreads in Foreign Exchange: Analysis and Machine Learning Prediction." *Journal of Financial Data Science* 6 (4): 61–75. doi:10.2139/ssrn.4701477.

Li, Sophia Zhengzi, and Yushan Tang. 2024. "Automated Volatility Forecasting." Working paper (3 April). doi:10.2139/ssrn.3776915.

Li, Yimou, David Turkington, and Alireza Yazdani. 2020. "Beyond the Black Box: An Intuitive Approach to Investment Prediction with Machine Learning." *Journal of Financial Data Science* 2 (1): 61–75. doi:10.3905/jfds.2019.1.023.

López de Prado, Marcos. 2019. "Beyond Econometrics: A Roadmap Towards Financial Machine Learning." Working paper (19 September). doi:10.2139/ssrn.3365282.

Molnar, Christoph. 2019. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. https://christophm.github.io/interpretable-ml-book.

Rasekhschaffe, Keywan Christoph, and Robert C. Jones. 2019. "Machine Learning for Stock Selection." *Financial Analysts Journal* 75 (3): 70–88. doi:10.1080/0015198X.2019.1596678.

Shapley, Lloyd S. 1953. "A Value for *n*-Person Games." In *Contributions to the Theory of Games, Volume II*, edited by H. W. Kuhn and A. W. Tucker, 307–17. Princeton, NJ: Princeton University Press. doi:10.1515/9781400881970-018.

Simonian, Joseph. 2022. "Forests for Fama." *Journal of Financial Data Science* 4 (1): 145–57. doi:10.3905/jfds.2021.1.086.

Simonian, Joseph, and Frank J. Fabozzi. 2019. "Triumph of the Empiricists: The Birth of Financial Data Science." *Journal of Financial Data Science* 1 (1): 10–13. doi:10.3905/jfds.2019.1.010.

Simonian, Joseph, Chenwei Wu, Daniel Itano, and Vyshaal Narayanam. 2019. "A Machine Learning Approach to Risk Factors: A Case Study Using the Fama–French–Carhart Model." *Journal of Financial Data Science* 1 (1): 32–44. doi:10.3905/jfds.2019.1.032.

Vovk, Vladmir, Alexander Gammerman, and Glenn Shafer. 2022. *Algorithmic Learning in a Random World*, 2nd ed. London: Springer. doi:10.1007/978-3-031-06649-8.

Wang, Xiaoqian, Rob J. Hyndman, Feng Li, and Yanfei Kang. 2023. "Forecast Combinations: An Over 50-Year Review." *International Journal of Forecasting* 39 (4): 1518–47. doi:10.1016/j.ijforecast.2022.11.005.

Wolpert, David H. 1992. "Stacked Generalization." *Neural Networks* 5 (2): 241–59. doi:10.1016/S0893-6080(05)80023-1.

Wolpert, David H. 1996. "The Lack of A Priori Distinctions between Learning Algorithms." *Neural Computation* 8 (7): 1341–90. doi:10.1162/neco.1996.8.7.1341.

Wolpert, David H. 2002. "The Supervised Learning No-Free-Lunch Theorems." In *Soft Computing and Industry*, edited by Rajkumar Roy, Mario Köppen, Seppo Ovaska, Takeshi Furuhashi, and Frank Hoffmann. London: Springer. doi:10.1007/978-1-4471-0123-9_3.

Yazdani, Alireza. 2020. "Machine Learning Prediction of Recessions: An Imbalanced Classification Approach." *Journal of Financial Data Science* 2 (4): 21–32. doi:10.3905/jfds.2020.1.040.

DEEP LEARNING

Paul Bilokon, PhD
CEO, Thalesians Ltd.
Visiting Professor, Imperial College London

Joseph Simonian, PhD
Senior Affiliate Researcher, CFA Institute

Introduction

Deep learning refers to a type of machine learning algorithm that uses multiple layers to progressively extract higher-level features from input data. For example, in image processing, lower layers may identify edges, while higher layers may identify more sophisticated concepts, such as digits, letters, or faces. The first major business application of deep learning was to check processing in the early 2000s. The modern deep learning revolution builds on connectionism—an approach in cognitive science that seeks to explain mental phenomena using artificial neural networks. Connectionism took its rise from the work of Warren McCulloch, Walter Pitts, Donald Olding Hebb, and Karl Lashley. Modern neural networks can be thought of as generalizations of the "perceptron" introduced by Frank Rosenblatt in 1957. In this chapter, we explore the foundations of deep learning and its applications to finance and investing.

Background: Deciphering the Human Brain

Neuron architecture, in various degrees, forms the basis of deep learning algorithms. The detailed study of neurons commenced in the early 1900s, when anatomists began using microscopes and new staining methods to study the microscopic parts of the brain. It was around this time that neuroanatomists Santiago Ramón y Cajal and Camillo Golgi discovered "that nerve cells (neurons) are the building blocks of the brain and showing there are many different types" of neurons (Jones 1999).

Neuroscience progressed significantly through discoveries about how neurons interact. Researchers eventually identified the synapse as the point of connection where nerve cells communicate, leading to major insights into the workings of the central nervous system. Later work revealed that neurons transmit signals through both electrical impulses and chemical processes. The understanding of how neural activity strengthens connections between cells introduced the concept often summarized as "neurons that fire together become more strongly linked," forming the basis for associative or Hebbian learning, where repeated activation strengthens connections between neurons involved in the same process.

Around the same period, Alan Turing developed the idea of a mechanical model of computation, now known as the Turing machine. His work provided a mathematical framework for defining what it means for a task or function to be computationally solvable. This led to the principle

 •

that any process considered effectively computable can be represented by a Turing machine, forming a cornerstone of modern computer science. In 1943, McCulloch and Pitts published "A Logical Calculus of the Ideas Immanent to Nervous Activity," which described a mathematical model of the nervous system as a network of simple logical elements, known as *artificial neurons*, or later as *McCulloch–Pitts neurons*. These neurons take inputs, calculate a weighted sum, and produce an output signal based on a threshold function.

In 1957, Frank Rosenblatt at the Cornell Aeronautical Laboratory simulated a simple artificial neuron called a *perceptron* on an IBM 704. Later, he obtained funding from the Information Systems Branch of the United States Office of Naval Research and the Rome Air Development Center to build a custom-made computer, the Mark I Perceptron. It was first publicly demonstrated on 23 June 1960. The machine was part of a previously secret four-year NPIC (US National Photographic Interpretation Center) project that ran from 1963 through 1966, with the goal of developing the Mark I into a useful tool for photo-interpreters. Indeed, the Mark I was a fairly powerful pattern learning and recognition device for its time and was able to reliably learn to classify visual patterns into groups on the basis of certain geometric similarities and differences, utilizing properties such as position in the retinal field of view, geometric form, occurrence frequency, and size.

Perceptrons and feed forward neural networks (FFNNs) feed information from the front to the back (respectively, input and output). A common characteristic of FFNNs is that in them, two adjacent layers are "fully connected," which means that every neuron from one layer is connected to every neuron from another layer. FFNNs are typically trained through backpropagation, giving the network paired datasets of "what goes in" and "what we want to have coming out." This is called supervised learning, as opposed to unsupervised learning, where we only give it input and let the network fill in the blanks. The error being backpropagated is often some variation of the difference between the input and the output (such as mean squared error, or MSE) or just the linear difference. Given that the network has enough hidden neurons, it can theoretically always model the relationship between the input and output. Practically, their use is a lot more limited, but they are popularly combined with other networks to form new networks. In **Exhibit 1**, we show a perceptron, and in **Exhibit 2**, we show an FFFN. (Note that all exhibits in this chapter were created by the authors).

Extreme learning machines (ELMs) (Huang 2015) are similar to FFNNs but have random connections. They have many similarities to liquid state machines and echo state networks but are neither recurrent nor spiking and do not use backpropagation. Instead of backpropagation, ELMs start with random weights and train the weights in a single step according to the least-squares fit (lowest error across all functions). This results in a considerably less expressive network but one that is also significantly faster than backpropagation.

Deep residual networks (DRNs) (He, Zhang, Ren, and Sun 2016) are very deep FFNNs with additional connections passing input from one layer to one or more further layers.

Exhibit 1. Perceptron Architecture

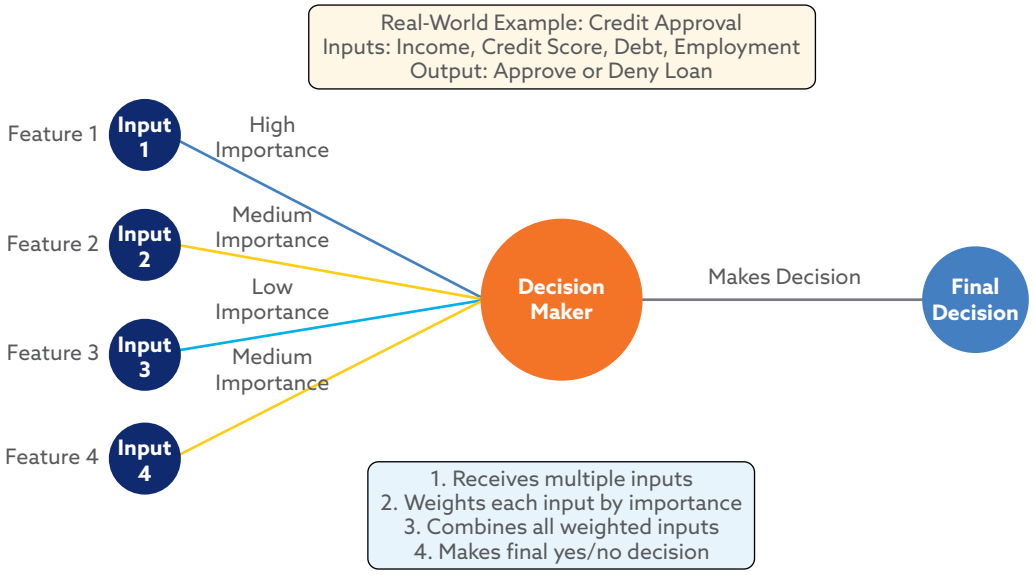
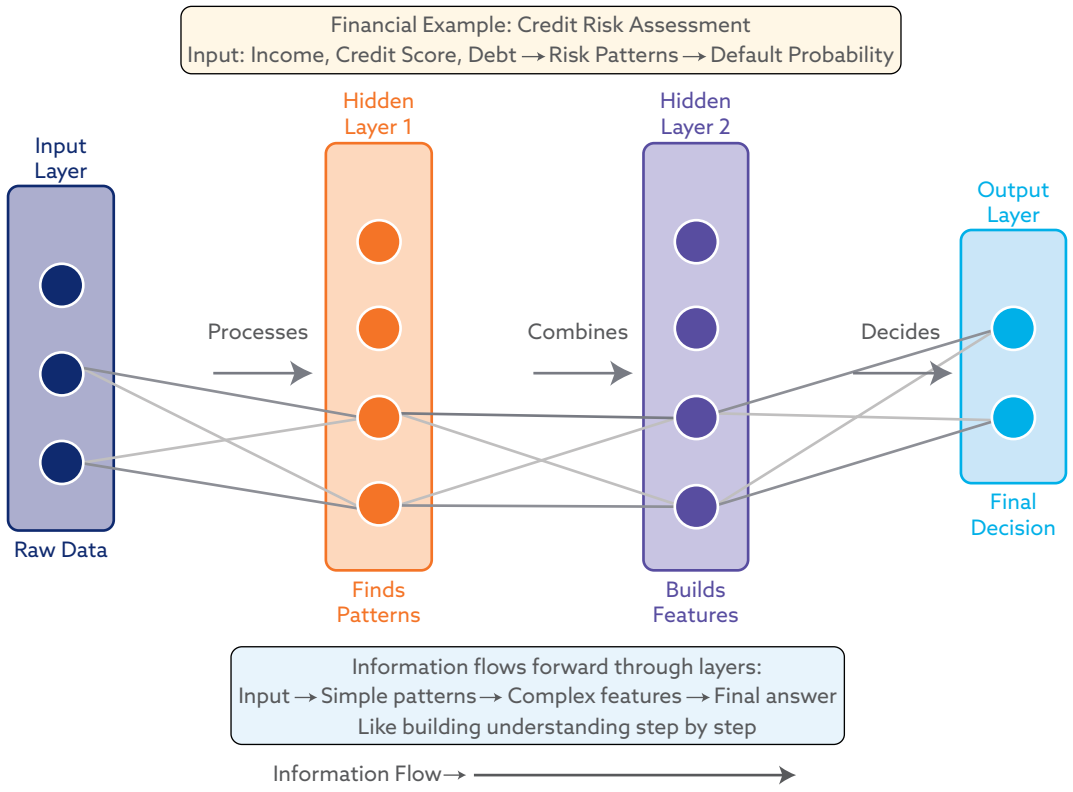


Exhibit 2. Feed Forward Neural Network Architecture



Now, let us implement a perceptron in Python. Going through the steps manually will give us a good idea about how neural networks solve problems. First, we generate some data:

```
X = []
for x1 in [0., 1.]:
for x2 in [0., 1.]:
X. append ([ x1 , x2 ])
y = []
for x in X:
y. append ( x[0] and x [1])
```

It is more convenient to work with NumPy arrays than native Python lists, so we convert accordingly:

```
X = np. array ( X) y = np. array ( y)
```

We initialize the weights and bias to (pseudo)random values sampled from the standard normal distribution:

```
weights = np. random . normal( size =(2 , 1)) bias = np. random . normal ()
```

The function "predict" will predict *y* given *X* as well as fixed weights and bias:

```
def predict(X, weights, bias):
    y_pred = []
    for x in X:
        x = x.reshape(-1, 1)
        v = weights.T @ x + bias
        y_pred.append(0. if v < 0. else 1.)
    return np.array(y_pred)
```

The function fit updates the weights and bias using gradient descent with γ set to the learning_rate:

```
def fit(X, y, weights, bias, learning_rate=.01, epochs=1):
    for i in range(epochs):
        for x, target in zip(X, y):
            x = x.reshape(-1, 1)
            v = weights.T @ x + bias
            y_pred = 0. if v < 0. else 1.
            if target != y_pred:
                bias -= learning_rate * (y_pred - target)
                weights -= learning_rate * (y_pred - target) * x
    return weights, bias
```

The epoch parameter represents the number of complete cycles (*epochs*) through the entire training dataset and indicates the number of passes that the machine learning algorithm must complete during that training. We proceed as follows:

```
weights, bias = fit(X, y, weights, bias, epochs=100)
y_pred = predict(X, weights, bias)
y_pred
# array([0., 0., 0., 1.])

y
# array([0., 0., 0., 1.])

weights, bias
# (array([[0.12671415],
#       [0.0117357]]),

# -0.13231146189930793)
```

These weights and bias have been found by the gradient descent algorithm. Our *procedural* code is a bit haphazard. It would be cleaner to use the *object-oriented* approach and encapsulate the notion of a perceptron in a dedicated class:

```
class Perceptron(object):
  def __init__(self, dim):
    self.dim = dim
    self.weights = np.random.normal(size=(self.dim, 1))
    self.bias = np.random.normal()
```

```
def fit(self, X, y, learning_rate=.01, epochs=1):
    for i in range(epochs):
        for x, target in zip(X, y):
            x = x.reshape(-1, 1)
            v = self.weights.T @ x + self.bias
            y_pred = 0. if v < 0. else 1.
            if target != y_pred:
                self.bias -= learning_rate * (y_pred - target)
                self.weights -= learning_rate * (y_pred - target) * x


def predict(self, X):
    y_pred = []
    for x in X:
        x = x.reshape(-1, 1)
        v = self.weights.T @ x + self.bias
        y_pred.append(0. if v < 0. else 1.)
    return np.array(y_pred)
```

The code we have provided is for pedagogical purposes; as such, a class already exists in *scikit-learn*, the popular free software machine learning library for Python. Scikit-learn grew out of a June 2007 Google Summer of Code project by David Cournapeau and now features various classification, regression, and clustering algorithms, including support vector machines, random forests, gradient boosting, *k*-means, and DB-SCAN. Scikit-learn is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Although individual perceptrons turned out to be of limited practical use, networks of perceptrons (or feed forward neural networks, FFNNs) were soon recognized as powerful universal function approximators. Their calibration in practice was impeded by computational restrictions, which were overcome algorithmically using the backpropagation algorithm (Rumelhart, Hinton, and Williams 1986), a major computational advance, and improvements in hardware, such as the emergence of GPUs. The progress was not uniform, and this academic area

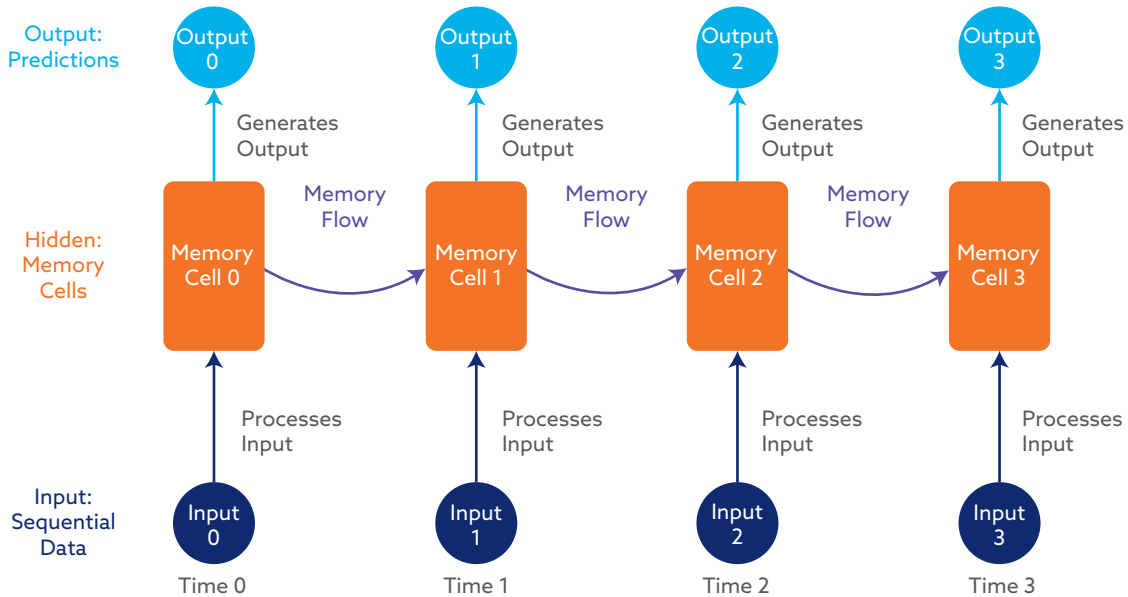
went through several periods of funding cuts, which are sometimes referred to as *AI winters* (Harguess and Ward 2022).

More Sophisticated Deep Learning Frameworks

In this section, we describe more sophisticated deep learning frameworks, several of which have become popular in finance. We begin with *recurrent neural networks* (RNNs) (Rumelhart et al. 1986), which are FFNNs that are not stateless—rather, they have connections between passes, connections through time. They are popular in financial applications. Neurons take input information not only from previous layers but also from themselves on previous passes. Thus, the order in which the input is fed into and trained in the network matters. One major challenge with RNNs is the vanishing gradient problem, where, depending on the activation functions used, information rapidly gets lost over time. This is similar to how some FFNNs lose information in depth. Nevertheless, RNNs are a good choice for many time series applications. We show an RNN in **Exhibit 3**.

Echo state networks (Jaeger and Haas 2004) are another type of (recurrent) network. They set themselves apart by having random connections between neurons (i.e., they are not organized into neat sets of layers). Instead of feeding input and backpropagating the error, they feed the input, update the neurons, and observe the output over time. The input and output layers have a somewhat unconventional role as the input layer is used to prepare the network and the output layer acts as an observer of the activation patterns that develop over time. During the training period, only the connections between the observer and the hidden units are changed.

Exhibit 3. Recurrent Neural Network Architecture Through Time

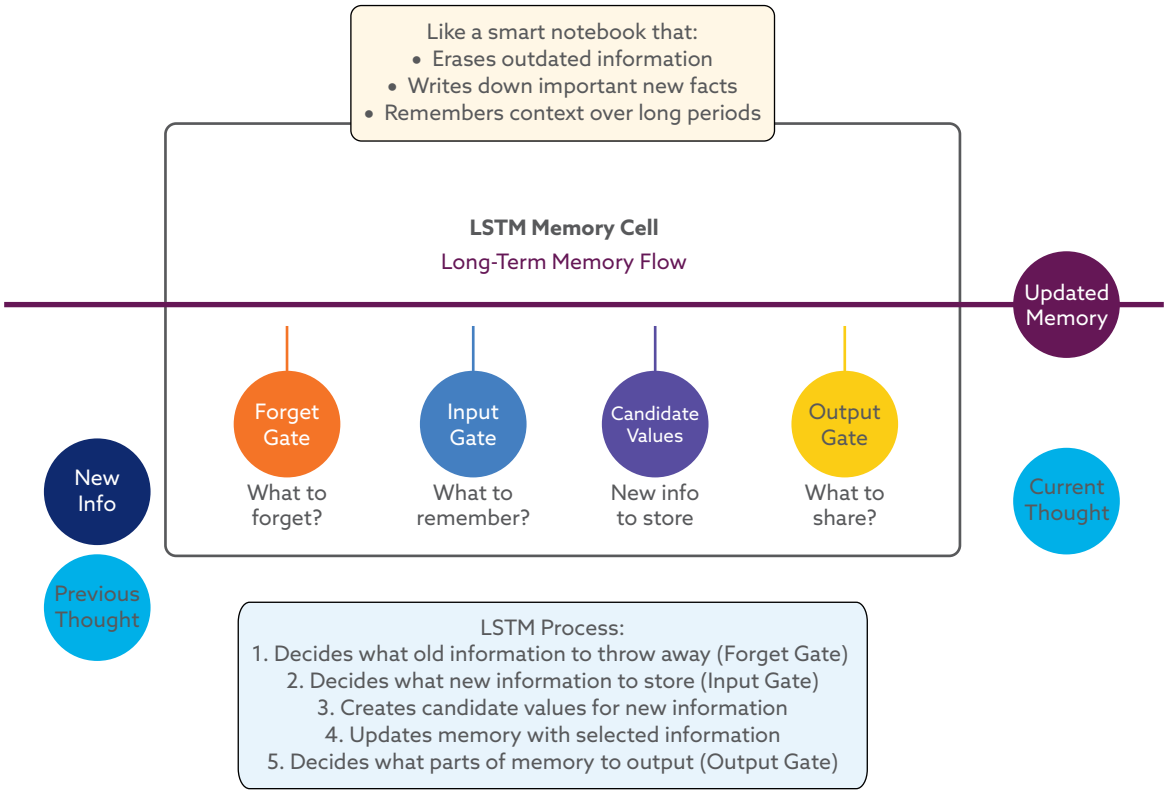


Key Concept: Memory cells remember information from previous time steps
This allows the network to understand sequences and context

Long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) networks try to combat the vanishing/exploding gradient problem by introducing gates and an explicitly defined memory cell. These are inspired mostly by circuitry, not so much biology. Each neuron has a memory cell and three gates: input, output, and forget. The function of these gates is to safeguard the information by stopping or allowing the flow of it. The input gate determines how much of the information from the previous layer is stored in the cell. The output layer takes the job on the other end and determines how much of the next layer gets to know about the state of this cell. The forget gate, as the name suggests, enables the network to forget. LSTMs have been shown to be able to learn complex sequences, such as writing prose or composing music. We show an LSTM in **Exhibit 4**.

Neural Turing machines (NTMs) (Graves, Wayne, and Danihelka 2014) can be understood as an abstraction of LSTMs and an attempt to undo the black-box nature of neural networks (and provide insight into what is going on in there). NTMs augment the traditional neural network architecture with an external memory bank, allowing it to perform tasks that require both computation and flexible memory manipulation, such as copying, sorting, and associative recall. The architecture consists of three main components: a controller, which processes inputs and determines how to interact with memory; a memory matrix, which serves as the external storage for information; and read/write heads, which focus attention on specific memory locations for reading or writing data.

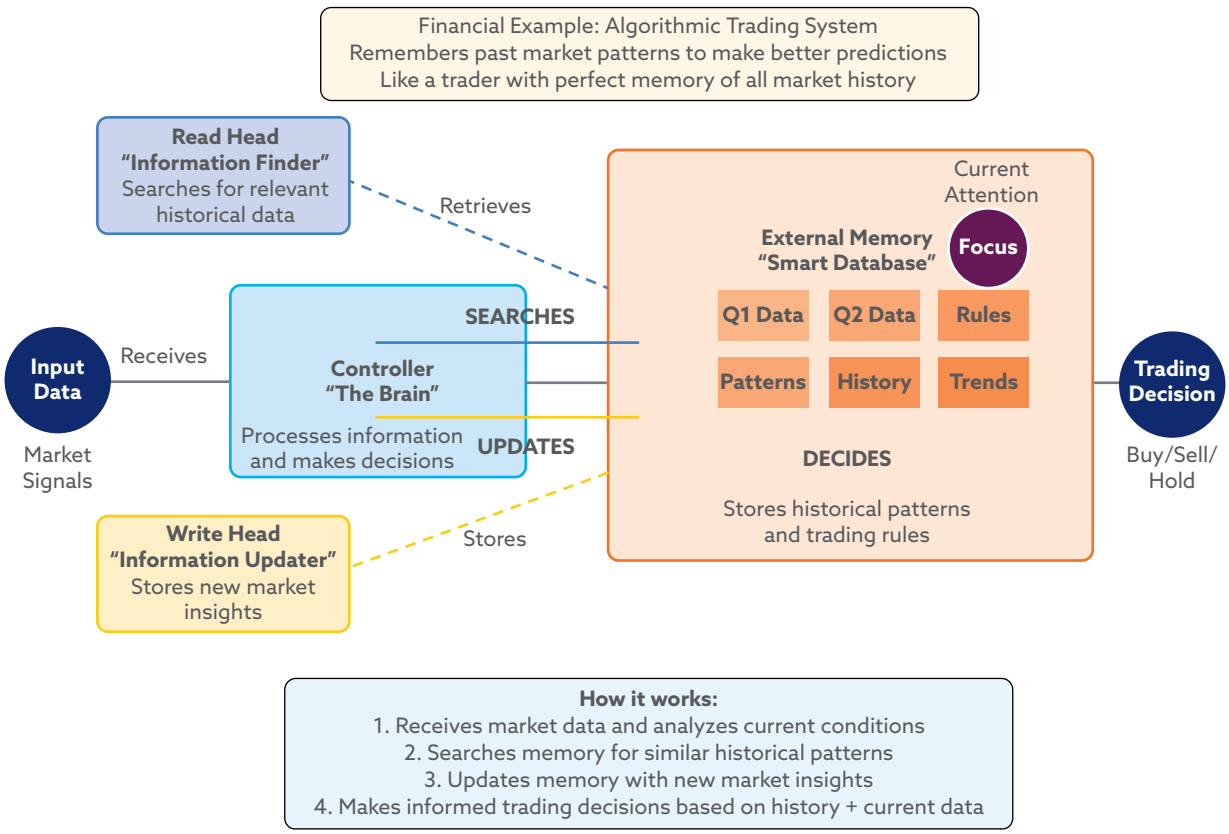
Exhibit 4. LSTM Architecture



The controller, often implemented as a recurrent neural network, receives both the current input and the previous memory readout, enabling it to make informed decisions about memory access. The read and write heads use attention mechanisms to determine how much focus to place on each memory location, allowing the NTM to interact with memory in a smooth, differentiable way. This differentiability means the entire system can be trained end to end using gradient descent, just like standard neural networks. The result is a model that combines the pattern recognition strengths of neural networks with the algorithmic flexibility of a Turing machine, making NTMs particularly suited for tasks that require reasoning over sequences and manipulating stored data in complex ways. *Differentiable neural computers* (Graves, Wayne, Reynolds, Harley, Danihelka, Grabska-Barwińska, Colmenarejo, Grefenstette, and Ramalho 2016) are enhanced neural Turing machines with scalable memory, inspired by how memories are stored by the human hippocampus. We show the architecture of an NTM in **Exhibit 5**.

Gated recurrent units (GRUs) (Cho, van Merrienboer, Bahdanau, and Bengio 2014) are a variation on LSTMs. They contain one less gate and are wired slightly differently: instead of an input, output, and forget gate, they have an update gate. The update gate determines both how much information to keep from the last state and how much information to let in from the previous layer. The reset gate functions much like the forget gate of an LSTM but is located at different points in the decision-making process. In most cases, they function similarly to LSTMs but are slightly faster and easier to run (albeit also slightly less expressive).

Exhibit 5. Neural Turing Machine Architecture



Bidirectional recurrent neural networks and *bidirectional long short-term memory networks* (BNs) (Schuster and Paliwal 1997) look identical to their unidirectional counterparts. The main difference between them is that BNs are not just connected to the past but also connected to the future. This means that during training, the network fills in gaps instead of simply advancing information. For example, instead of advancing an image on the edge, it could fill a hole in the middle of an image.

Autoencoders (AEs) represent a different use of FFNNs rather than a fundamentally different architecture. In autoencoders, we compress information. In AEs, the entire network resembles an hourglass, having smaller hidden layers relative to the input and output layers. AEs can be trained using backpropagation by feeding input and setting the error to be the difference between the input and what came out (Hinton and Salakhutdinov 2006). *Variational autoencoders* (VAEs) have the same architecture as AEs but are "taught" an approximated probability distribution of the input samples data (Kingma and Welling 2014). *Denoising autoencoders* are AEs where we feed in the input data with noise. The output of the network is compared with the original input without the noise, which encourages the network to learn broader features instead of details (Vincent, Larochelle, Bengio, and Manzagol 2008).

With *sparse autoencoders* (SAEs) (Makhzani and Frey 2013) we encode information in more space. So instead of the network converging in the middle and then expanding back to the input size, the middle of the network is the zone of expansion. SAEs are useful in extracting small features from a dataset. Instead of simply feeding back the input as in some other networks, we feed back the input with the addition of a sparsity driver. This sparsity driver is often a "threshold filter," where only a certain error is passed back and trained; other errors will be "irrelevant" for that pass and set to zero. This is somewhat similar to spiking neural networks, where not all neurons fire all the time. Among the various types of encoders, VAEs in particular have become popular in finance because of their utility in anomaly detection and generating synthetic data. We show VAE architecture in **Exhibit 6**.

Generative adversarial networks (GANs) (Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, and Bengio 2014) are a class of generative models that use a game-theoretic framework to learn and generate new data that mimics the distribution of a given dataset. GANs consist of two neural network twins: the *generator* and the *discriminator*. The generator creates synthetic data from random noise, attempting to mimic the real data distribution. The discriminator distinguishes between real data (from the dataset) and fake data (produced by the generator). The discriminator receives either training data or generated content from the generator. Information regarding how well the discriminator is able to correctly predict the data source is then used as part of the error for the generating network. This process in essence creates a competitive game in which the discriminator gets better at distinguishing real data from generated data and the generator learns to become less predictable to the discriminator. GANs have become popular in finance as a means to generate synthetic data. We show the GAN architecture in **Exhibit 7**.

Liquid state machines (Maass, Natschläger, and Markram 2002) are a type of spiking neural network that replace the usual sigmoid activation functions with discrete threshold mechanisms, where each neuron also maintains an internal state or accumulated potential. Instead of overwriting the neuron's current value with the weighted sum of its neighbors, the input is incrementally added to the neuron's stored energy. When this accumulated value surpasses a defined threshold, the neuron emits a spike, transferring energy to connected units.

Exhibit 6. Variational Autoencoder Architecture

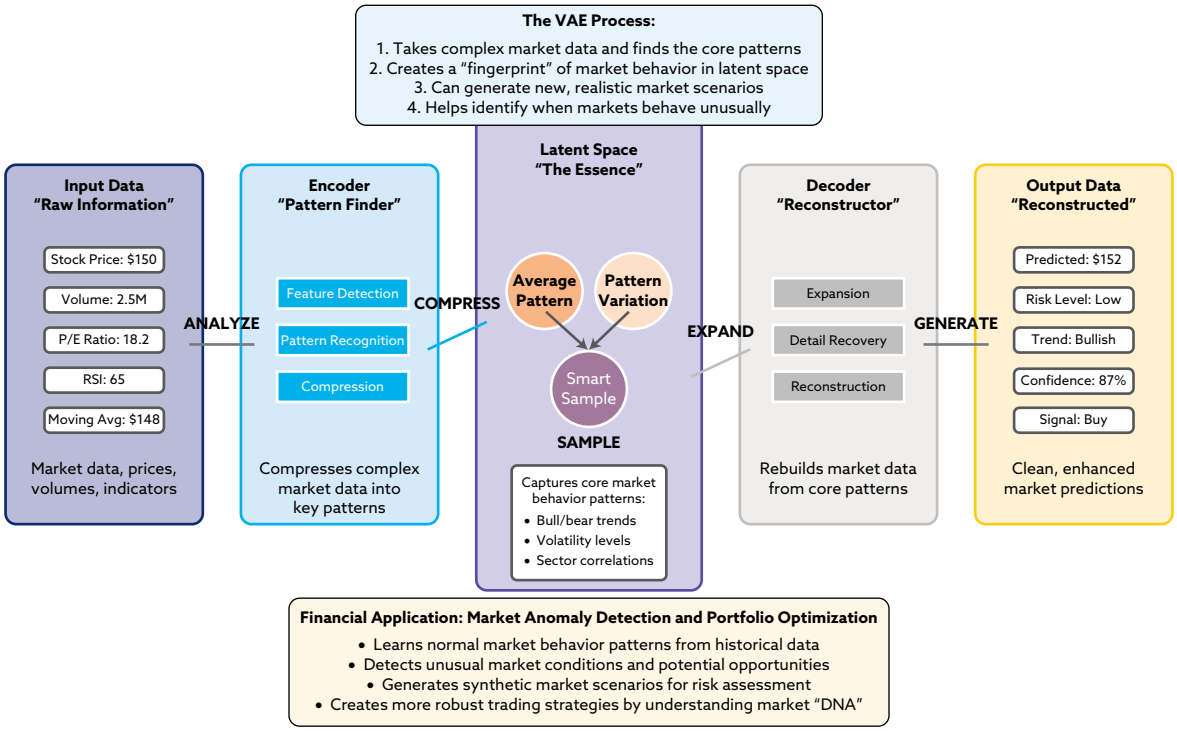
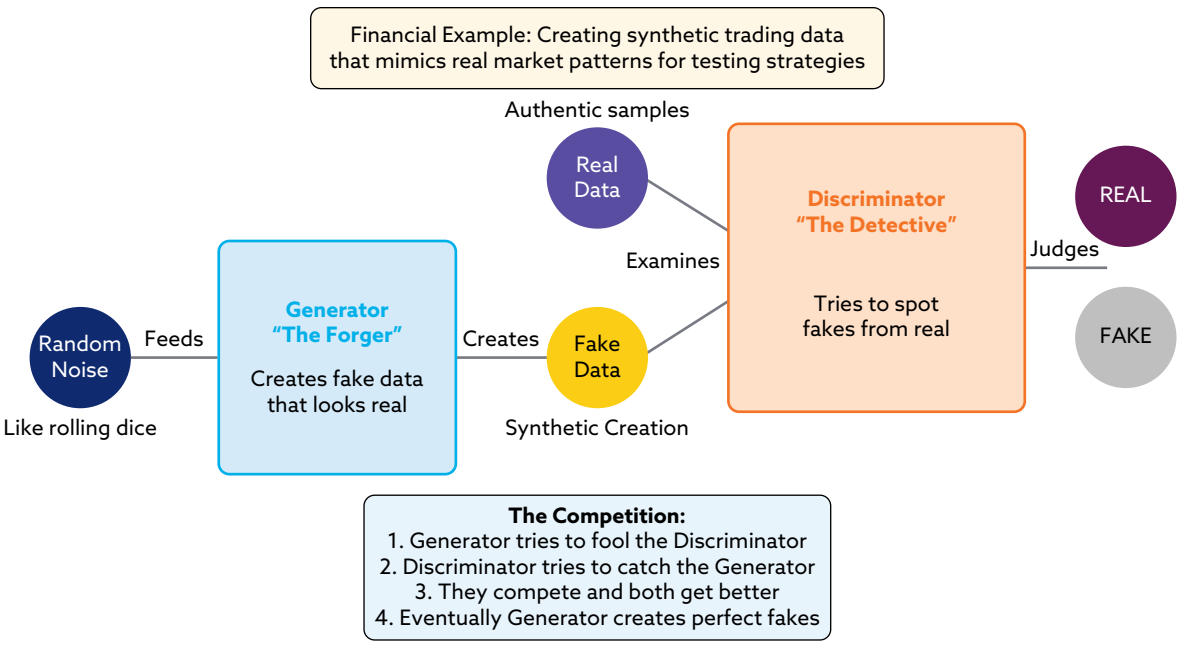


Exhibit 7. GAN Architecture



This produces a characteristic firing pattern with long periods of inactivity punctuated by sudden bursts of activity, behavior that is typical of spiking behaviors. A *Hopfield network* (HN) is a network in which each neuron is connected to every other neuron. Every node is input before training, hidden during training, and output afterward. These networks are trained by setting the neurons' value to the desired pattern; then, the weights can be computed. These networks are often called *associative memory* because they converge to the most similar state as the input. *Boltzmann machines* (BMs) are similar to HNs, except that some neurons are marked as input neurons and others remain "hidden." The algorithm begins by assigning weights randomly and learns through backpropagation, or *contrastive divergence*, where a Markov chain is used to determine the gradients between two informational gains. At the end of a full network update, the input neurons become output neurons. In contrast to HNs, in BMs, the neurons mostly have binary activation patterns.

Convolutional neural networks (CNNs) (LeCun, Bottou, Bengio, and Haffner 1998) or *deep convolutional neural networks* (DCNNs) are different from most other deep learning algorithms. They are primarily used for image processing but can also be applied to other types of data such as audio. A typical use case for CNNs is where you input network images and the network classifies the data—for example, "cat" versus "dog." CNNs tend to start with an input "scanner" that is not intended to parse all training data at once. The input data are then processed through convolutional layers, where not all nodes are connected to all nodes. Each node only concerns itself with neighboring cells in close proximity (how close varies by application). Convolutional layers also tend to shrink as they deepen.

Aside from these convolutional layers, CNNs also frequently feature pooling layers. Pooling is a way to filter out details. One commonly used pooling technique is max pooling, where we take, for example, three pixels and pass on the pixel with the most amount of red. Real-world implementations of CNNs often attach an FFNN to the end of the algorithm to further process the data, a maneuver that allows for highly nonlinear abstractions.

Deconvolutional networks (DNs), also known as *inverse graphics networks* (IGNs), are reversed convolutional neural networks. For example, consider the case where we feed a network the word "dog" (or a binary classification input vector) and train it to produce dog-like pictures by comparing what it generates to real pictures of dogs. DNNs can also be combined with FFNNs just like regular CNNs can. When this is done, the pooling layers often found in CNNs are frequently replaced with analogous inverse operations, primarily interpolation and extrapolation with biased assumptions.

Finally, *capsule networks* (CapsNet) (Sabour, Frosst, and Hinton 2017) are biologically inspired alternatives to pooling, where neurons are connected with a vector of weights instead of just one weight (a scalar). *Kohonen networks* (Kohonen 1990), on the other hand, use competitive learning to classify data without supervision. In the next section, we describe some specific applications of deep learning to finance.

Applications in Finance

Derivatives pricing was one of the early targets of applied neural networks in finance. Early adopters of neural networks in option pricing include Malliaris and Salchenberger (1993); Hutchinson, Lo, and Poggio (1994); Yao, Li, and Tan (2000); Bennell and Sutcliffe (2004); and Gradojevic, Gencay, and Kukolj (2009). With the advent of deep learning

(Goodfellow 2016), neural networks started to become mainstream, and deep learning reentered quants' collective consciousness, particularly following publication of work on *deep learning volatility* (Ferguson and Green 2018; Horvath, Muguruza, and Tomas 2021), which soon became mainstream.

In these articles, the authors presented neural network-based calibration methods that perform the calibration task within a few milliseconds for the full implied surface. These frameworks are applicable across a range of volatility models—including second-generation stochastic volatility models and the rough volatility family—and a range of derivative contracts. Neural networks are being used in offline approximations of complex pricing functions, which are difficult to represent or time consuming to evaluate by other means.

In some instances, finance has generated algorithmic advances, such as *differential deep learning* (Huge and Savine 2020). It combines automatic adjoint differentiation (Capriotti and Giles 2024) with machine learning, where the models are trained on examples of not only inputs and labels but also differentials of labels with regard to inputs, yielding highly effective pricing and risk approximations. More recently, these approaches have been applied in a wider range of settings, such as stochastic volatility (Sridi and Bilokon 2023), including exotic products (Ma, Ventre, Tiranti, and Chen 2025).

Outside the context of derivatives pricing, deep methods have been applied for *alpha generation*. Kolm, Turiel, and Westray (2023) deployed deep learning to forecast high-frequency returns at multiple horizons for 115 stocks traded on Nasdaq using order book information at the most granular level. State-of-the-art predictive accuracy was achieved by running "off-the-shelf" artificial neural networks on stationary inputs derived from the order book. Using cross-sectional regressions, the authors linked an LSTM network's forecasting performance to stock characteristics at the market microstructure level, suggesting "information-rich" equities can be predicted more accurately. The effective horizon of stock-specific forecasts was found to be approximately two average price changes.

Deep econometrics (Bilokon 2025) is a principled rethinking of the classical econometric (Ruud 2000) and time-series (Tsay 2010) analyses using deep learning techniques (Goodfellow 2016; Dixon, Halperin, and Bilokon 2020). The focus of Bilokon (2025) is on the estimation of parameters in various econometric settings. Some applications focus on the rethinking of the Wiener–Kolmogorov filtering theory, the so-called deep stochastic filters (Horvath, Kratsios, Limmer, and Yang 2023; Stok, Bilokon, and Simonian 2024).

Some interesting applications have arisen out of the combination of reinforcement deep learning and reinforcement learning, a framework where agents learn through a system of rewards and punishments, based on their actions in specific states. Reinforcement learning (Sutton 2020) differs from supervised learning in that the ground truth may not necessarily be known. Feedback is often evaluative rather than prescriptive, is often delayed, and may be sourced from the environment. In recent years, this subfield of machine learning/artificial intelligence gained public recognition when a reinforcement-learning-based system beat the human champion at the game of Go (Silver, Huang, Maddison, Guez, Sifre, van den Driessche, Schrittwieser, et al. 2016). Soon after, reinforcement learning began to gain popularity in finance.

Early adopters started to use deep reinforcement learning for hedging derivative contracts, giving rise to *deep hedging* (Halperin 2017; Buehler, Gonon, Teichmann, and Wood 2019; Kolm and Ritter 2019a; Cao, Chen, Hull, and Poulos 2021). Reinforcement learning in this

application was used to derive optimal hedging strategies for derivatives in cases where transaction costs and other frictions are present.

Far from being a novelty, many of these algorithms have been extensively studied and evaluated (see, e.g., Stoiljkovic 2025). Financial applications have led to a cross-pollination of ideas, which has contributed new and enhanced reinforcement learning techniques, such as enhancements to inverse reinforcement learning (Halperin, Liu, and Zhang 2022) and distributional reinforcement learning (Halperin 2024). Other researchers focused on applications of reinforcement learning to wealth management (e.g., Dixon, Gvozdanovic, and O'Kane 2023). This gave rise to G-Learner (Dixon and Halperin 2020), a reinforcement learning algorithm that uses explicitly defined one-step rewards, does not assume a data generation process, and is appropriate for use with noisy data. GIRL (Dixon and Halperin 2020) applies goal-based G-learning to inverse reinforcement learning (IRL) (Dixon et al. 2020), where rewards collected by the agent are not observed but inferred.

Others have combined ideas from the emerging subfields of reinforcement learning, such as multiarmed bandits, to update the now classic Markowitz–Sharpe framework (Varlashova and Bilokon 2025; Bilokon and Varlashova 2025). This framework arose from the rethinking of some of the issues relevant to finance, such as nonstationarity, in novel and nontrivial ways. Needless to say, the extensive work on the uses of machine learning for time-series forecasting (Dixon, Klabjan, and Bang 2017; Stok et al. 2024) is the foundation of many trading applications. Jaddu and Bilokon (2024) combined deep learning on the order books with reinforcement learning and backtested the resulting strategies in the presence of frictions. Zejnullahu, Moser, and Osterrieder (2022) explored in considerable detail the use of double deep Q-networks for trading purposes. Pendharkar and Cusatis (2018) explored applications of reinforcement learning agents to trading financial indexes. We point out that financial applications of reinforcement learning have been extensively reviewed by Kolm and Ritter (2019b); Charpentier, Élie, and Remlinger (2023); and Hambly, Xu, and Yang (2023).

Other advances in machine learning have been applied to create synthetic financial data, which are particularly useful in small data environments (Buehler, Horvath, Lyons, Arribas, and Wood 2020; Bühler, Horvath, Lyons, Arribas, and Wood 2020). Some research has focused on speeding up the calculations on a wider range of devices, such as field-programmable gate arrays (Sobakinskikh and Bilokon 2025), rather than algorithmic advances. There has also been cross-disciplinary work, which is difficult to classify, at the boundaries of finance, machine learning, and physics (Halperin and Dixon 2020). Progress has occurred in one of the most controversial areas of applications of machine learning and artificial intelligence to finance—explainability (Bussmann, Giudici, Marinelli, and Papenbrock 2021). The rise of large language models, such as ChatGPT (OpenAI, Achiam, Adler, Agarwal, Ahmad, Akkaya, Aleman, et al. 2023) and Claude are likely to further revolutionize finance.

Concluding Thoughts

The application of deep learning to finance has evolved from early neural network experiments in derivative pricing to sophisticated deep learning systems that now permeate virtually every aspect of financial markets. Financial applications of deep learning have increasingly focused on practical implementation challenges. Early research often ignored market microstructure effects, transaction costs, and regulatory constraints. Contemporary work in deep hedging,

order flow prediction, and portfolio optimization explicitly incorporates these real-world frictions, making the resulting strategies more robust and implementable.

Several developments promise to further revolutionize finance. The emergence of large language models creates opportunities for natural language processing of financial documents, automated report generation, and sophisticated conversational interfaces for financial analysis. Quantum computing, although still in its infancy, may eventually enable the solution of optimization problems that are currently intractable. Meanwhile, regulatory developments around algorithmic transparency and explainable AI will likely shape how these technologies are deployed in practice.

The cross-pollination between finance and deep learning has benefited both fields. Finance has provided challenging real-world problems that have spurred methodological advances in such areas as differential machine learning and distributional reinforcement learning. Conversely, techniques developed in computer science have enabled financial practitioners to tackle previously unsolvable problems in risk management, trading, and asset allocation.

As we stand at this inflection point, with deep learning capabilities advancing at an unprecedented pace, the integration of artificial intelligence into financial markets appears not merely inevitable but already well underway. The question is no longer whether deep learning (and AI as a whole) will transform finance but, rather, how quickly and in what specific directions this transformation will proceed.

References

Bennell, Julia, and Charles Sutcliffe. 2004. "Black–Scholes versus Artificial Neural Networks in Pricing FTSE 100 Options." *Intelligent Systems in Accounting, Finance & Management* 12 (4): 243–60. doi:10.1002/isaf.254.

Bilokon, Paul. 2025. "Deep Econometrics." Working paper (9 June). doi:10.2139/ssrn.5286898.

Bilokon, Paul, and Valeria Varlashova. 2025. "Tail-Aware Portfolio Optimization Using Hoeffding-Informed Thresholds." Working paper (22 May). doi:10.2139/ssrn.5265442.

Buehler, Hans, Lukas Gonon, Josef Teichmann, and Ben Wood. 2019. "Deep Hedging." *Quantitative Finance* 19 (8): 1271–91. doi:10.1080/14697688.2019.1571683.

Buehler, Hans, Blanka Horvath, Terry Lyons, Imanol Perez Arribas, and Ben Wood. 2020. "Generating Financial Markets with Signatures." Working paper (21 July). doi:10.2139/ssrn.3657366.

Bühler, Hans, Blanka Horvath, Terry Lyons, Imanol Perez Arribas, and Ben Wood. 2020. "A Data-Driven Market Simulator for Small Data Environments." Working paper (21 June). doi:10.48550/arXiv.2006.14498.

Bussmann, Niklas, Paolo Giudici, Dimitri Marinelli, and Jochen Papenbrock. 2021. "Explainable Machine Learning in Credit Risk Management." *Computational Economics* 57 (1): 203–16. doi:10.1007/s10614-020-10042-0.

Cao, Jay, Jacky Chen, John Hull, and Zissis Poulos. 2021. "Deep Hedging of Derivatives Using Reinforcement Learning." *Journal of Financial Data Science* 3 (1): 10–27. doi:10.3905/jfds.2020.1.052.

Capriotti, Luca, and Mike Giles. 2024. "15 Years of Adjoint Algorithmic Differentiation (AAD) in Finance." *Quantitative Finance* 24 (9): 1353–79. doi:10.1080/14697688.2024.2325158.

Charpentier, Arthur, Romuald Élie, and Carl Remlinger. 2023. "Reinforcement Learning in Economics and Finance." *Computational Economics* 62 (1): 425–62. doi:10.1007/s10614-021-10119-4.

Cho, K., B. van Merrienboer, D. Bahdanau, and Y. Bengio. 2014. "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches." *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*: 103–11.

Dixon, Matthew Francis, Ivan Gvozdanovic, and Dominic O'Kane. 2023. "Time Consistent Reinforcement Learning for Optimal Consumption under Epstein–Zin Preferences. Working paper (14 March). doi:10.2139/ssrn.4388762.

Dixon, Matthew, and Igor Halperin. 2020. "G-Learner and GIRL: Goal Based Wealth Management with Reinforcement Learning." Working paper (25 February). doi:10.48550/arXiv.2002.10990.

Dixon, Matthew F., Igor Halperin, and Paul Bilokon. 2020. *Machine Learning in Finance: From Theory to Practice*. Cham, Switzerland: Springer.

Dixon, Matthew, Diego Klabjan, and Jin Hoon Bang. 2017. "Classification-Based Financial Markets Prediction Using Deep Neural Networks." *Algorithmic Finance* 6 (3–4): 67–77. doi:10.3233/AF-170176.

Ferguson, Ryan, and Andrew Green. 2018. "Deeply Learning Derivatives." Working paper (17 October). doi:10.48550/arXiv.1809.02233.

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. "Generative Adversarial Nets." *Proceedings of the 27th International Conference on Neural Information Processing Systems* 2: 2672–80.

Goodfellow, Ian. 2016. *Deep Learning: Adaptive Computation and Machine Learning*. Cambridge, MA: MIT Press.

Gradojevic, Nikola, Ramazan Gencay, and Dragan Kukolj. 2009. "Option Pricing with Modular Neural Networks." *IEEE Transactions on Neural Networks* 20 (4): 626–37. doi:10.1109/TNN.2008.2011130.

Graves, A., G. Wayne, and I. Danihelka. 2014. "Neural Turing Machines." arXiv (10 December). doi:10.48550/arXiv.1410.5401.

Graves, A., G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, and T. Ramalho. 2016. "Hybrid Computing Using a Neural Network with Dynamic External Memory." *Nature* 538 (7626): 471–76.

Halperin, Igor. 2017. "QLBS: Q-Learner in the Black–Scholes (–Merton) Worlds." Working paper (17 December). doi:10.48550/arXiv.1712.04609.

Halperin, Igor. 2024. "Distributional Offline Continuous-Time Reinforcement Learning with Neural Physics-Informed PDEs (SciPhy RL for DOCTR-L)." *Neural Computing & Applications* 36 (9): 4643–59. doi:10.1007/s00521-023-09300-7.

Halperin, Igor, and Matthew Dixon. 2020. "'Quantum Equilibrium–Disequilibrium': Asset Price Dynamics, Symmetry Breaking, and Defaults as Dissipative Instantons." *Physica A* 537 (1 January). doi:10.1016/j.physa.2019.122187.

Halperin, Igor, Jiayu Liu, and Xiao Zhang. 2022. "Combining Reinforcement Learning and Inverse Reinforcement Learning for Asset Allocation Recommendations." Working paper (6 January). doi:10.48550/arXiv.2201.01874.

Hambly, Ben, Renyuan Xu, and Huining Yang. 2023. "Recent Advances in Reinforcement Learning in Finance." *Mathematical Finance* 33 (3): 437–503. doi:10.1111/mafi.12382.

Harguess, Josh, and Chris M. Ward. 2022. "Is the Next Winter Coming for AI? Elements of Making Secure and Robust AI." In *2022 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, 1–7. doi:10.1109/AIPR57179.2022.10092230.

He, K., X. Zhang, S. Ren, and J. Sun. 2016. "Deep Residual Learning for Image Recognition." In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–78.

Hinton, Geoffrey E., and Ruslan Salakhutdinov. 2006. "Reducing the Dimensionality of Data with Neural Networks." *Science* 313 (5786): 504–07. doi:10.1126/science.1127647.

Hochreiter, S., and J. Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9 (8): 1735–80.

Horvath, Blanka, Anastasis Kratsios, Yannick Limmer, and Xuwei Yang. 2023. "Deep Kalman Filters Can Filter." Working paper (27 October). doi:10.13140/RG.2.2.22953.57445.

Horvath, Blanka, Aitor Muguruza, and Mehdi Tomas. 2021. "Deep Learning Volatility: A Deep Neural Network Perspective on Pricing and Calibration in (Rough) Volatility Models." *Quantitative Finance* 21 (1): 11–27. doi:10.1080/14697688.2020.1817974.

Huang, Guang-Bin. 2015. "What are Extreme Learning Machines? Filling the Gap between Frank Rosenblatt's Dream and John von Neumann's Puzzle." *Cognitive Computation* 7 (3): 263–78.

Huge, Brian, and Antoine Savine. 2020. "Differential Machine Learning." Working paper (30 September). doi:10.48550/arXiv.2005.02347.

Hutchinson, James M., Andrew W. Lo, and Tomaso Poggio. 1994. "A Nonparametric Approach to Pricing and Hedging Derivative Securities via Learning Networks." *Journal of Finance* 49 (3): 851–89. doi:10.1111/j.1540-6261.1994.tb00081.x.

Jaddu, Koti S., and Paul A. Bilokon. 2024. "Deep Learning with Reinforcement Learning on Order Books." *Journal of Financial Data Science* 6 (1): 61–84. doi:10.3905/jfds.2024.1.149.

Jaeger, H., and H. Haas. 2004. "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication." *Science* 304 (5667): 78–80.

Jones, Edward G. 1999. "Golgi, Cajal and the Neuron Doctrine." *Journal of the History of the Neurosciences* 8 (2): 170–78. doi:10.1076/jhin.8.2.170.1838.

Kingma, D. P., and M. Welling. 2014. "Auto-Encoding Variational Bayes." arXiv (1 May). doi:10.48550/arXiv.1312.6114.

Kohonen, T. 1990. "The Self-Organizing Map." *Proceedings of the IEEE* 78 (9): 1464–80.

Kolm, Petter N., and Gordon Ritter. 2019a. "Dynamic Replication and Hedging: A Reinforcement Learning Approach." *Journal of Financial Data Science* 1 (1): 159–71. doi:10.3905/jfds.2019.1.1.159.

Kolm, Petter N., and Gordon Ritter. 2019b. "Modern Perspectives on Reinforcement Learning in Finance." Working paper (6 September). doi:10.2139/ssrn.3449401.

Kolm, Petter N., Jeremy Turiel, and Nicholas Westray. 2023. "Deep Order Flow Imbalance: Extracting Alpha at Multiple Horizons from the Limit Order Book." *Mathematical Finance* 33 (4): 1044–81. doi:10.1111/mafi.12413.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner. 1998. "Gradient-Based Learning Applied to Document Recognition." *Proceedings of the IEEE* 86 (11): 2278–324.

Ma, Yanqing, Carmine Ventre, Renzo Tiranti, and Aiming Chen. 2025. "Deep Generative Calibration on Stochastic Volatility Models with Applications in FX Barrier Options." In *SAC '25: Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*, 122–30.

Maass, W., T. Natschläger, and H. Markram. 2002. "Real-Time Computing without Stable States: A New Framework for Neural Computation Based on Perturbations." *Neural Computation* 14 (11): 2531–60.

Makhzani, A., and B. J. Frey. 2013. "k-Sparse Autoencoders." arXiv (19 December). doi:10.48550/arXiv.1312.5663.

Malliaris, Mary, and Linda Salchenberger. 1993. "A Neural Network Model for Estimating Option Prices." *Applied Intelligence* 3 (3): 193–206. doi:10.1007/BF00871937.

McCulloch, Warren, and Walter Pitts. 1943. "A Logical Calculus of the Ideas Immanent to Nervous Activity." *Bulletin of Mathematical Biophysics* 5 (4): 115–33.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, et al. 2023. "GPT-4 Technical Report." Working paper (15 March). doi:10.48550/arXiv.2303.08774.

Pendharkar, Parag C., and Patrick Cusatis. 2018. "Trading Financial Indices with Reinforcement Learning Agents." *Expert Systems with Applications* 103 (August): 1–13. doi:10.1016/j.eswa.2018.02.032.

Rosenblatt, F. 1957. "The Perceptron—A Perceiving and Recognizing Automaton." *Cornell Aeronautical Laboratory* 85 (460-1).

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. 1986. "Learning Representations by Back-Propagating Errors." *Nature* 323 (6088): 533–36. doi:10.1038/323533a0.

Ruud, Paul Arthur. 2000. *An Introduction to Classical Econometric Theory*. New York: Oxford University Press.

Sabour, S., N. Frosst, and G. E. Hinton. 2017. "Dynamic Routing between Capsules." arXiv (26 October). doi:10.48550/arXiv.1710.09829.

Schuster, M., and K. K. Paliwal. 1997. "Bidirectional Recurrent Neural Networks." *IEEE Transactions* on *Signal Processing* 45 (11): 2673–81.

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, et al. 2016. "Mastering the Game of Go with Deep Neural Networks and Tree Search." *Nature* 529 (7587): 484–89. doi:10.1038/nature16961.

Sobakinskikh, Ilia, and Paul Alexander Bilokon. 2025. "Optimizing Transformer Neural Network for Real-Time Outlier Detection on FPGAs." *Journal of FinTech* 5 (1). doi:10.1142/S2705109925500014.

Sridi, Abir, and Paul Bilokon. 2023. "Applying Deep Learning to Calibrate Stochastic Volatility Models." Working paper (25 September). doi:10.48550/arXiv.2309.07843.

Stoiljkovic, Zoran. 2025. "Advanced Option Pricing and Hedging with Q-Learning: Performance Evaluation of the QLBS Algorithm." *Journal of Derivatives* 32 (3): 48–79. doi:10.3905/jod.2025.1.222.

Stok, Robert, Paul Bilokon, and Joseph Simonian. 2024. "From Deep Learning to Deep Econometrics." *Journal of Financial Data Science* 6 (2): 54–73. doi:10.3905/jfds.2024.1.155.

Sutton, Richard S. 2020. *Reinforcement Learning: Adaptive Computation and Machine Learning*, 2nd ed. Cambridge, MA: MIT Press.

Tsay, Ruey S. 2010. *Analysis of Financial Time Series*, 3rd ed. Hoboken, NJ: Wiley.

Varlashova, Valeria, and Paul Alexander Bilokon. 2025. "Optimal Allocation with Continuous Sharpe Ratio Covariance Bandits." *Journal of Financial Data Science* 7 (3): 171–91. doi:10.3905/jfds.2025.1.191.

Vincent, P., H. Larochelle, Y. Bengio, and P.-A. Manzagol. 2008. "Extracting and Composing Robust Features with Denoising Autoencoders." In *ICML '08: Proceedings of the 25th International Conference on Machine Learning*, 1096–103.

Yao, Jingtao, Yili Li, and Chew Lim Tan. 2000. "Option Price Forecasting Using Neural Networks." *Omega* 28 (4): 455–66. doi:10.1016/S0305-0483(99)00066-3.

Zejnullahu, Frensi, Maurice Moser, and Joerg Osterrieder. 2022. "Applications of Reinforcement Learning in Finance—Trading with a Double Deep Q-Network." Working paper (28 June). doi:10.48550/arXiv.2206.14267.

REINFORCEMENT LEARNING AND INVERSE REINFORCEMENT LEARNING: A PRACTITIONER'S GUIDE FOR INVESTMENT MANAGEMENT

Igor Halperin, PhD
Group Data Science Leader, GenAI Asset Management Technology, Fidelity Investments

Petter N. Kolm, PhD
Clinical Professor, Director of the MS in Mathematics in Finance Program, Courant Institute of Mathematical Sciences, New York University

Gordon Ritter, PhD
Adjunct Professor, Courant Institute of Mathematical Sciences and Tandon School of Engineering, New York University
Partner, Ritter Alpha, LP

Introduction

Traditional quantitative finance relies heavily on predictive models: forecasting returns, estimating volatilities, predicting default probabilities. These supervised learning approaches excel at pattern recognition but fall short when the goal is to determine *what actions to take* in dynamic, uncertain environments where decisions have lasting consequences.

Consider a portfolio manager deciding how to rebalance a multiasset portfolio. Traditional approaches might forecast expected returns and use mean–variance optimization. This static approach ignores several crucial factors, however:

- Transaction costs and market impact that depend on order size and timing
- The dynamic nature of markets where today's trades affect tomorrow's opportunities
- The need to adapt strategies as market conditions evolve
- Risk considerations that go beyond simple variance measures

Reinforcement learning (RL) is a branch of machine learning capable of addressing these issues. RL is concerned with how intelligent agents should take actions in an environment to maximize some notion of cumulative reward, as we will explain in more detail later. Unlike supervised learning, where an agent is given explicit "correct answers," an RL agent learns through experience—by trying out actions and observing the consequences.

Inverse reinforcement learning addresses the complementary problem: Given observed behavior from an expert (successful trader, market participant, or even the market itself), what underlying objectives or preferences explain that behavior? This approach is particularly

 •

valuable in finance, where true utility functions are rarely known explicitly but observed behavior is abundant.

Why RL and IRL Matter for Finance

The relevance of RL to finance stems from several key characteristics:

Sequential decision making: Most financial problems involve sequences of interdependent decisions. A trade execution strategy unfolds over time, with each trade affecting market conditions for subsequent trades. Portfolio rebalancing decisions today influence the risk–return profile tomorrow.

Uncertainty and adaptation: Financial markets are inherently uncertain and nonstationary. Successful strategies must adapt to changing conditions. RL agents can learn to recognize different market regimes and adjust their behavior accordingly.

Delayed and complex rewards: The consequences of financial decisions often manifest with significant delays and through complex causal chains. A hedging decision today may prove its worth only during a market crisis months later.

Market impact and feedback loops: In many financial contexts, an agent's actions influence the environment itself. Large trades move prices, which affects subsequent trading opportunities. This dynamic creates feedback loops that traditional static optimization cannot capture.

A Practitioner's Guide to RL Fundamentals

The RL Framework: Key Components

Understanding RL begins with its core components (Sutton and Barto 2018).

Agent

The agent is a decision maker that represents a market participant, such as a risk taker, liquidity provider, market maker, or institutional or retail trader. The agent operates (acts) over a time interval $[0, T]$, where T is the planning/acting time horizon. Most RL algorithms are formulated for discrete time steps. With a slight abuse of notation, which is quite common in the RL literature, we use the symbol t to denote both the discrete-value time and the index on the time grid, so that we can write $t = 0,1,\ldots,T$. Depending on the specific setting, T can vary between milliseconds and months or even years.

Environment

The environment is the market structure or venue where the agent operates and takes actions. In finance, this includes the following:

- Electronic limit order books (exchanges, such as NYSE and Nasdaq)
- Over-the-counter (OTC) markets for bonds and derivatives
- Alternative trading systems (dark pools, crossing networks)
- Auction mechanisms (opening/closing auctions)

- Bilateral negotiation markets (institutional block trading)
- Market-maker networks and dealer markets

State (S)

The state is a snapshot of all relevant information available to the agent at a given time. Examples include the following:

- Current portfolio positions and cash holdings
- Market data: prices, volumes, spreads, volatility measures
- Order book depth and imbalance (if observable)
- Recent price movements and technical indicators
- Time of day, calendar effects, and time to market close
- Macroeconomic indicators and news sentiment
- Risk metrics: value at risk (VaR), exposure concentrations, correlation estimates

Action (A)

Action consists of the choices available to the agent:

- Trade quantities and directions (buy/sell/hold)
- Order types and timing
- Portfolio allocation adjustments
- Risk management decisions (hedging, position sizing)

Reward (R)

Reward is the feedback signal that guides learning. The reward in RL is typically assumed to be received at each time step $t \in [0, T]$ over the course of action of the agent, where T is a time horizon. The reward at time step t is usually defined to be a function of action A_t taken at this step, as well as the current-step and next-step values of the state variable—respectively, S_t and S_{t+1}, so it is usually written as $R(S_t, A_t, S_{t+1})$. In finance, rewards typically reflect the following:

- Profit and loss (P&L)
- Risk-adjusted returns (Sharpe ratio, information ratio)
- Transaction costs and market impact
- Risk penalties (VaR, conditional value at risk, drawdown measures)

Policy (π)

The policy is the agent's strategy—a mapping from states to actions. This is what the RL algorithm learns and optimizes. Depending on the type of the specific RL algorithm, the policy can be deterministic, when the same current state always produces the same action, or it can be stochastic. In the latter case, instead of being a function, the policy is given by a probability

distribution, such that for a given state, only the probabilities of different actions, rather than actions themselves, are fixed.

By definition, the optimal policy is a policy that maximizes the total expected reward from taking actions.[1] The latter quantity is often written as follows:

$$E^{\pi}\left[\sum_{t=0}^{T} e^{-\gamma t} R(S_t, A_t, S_{t+1})\right].$$

Here, $E^{\pi}[\cdot]$ stands for the expected value assuming that policy π will be used to pick all future actions A_t. Furthermore, $\gamma \in [0,1]$ is a discount factor that specifies how much immediate rewards are more preferred over rewards received in the future. The meaning of the discount factor, γ, in RL is quite similar to the meaning of the discount factor in finance. The expected total reward defined in Equation 1 is often referred to as the *return* in the RL literature.

Example: Optimal Trade Execution

To help contextualize the foregoing information, consider an optimal execution problem in which you need to sell 100,000 shares of a stock over the next hour:

State: Remaining shares to sell (100,000 $\rightarrow$ 0), current stock price, bid–ask spread, order book depth, time remaining (60 minutes $\rightarrow$ 0), recent volatility

Actions: Number of shares to sell in the current time period (0 to remaining quantity)

Rewards: Execution price received minus a penalty for price impact and inventory risk—for example,

$$R_t = \text{Price received} - \lambda_1 \times \text{Market impact} - \lambda_2 \times \text{Inventory risk}.$$

Policy: A function that determines how many shares to sell given the current state

The RL agent learns through trial and error, experimenting with different execution rates under various market conditions, gradually improving its strategy based on the rewards received.

Mathematical Foundations: MDPs and Beyond

Markov Decision Processes

The mathematical foundation for many RL problems is the *Markov decision process* (MDP), defined by the tuple (S,A,P,R,γ):

- S = Set of possible states. States can be continuous or discrete.
- A = Set of possible actions. Actions can be continuous or discrete.
- $P(S_{t+1} \mid S_t, A_t)$ = State transition probabilities.
- $R(S_t, A_t, S_{t+1})$ = Reward function.
- $\gamma \in [0,1]$ = Discount factor for future rewards.

[1]This definition applies to the most commonly used version of RL, called the *risk-neutral RL*. Other modeling choices will be described later.

The key assumption is the *Markov property*: The future depends on only the current state, not the history of how we arrived there. This assumption allows one to define the state dynamics in terms of single-step transition probabilities $P(S_{t+1} \mid S_t, A_t)$ that reference only the current-step and next-step values of the state variable. Although the Markovian assumption may appear somewhat restrictive for financial applications, it can often be satisfied by including sufficient information in the state representation.

Partially Observable Environments

In practice, financial environments are often partially observable. Traders do not have access to all relevant information—such as other participants' intentions, unrevealed news, central bank decisions, and so on. This situation leads to *partially observable MDPs* (POMDPs), where agents receive observations O_t that provide only partial information about the true state, S_t.

Financial POMDP Example: Trading with Hidden Liquidity

Consider a trader executing a large order in a market with hidden liquidity (e.g., dark pools or iceberg orders):

- True state S_t: Total available liquidity at each price level (including hidden orders)
- Observation O_t: Only visible order book depth
- Belief state b_t: Probability distribution over possible hidden liquidity levels
- Action A_t: Order size and aggressiveness

The agent must maintain a belief state $b_t = P(S_t \mid O_{1:t}, A_{1:t-1})$ and update it using Bayesian inference:

$$b_{t+1}(s') = \frac{P(O_{t+1} \mid s')}{P(O_{t+1} \mid b_t)} \sum_s P(s' \mid s, A_t) b_t(s).$$

POMDPs are significantly more complex to solve, often requiring agents to maintain belief states (probability distributions over possible true states). However, they provide a more realistic model of financial decision making under uncertainty.

Bellman Equations: The Optimization Principle

Historically, RL grew out of dynamic programming, developed by Richard Bellman in the 1960s. Dynamic programming offers a systematic approach to the problem of sequential control that essentially relies on additivity of the RL return defined in Equation 1. With dynamic programming, the total return is viewed as a function of either the current state or a combination of the current state and the first action taken. In the former case, the total return (Equation 1) is referred to as the value function (or *V*-function), while in the latter case, it is referred to as the value-action function (or *Q*-function). The policy optimization amounts to finding an optimal policy π_* that maximizes the *V*-function or the *Q*-function. The optimal *V*- and *Q*-functions are denoted, respectively, as $V^*(S_t)$ and $Q^*(S_t, A_t)$. These functions satisfy recursive relations known as Bellman optimality equations:

$$V^*(s) = \max_a \left[\mathbb{E}[R_{t+1} \mid S_t = s, A_t = a] + \gamma \sum_{s'} P(s' \mid s, a) V^*(s') \right].$$

$$Q^*(s,a) = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a] + \gamma \sum_{s'} P(s' \mid s,a) \max_{a'} Q^*(s',a').$$

These equations state that the optimal value of a state (or state-action pair) equals the expected immediate reward plus the discounted expected value of the best possible next state.

The classical dynamic programming typically focuses on solution of the first Bellman optimality equation for the *V*-function for low-dimensional and discrete sets of states and action, while assuming that the dynamics of the transitions between states are known. Such constraints on the dimensionality of a state-action space and the need for an explicit model of the environment presented severe limitations for the use of dynamic programming for many real-life applications where the dimensionality of the state-action space is high and where the explicit model of the world is typically not available or hard to estimate.

Respectively, one approach of RL extends the dynamic programming formulation of sequential decision-making problems to models with a high-dimensional continuous or discrete state-action space, without assuming that dynamics of the world are known. Instead, this approach, known as the *value-based RL*, relies on samples from data obtained from interactions of an agent with its environment, giving rise to sample-based solutions of Bellman optimality equations.

Value-Based RL vs. Policy-Gradient vs. Actor-Critic Methods

The value-based RL that uses value or action-value functions together with Bellman equations is not the only available approach to RL. Other approaches exist that do not rely on Bellman equations but, rather, directly optimize a parameterized policy to maximize the total return defined in Equation 1. These methods are collectively known as *policy-gradient methods*. The most basic policy-gradient method, REINFORCE, is very simple to implement but has the drawback of producing high variance for total returns (Sutton and Barto 2018). Finally, with actor-critic RL methods, two different parameterized functions are used to represent the value function and the policy function. Actor-critic methods produce lower variance of total returns than pure policy-gradient methods.

RL Algorithm Landscape

Classical Algorithms

Perhaps the most famous RL algorithm is the value-based RL algorithm called *Q-learning*. It learns the optimal action-value function, $Q^*(s,a)$, through temporal difference updates:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[R_{t+1} + \gamma \max_{a'} Q(s_{t+1},a') - Q(s,a)\right].$$

The Q-learning algorithm can be interpreted as a sample-based solution of the Bellman optimality equation for the Q-function. For discrete state-action systems, values of the Q-functions for different combinations of the state and action can be stored in a table in which rows and columns correspond to different values of the state and action, while the Q-update equation shown here is used to update the values in the table upon observing a new state and rewards from taking a certain action in the current state. This process is referred to as *tabular Q-learning*. Q-learning is considered "off-policy," meaning it can learn the optimal policy while

following a different (exploratory) policy. A different form of tabular learning is presented by the SARSA algorithm, an "on-policy" alternative that updates according to the action actually taken:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s,a)].$$

From Tabular to Deep Learning: Function Approximation and Deep RL

Financial state spaces are typically enormous or continuous (asset prices, portfolio weights, market indicators). *Function approximation* addresses this situation by learning parameterized functions that approximate value functions (for value-based RL methods), policies (for policy-gradient methods), or both (for actor-critic methods). Although many machine learning approaches (e.g., trees) are able to produce universal function approximation methods, today, various versions of artificial neural networks serve as the most popular function approximation approach.

Among value-based deep RL methods, *deep Q-networks* (DQNs) are most widely known. DQNs combine Q-learning with deep neural networks, enabling RL to handle high-dimensional state spaces. In addition to using deep neural networks for value function approximation, DQNs introduce other innovations that improve their training. In particular, they use experience replay, which amounts to storing and randomly sampling past experiences for training. The other innovations are target networks: separate networks for stability during training.

With policy-gradient methods, function approximations are used to approximate policies (for pure policy-based approaches, such as REINFORCE) or both value and policy functions (for actor-critic methods). Function approximations for both functions are also used in proximal policy optimization (PPO) algorithms—a policy-gradient method that uses a clipped objective function to ensure that policy updates stay within a certain "trust region," which makes training more stable.

Model-Based vs. Model-Free RL

One of the main paradigms of RL is its reliance on learning directly from experience, without modeling environment dynamics. This is commonly referred to as *model-free RL*. Examples of model-free RL methods include Q-learning and policy gradients, outlined previously. Model-free RL approaches have both pros and cons:

- *Pros:* Simple to implement, effective when dynamics are unknown/complex
- *Cons:* Sample inefficient, requires many interactions

An alternative to model-free RL is *model-based RL*. With this approach, one first learns a model of environment dynamics and then uses this model for planning. In a sense, model-based RL takes us one step back to the setting of dynamic programming, which likewise assumes that the dynamics of the environment are known. Model-based RL methods have their own pros and cons:

- *Pros:* More sample efficient, enables planning and scenario analysis
- *Cons:* Possibility of model errors compounding, more complex to implement

For financial applications, the choice between model-free and model-based RL depends on the specific problem. High-frequency trading with complex, fast-changing dynamics might favor model-free approaches, whereas strategic asset allocation could benefit from model-based methods.

Online vs. Offline RL

Another important distinction between different RL algorithms concerns how they are trained. In online RL methods, an agent learns by taking actions and observing rewards received from its environment. Such settings are very common in applications of RL in robotics. The important question faced by the agent is how to optimally combine taking actions that have previously shown good reward versus trying new actions whose rewards will be observed only after trying them. This problem is known as the exploration–exploitation dilemma of RL (Sutton and Barto 2018).

A different setting is provided by offline RL. In this formulation, the agent learns the optimal policy without a direct interaction with its environment but, rather, using historical data collected from previous interactions of another (or the same) agent with the same environment. In this setting, the agent can no longer rely on trial-and-error methods to find optimal actions: The exploration–exploitation dilemma was already addressed (potentially not optimally!) by the previous agent. As a result of its inability to explore different actions in interaction with its environment, offline RL is generally harder than online RL. However, the setting of offline RL closely matches the setting of classical financial models that are typically fit to fixed datasets consisting some historical data.

Risk-Aware RL: Beyond Expected Returns

The total return given by the sum of all future reward is clearly a random quantity as seen at the start of agent's action (at time 0) because it depends on future states and actions that are not yet known at time 0. This situation is quite similar to how the future return of an investment portfolio is a random quantity at the initial time of portfolio initiation. However, the standard RL optimizes only the expected cumulative reward (i.e., the mean of this distribution) and does not try to control its higher moments, such as variance, kurtosis, or tail risk measures. Again similar to classical financial settings, this condition is often not sufficient for the practical purposes of using RL for financial applications, where we want to control not only the mean future total reward but also risk, or variability around this mean, as expressed, for example, by the variance of the total reward. Because standard RL does not address risk/uncertainty around the mean expected total reward, it is sometimes referred to as *risk-neutral RL*.

Because financial decision making is inherently risk sensitive, risk-neutral RL is often inadequate for this task, especially if pure P&L (or return) is taken as the reward for RL. Several extensions address this limitation.

Risk-Sensitive RL

Mean–variance RL incorporates both expected return and variance into the reward:

$$\text{Reward} = \mathbb{E}[\text{Return}] - \lambda \times \text{var}[\text{Return}].$$

In this approach, one uses a risk-adjusted return as a reward function, very much in the spirit of the classical Markowitz portfolio theory. The attractive property of this approach is that it still can use the risk-neutral RL formulation, where the variance penalty is simply added to the definition of the reward such that we still optimize its expected (mean) value. The other attractive feature of this specification is that the reward defined by this relation is a quadratic function of actions, which tremendously simplifies the RL algorithm and in fact enables its implementation without using any neural networks altogether (Dixon, Halperin, and Bilokon 2020). The known drawback of using variance as a risk measure is that it penalizes both negative and positive returns, while ideally we may want to penalize only negative returns.

By using *conditional value at risk* (CVaR) as the risk penalty component of the reward function, RL methods with such rewards directly optimize tail risk measures, which is crucial for downside protection. Unlike mean–variance RL, which can proceed without using neural nets, RL methods that use CVAR typically need to use deep neural networks as function approximation tools.

Risk-constrained RL methods are similar to CVAR-based approaches. They maximize expected returns subject to risk constraints, such as maximum drawdown or VaR limits.

Distributional RL

Instead of learning expected values, *distributional RL* learns the full probability distribution of cumulative future rewards. Having access to the full distribution of the total reward allows one to compute any risk measure (e.g., VaR, CVaR, skewness) for the ultimate decision making. Such algorithms as quantile regression DQN (QR-DQN) and implicit quantile networks (IQNs) represent the return distribution using quantiles, making them particularly suitable for financial applications focused on tail risks. Distributional RL can also be constructed with the continuous time formulation. With this approach, policy optimization amounts to a numerical solution of certain partial differential equations (Halperin 2024).

Use Cases for RL Applications in Finance

Here, we provide a brief outlook for various use cases for RL in financial applications. Without attempting a detailed presentation, we focus here on the few key elements, including specifications of state, action, and reward, as well as outlining implementation consideration.

Optimal Trade Execution

Problem Setting

A fundamental problem for any large institutional investor is how to execute a large order with minimal price disruption. Standard benchmarks such as time-weighted average price or volume-weighted average price provide simple, static schedules but fail to adapt to changing market conditions during the execution horizon. An RL agent, in contrast, can learn a dynamic policy that breaks a large "parent" order into a sequence of smaller "child" orders, adapting the pace of trading to minimize costs. The core challenge is to balance the trade-off between the *market impact cost* of trading aggressively and the *timing risk* of trading slowly in a volatile market.

RL Formulation

The problem is naturally framed as a finite-horizon MDP. The *state space* typically includes the remaining quantity to be traded, the time left in the execution window, current asset price, and potentially microstructure features, such as order book imbalance or recent volatility. The *action* is the size of the child order to submit in the current time slice. The *reward function* is crucial and must encapsulate the trade-off: It is often formulated as the negative of implementation shortfall or, more explicitly, the execution price achieved, penalized by terms representing price slippage resulting from market impact and the risk exposure of the remaining inventory.

Implementation Considerations

A significant challenge is developing a realistic market impact model, which can be either estimated offline from historical data or learned online as part of the RL agent's interaction. Furthermore, a robust execution agent must be able to generalize across different market volatility regimes and asset-specific behaviors. For training and validation, a high-fidelity market simulator that can accurately model price dynamics and market impact is indispensable because training in a live market is impractical and costly. Finally, the learned policy must operate within the constraints of regulatory requirements, such as "best execution" mandates.

Dynamic Portfolio Optimization

Problem Setting

Classical single-period portfolio optimization, such as Markowitz's mean–variance framework, is static and highly sensitive to estimation errors in its inputs (expected returns and covariances). Its static nature prevents optimal portfolio allocation decisions in terms of minimization of transaction costs or optimal use of multihorizon predictive signals. Dynamic portfolio optimization extends the static mean–variance framework to a multiperiod setting, where an RL agent learns a rebalancing policy over a long horizon. This approach aims to maximize cumulative risk-adjusted returns while accounting for real-world frictions, such as transaction costs and market impact.

RL Formulation

The *state space* for a dynamic portfolio agent includes the current portfolio weights, recent asset returns, market regime indicators (e.g., from a hidden Markov model), and predictive macroeconomic factors. The *action space* is the set of target portfolio weights for the next period. Because weights are continuous variables, this problem is ill suited for tabular RL methods and requires function approximation. The *reward* is typically a risk-adjusted return metric, such as the period's Sharpe ratio or a utility function of the portfolio's return, net of transaction and holding costs.

Implementation Considerations

The continuous and high-dimensional nature of the state and action spaces makes this application of RL challenging in practice. Advanced techniques are often used in this setting. In particular, with hierarchical RL approaches, the problem can be decomposed into a high-level strategic allocation agent that sets broad targets over long horizons (e.g., quarterly) and a low-level tactical agent that makes finer adjustments (e.g., monthly) to achieve those targets. Alternatively, multi-objective RL can be considered to navigate the complex trade-offs between competing

goals, such as maximizing returns, minimizing volatility, and reducing portfolio turnover to limit transaction costs. Lastly, a computationally efficient and noise-robust approach particularly helpful for financial portfolio optimization tasks is provided by G-learning, a probabilistic extension of Q-learning. This method was applied to goal-based wealth management by Dixon and Halperin (2020).

Option Pricing and Hedging

Problem Setting

The classical Black–Scholes–Merton model (Black and Scholes 1973; Merton 1974) provides a cornerstone for derivative pricing but relies on idealized assumptions, such as continuous and costless hedging, that do not hold in practice. When hedging is discrete and involves transaction costs, the problem of pricing and hedging a derivative becomes a sequential decision-making problem under uncertainty. The goal is to design a dynamic hedging strategy that minimizes a risk-adjusted measure of the total hedging cost. Several RL-inspired frameworks have been developed to address this problem, moving beyond the classical risk-neutral paradigm.

RL Formulation and Approaches

Two main classes of methods have emerged, which can be broadly understood as value-based and policy-based approaches to the hedging problem.

Value-Based RL

The QLBS Model proposed by Halperin (2020) directly applies the principles of value-based RL. The problem is framed as an MDP where the agent (an option seller) seeks to learn an optimal hedging policy. The *state* is defined by the underlying asset price and time to expiration. The *action* is the hedge adjustment (the amount of the underlying to hold). The *reward function* is defined as the negative of a risk-adjusted hedging cost. For tractability, this cost takes a quadratic form (mean–variance utility), including a penalty for the variance of the hedge portfolio's value, which directly incorporates the hedger's risk aversion. The agent learns an optimal action-value function (Q-function), from which both the optimal hedge policy and the corresponding option price (the negative of the Q-value) are derived simultaneously.

Deep value-based RL for option hedging is a more end-to-end approach that has been explored. Du, Jin, Kolm, Ritter, Wang, and Zhang (2020) applied state-of-the-art DRL algorithms, such as proximal policy optimization (PPO), to learn hedging strategies directly from market simulations. A key advantage of their framework is its ability to handle practical market frictions, such as discrete trading times and nonlinear transaction costs. Furthermore, their approach is highly efficient because a single trained DRL agent can learn to hedge a whole range of option strikes simultaneously, eliminating the need for retraining on a per-strike basis. The authors demonstrated that the DRL agent can learn strategies that match or outperform traditional delta hedging in terms of profit and loss.

Direct Policy Optimization

An alternative framework is *deep hedging*, pioneered by Buehler, Gonon, Teichmann, and Wood (2019). This approach can be seen as a form of direct policy optimization. Instead of solving

the recursive Bellman equation, it frames the entire hedging problem as a single end-to-end optimization. A neural network is used to directly represent the hedging policy, taking the current market state as input and outputting the hedge position. The training process involves the following:

- Simulating a large number of market scenarios (paths) for the underlying asset(s)
- For each path, applying the hedging strategy defined by the current neural network
- Calculating the final P&L distribution of the fully hedged portfolio across all paths
- Using backpropagation to update the neural network's weights to minimize a chosen risk measure of this final P&L distribution (e.g., CVaR, mean–variance utility, or expected shortfall)

Implementation Considerations

All RL approaches directly address the limitations of the classical model by embedding real-world frictions, such as *discrete hedging* and *transaction costs*. However, they differ in their philosophy and implementation. The QLBS approach aligns closely with traditional RL theory (MDPs, Bellman equations, Q-functions) and can be computationally efficient when its quadratic reward assumption leads to semi-analytical solutions, as explored in fitted Q-iteration (Ernst, Geurts, and Wehenkel 2005). Also, because Q-learning is a model-independent method, the QLBS approach amounts to a model-independent and data-driven option hedging and pricing method. Deep hedging, while conceptually related, is implemented as a global optimization that is very flexible; changing the risk objective simply means changing the final loss function. It is inherently a model-based approach because it learns the optimal policy for a given simulation model of the market (e.g., Heston, SABR). The power of neural networks also allows it to handle very complex and non-Markovian state representations.

End-to-End Deep RL for Asset Allocation

Problem Setting

Traditional quantitative asset allocation involves a two-step process: First, predict expected returns and covariances, and second, use these predictions in an optimization framework (e.g., mean–variance). This process is fragile because performance is highly sensitive to errors in the initial prediction step. A more robust approach would be to learn the mapping from market data to portfolio weights in an end-to-end fashion.

RL Formulation and Approaches

Noguer i Alonso and Srivastava (2020) proposed a model-free, end-to-end deep reinforcement learning approach that bypasses the explicit forecasting step. The RL agent learns a direct mapping from raw market data to optimal portfolio weights.

- *State:* The state is represented as a tensor of recent price history (e.g., 50 days of high, low, and close prices for a universe of stocks).
- *Agent:* The agent is a deep neural network (the authors tested various architectures, such as CNNs, RNNs, and LSTMs) that learns the allocation policy.

- *Action:* The action is an output vector of portfolio weights for the next period.
- *Reward:* The reward is a simple and direct financial objective, such as the portfolio's average logarithmic return, adjusted for transaction costs. To encourage stable portfolios, the architecture incorporates a "portfolio vector memory," which considers past weights when determining new ones and implicitly penalizes high turnover.

Implementation Considerations

This end-to-end framework learns to predict and optimize simultaneously. Noguer i Alonso and Srivastava (2020) showed it can construct portfolios that outperform traditional methods, such as mean–variance optimization and risk parity, even when only given raw price series as input. The use of a portfolio memory is a key architectural choice to control for turnover, which is a critical practical consideration.

Algorithmic Trading

Problem Setting

Algorithmic trading seeks to automate trading decisions to capitalize on market opportunities more efficiently than human traders can. The core challenge is to develop a strategy that can adapt to changing market conditions and dynamically balance the trade-off between expected returns and various forms of risk.

RL Formulation and Approaches

The problem is naturally framed as an MDP where an agent learns an optimal trading policy.

- *State space:* This includes such features as historical price data, technical indicators (e.g., moving averages), market volatility, and the agent's current portfolio state (cash and asset holdings).
- *Action space:* This consists of such actions as buying, selling, and holding assets. The action can be discrete (e.g., buy one unit) or continuous (e.g., allocate 15% of capital).
- *Reward function:* The reward function is typically defined to reflect a risk-adjusted return measure. A simple profit-and-loss reward can be augmented with risk measures, such as a penalty for high portfolio variance, large drawdowns, or a direct optimization of the Sharpe ratio.

A comprehensive review by Pricope (2021) considered the application of deep RL to this problem, noting that many studies show statistically significant outperformance over simpler baselines in simulated environments.

Implementation Considerations

A key challenge highlighted in the literature is the gap between simulated performance and real-world applicability. Many studies are proofs of concept conducted in environments that do not fully capture real-time market frictions, latency, and data imperfections. Successful implementation requires careful consideration of transaction costs, market impact, and robust out-of-sample validation.

Market Making

Problem Setting

Market making is a high-frequency strategy in which a trader provides liquidity by simultaneously placing bid and ask limit orders, aiming to profit from the spread. The main challenge is to set optimal quotes that balance maximizing spread capture against managing two key risks: inventory risk (holding a large, undiversified position) and adverse selection risk (trading against an informed counterparty who knows where the price is headed).

RL Formulation and Approaches

The market making problem is well suited to an MDP formulation where the agent learns an optimal quoting strategy.

- *State space:* This encompasses the agent's current inventory, features of the limit order book (e.g., bid–ask spread, volume imbalance), and market indicators, such as volatility.
- *Action space:* This includes setting the bid and ask quotes relative to the market midpoint and deciding the size of the orders to be placed.
- *Reward function:* Carefully designed, this function rewards captured spreads while penalizing inventory risk and losses from adverse selection.

Spooner, Fearnley, Savani, and Koukorinis (2018) presented a foundational example in which a deep RL agent learns to perform market making. Their agent learns a value function to optimize quotes, demonstrating that an RL approach can outperform traditional stochastic control-based strategies in simulated environments.

Implementation Considerations

The primary challenge in applying RL to live market making is latency. High-frequency environments require decisions on microsecond timescales, which can be difficult for complex neural network models to meet. Furthermore, creating a high-fidelity market simulator that accurately captures order flow dynamics and adverse selection is a significant undertaking yet is crucial for training a robust agent.

Inverse Reinforcement Learning: Inferring Hidden Objectives

Reinforcement learning learns optimal policies given known objectives, as codified by rewards. The reward in RL is observable, as a result of online interaction of an agent with its environment or for offline RL, as a part of historical data. In many real-world problems, however, we observe only agents' actions—not their reward. Arguably, in real life, such scenarios are encountered more often relative to scenarios where rewards are observed. The ultimate objective of learning in this setting is still the same as in the RL scenario—that is, to learn the optimal policy by observing the agent's behavior. In contrast to the standard RL scenario, however, we now observe only the states of the environment and actions taken by the agent and do not observe rewards received by the agent.

Clearly, without additional assumptions, such problems do not have a solution. For example, if an agent's actions are purely random, not much (if anything) can be learned from such observations. If we assume that the agent's actions were supposed to achieve some objectives, however, then we can address the inverse problem: Given observed behavior, what objectives explain that behavior?

If the objective is to maximize the total expected reward, this inverse problem formulation gives rise to *inverse reinforcement learning* (IRL). IRL uses the observed behavior of an agent and infers the reward function that is assumed to be optimized by the agent. Note that in most IRL applications, such inference of the reward function is *not* the end goal. The end goal is rather the same as in conventional (direct) RL, which is to find the optimal policy that maximizes the total expected reward. Unlike the task of RL, however, where rewards are a part of the inputs, in IRL we first have to infer them from the behavior, and then we use them to find the optimal policy.

The Challenge of IRL: An Ill-Posed Problem

As with many inverse problems, IRL is an *ill-posed problem*: For any given set of observed behaviors, an infinite number of reward functions could explain that behavior. For instance, a policy of doing nothing is optimal for a reward of zero but also for any reward function that depends only on the state, not the action. This is the issue of *reward shaping invariance*: an optimal policy is unchanged if we transform the reward function by adding a potential-based term (Ng, Harada, and Russell 1999).

In order to have a solution, one needs to make additional assumptions about the agent in order to select the "best" or most plausible reward function from the many possibilities. One such assumption can be that the agent's behavior is optimal or close to optimal. Although such assumptions may be reasonable in some applications (e.g., robotics), it is hardly appropriate in finance, where optimality does not even exist in absolute terms and can be defined only relative to some benchmark.

IRL vs. Imitation Learning

The other question that can be asked in relation to the declared objective of IRL is, Why do we need to first infer the reward, as long as we assume that the observed behavior is already optimal? Can we simply build a supervised learning–type model that would simply mimic the observed behavior of an agent in different states of the world, without even asking a question about the agent's reward function? It turns out that such strategies are indeed feasible in certain circumstances, and the subfield of machine learning that studies these methods is known as imitation learning (IL). Although IL and IRL have the shared final goal of finding optimal policy from the observed behavior, they differ in the intermediate steps. IRL infers the reward function as the intermediate step, while IL proceeds without it. In general, IRL methods often work better than IL methods, and they offer more flexible and portable solutions because a reward function offers a succinct description of agents' goals that is portable across different environments (Dixon et al. 2020). Therefore, we will mostly focus on IRL methods for financial applications.

Key IRL Approaches

To deal with its ill-posed nature, IRL methods impose additional principles to find a unique reward function.

Maximum Entropy IRL

The *maximum entropy (MaxEnt) IRL* principle is a popular approach to regularize the problem (Ziebart, Maas, Bagnell, and Dey 2008).

- *Principle:* Among all reward functions that explain the expert's behavior, choose the one that makes the expert's policy as random as possible (i.e., maximizes its entropy). The intuition is to match the observed behavior while being maximally non-committal about behavior that hasn't been observed.
- *Probabilistic policy:* This principle naturally leads to a stochastic policy in which the probability of taking an action is exponentially proportional to its value. This is often called a Boltzmann or softmax policy.
- *Implementation:* This approach turns IRL into a maximum likelihood problem on the observed expert trajectories. The main computational challenge is often calculating the normalization constant (partition function) for this policy distribution, which requires summing or integrating over all possible actions at each step.

Bayesian and Gaussian Process IRL

Another way to handle the ill-posed nature of IRL is through a Bayesian lens. Instead of seeking a single best-fit reward function, Bayesian IRL aims to find a posterior distribution over all plausible reward functions, given the observed expert data.

Gaussian process IRL (GPIRL) is a specific and powerful nonparametric implementation of this idea (Levine, Popović, and Koltun 2011).

- *Principle:* A Gaussian process (GP) is placed as a prior over the unknown reward function. A GP can be thought of as a "distribution over functions." It provides a flexible way to represent the belief that the reward function is likely to be smooth, without having to specify its exact functional form (e.g., linear or quadratic).
- *Learning process:* Starting with this GP prior, the algorithm observes the expert's state-action trajectories. It then uses Bayesian inference to update the prior, resulting in a posterior distribution over reward functions that are consistent with the observed behavior.
- *Benefits:* The main advantage is flexibility. GPIRL can capture complex, nonlinear reward functions without manual feature engineering. This ability makes it particularly suitable for financial applications for which the relationship between market states and a trader's implicit rewards can be highly nuanced. This method was notably used in the financial applications discussed later.

Adversarial Imitation Learning

A different and powerful class of methods frames imitation learning as a two-player game.

- *Generative adversarial imitation learning (GAIL):* Ho and Ermon (2016) proposed GAIL, which does not explicitly recover a reward function. It trains a *generator* (the agent's policy) to produce state-action trajectories that are indistinguishable from an expert's trajectories, as judged by a *discriminator*. The discriminator is trained simultaneously to tell the difference between the agent's and the expert's behavior. GAIL is pure imitation learning.

- *Adversarial inverse reinforcement learning (AIRL):* Finn, Christiano, Abbeel, and Levine (2016) extended this framework by structuring the discriminator in a specific way. In AIRL, the discriminator's output can be decomposed to recover not only a policy but also a reward function. This approach elegantly connects adversarial learning back to the original goal of IRL.

T-REX: Learning from Ranked Demonstrations

Standard IRL often assumes expert demonstrations are (nearly) optimal. This can be a strong and often incorrect assumption. *Trajectory-ranked reward extrapolation (T-REX)* offers a powerful alternative by learning from demonstrations of varying quality (Brown, Goo, Nagarajan, and Niekum 2019).

- *Principle:* Instead of a set of optimal demonstrations, T-REX uses a set of trajectories that have been *ranked* by preference (e.g., "trajectory A is better than B"). It does not require knowing the absolute quality, only the relative ranking.
- *Learning intent:* The objective is to learn a reward function such that the total reward assigned to each trajectory is consistent with the given ranking. By learning what makes one trajectory better than another, T-REX can infer the underlying *intent* of the demonstrator.
- *Surpassing the teacher:* Because it learns an underlying reward function rather than simply mimicking actions, the learned reward can be used with a standard RL algorithm to find a policy that is even better than the best demonstration provided. This is a crucial step toward building agents that can learn from and improve on human behavior.

Inverse Reinforcement Learning in Action: Financial Use Cases

IRL opens up new avenues for analyzing financial behavior and markets. In this section, we present a short and nonexhaustive overview of applications of IRL in the financial domain.

Algorithmic Trading Strategy Identification

- *Problem:* High-frequency trading (HFT) firms use a diverse set of strategies. Regulators and market operators are interested in identifying and clustering these strategies from observable order data to monitor market health and detect manipulative behavior. Standard clustering based on statistical features of trading activity (e.g., order-to-trade ratio) can be crude and may not capture the underlying objectives.
- *IRL approach:* Yang, Qiao, Beling, Scherer, and Kirilenko (2015) pioneered an approach using Bayesian IRL (specifically, Gaussian process IRL). Their approach treats HFT strategies as "experts" and uses their observed order placements (actions) in the limit order book (state) to infer the reward function each strategy is optimizing. By clustering strategies based on the parameters of their learned reward functions, they achieved more meaningful and interpretable groupings of behavior than by using simple statistical features. The reward function captures the agent's implicit trade-offs between, for instance, aggressive execution and inventory risk.

Sentiment-Based Trading Strategies

- *Problem:* Build a trading system that systematically exploits the relationship between investor sentiment and market dynamics.
- *IRL approach:* Yang, Yu, and Almahdi (2018) framed the problem using GPIRL. Their approach treats aggregate news sentiment as the "action" of a single, collective market agent. The market state is defined by recent price dynamics, and GPIRL is used to infer the reward function that this collective agent is maximizing. This learned reward function, which implicitly captures how the "market" values taking bullish or bearish actions given certain conditions, can then be used by a direct RL agent to make its own trading decisions.

Inferring Customer Preferences in Consumer Finance

- *Problem:* Businesses offering subscription or recurring utility services (e.g., mobile data plans, cloud computing, energy) need to understand customer behavior to design better products and pricing plans. Customer decisions, such as daily consumption, are sequential and depend on the current state (e.g., remaining quota, days left in the billing cycle).
- *IRL approach:* This problem can be framed as inferring a customer's latent utility (reward) function from their observed consumption patterns. A MaxEnt IRL algorithm for this problem was proposed by Dixon et al. (2020). It uses a parametric reward function that captures the trade-offs a customer makes, including the utility of consumption, a penalty for exceeding a quota (and paying an overage price), and a potential reward for forgoing consumption. By observing a customer's consumption history, the MaxEnt IRL algorithm finds the utility parameters that make the observed behavior most probable.

Once this customer-specific utility function is learned, the firm can perform powerful counterfactual simulations. For example, it can predict how that customer's consumption would change if the monthly price were lowered or the data quota were increased. This ability provides a principled, data-driven method for product design and targeted marketing that goes far beyond simple statistical analysis.

Goal-Based Wealth Management and Robo-Advising

- *Problem:* A core challenge in wealth management is optimizing a client's portfolio over a long horizon to meet a specific goal, such as funding retirement. This problem differs from standard portfolio optimization because it involves periodic cash flows (contributions during the accumulation phase, withdrawals during decumulation) in addition to asset rebalancing. The objective is often to reach a target wealth level, a more intuitive goal for retail investors than maximizing a Sharpe ratio or tracking a mean–variance-efficient frontier.
- *IRL–RL approach:* Dixon and Halperin (2020) proposed a two-part framework to tackle this problem.
    1. *G-learner (the RL agent):* First, they define a direct RL agent, the *G-learner*, which solves the goal-based wealth management problem. The G-learner uses a specific quadratic reward function that penalizes underperformance relative to a target wealth path and accounts for transaction costs. By defining actions as absolute dollar changes in asset positions, it handles cash flows naturally and scales to high-dimensional portfolios.

This G-learner serves as a powerful, computationally tractable solver for the direct RL problem.

2. *GIRL (the IRL method):* The second part, *GIRL (G-learning IRL)*, addresses the inverse problem. Many investors cannot explicitly define their reward function parameters (e.g., their exact risk aversion or how they weigh tracking a benchmark versus growth). GIRL takes the observed trading history of an investor (or a human portfolio manager) and infers the most likely reward function parameters that the investor was implicitly optimizing. It assumes the investor behaves like a G-learner and uses maximum likelihood to find the reward parameters that best explain the observed actions.

- *Application to robo-advising:* The combination of these two algorithms creates a powerful tool for robo-advising. GIRL can be used to learn the implicit reward functions (i.e., the investment "styles" and risk preferences) of successful human portfolio managers. This learned "best-in-class" reward function can then be given to the G-learner, which computes a new, enhanced optimal policy. This process creates a system that can learn from human expertise, formalize it, and then use AI to find an even better strategy, providing superior, data-driven recommendations.

Learning Optimal Asset Allocation from Collective Behavior of Fund Managers

- *Problem:* How can we learn from the collective behavior of a group of active fund managers to provide improved asset allocation recommendations? Although individual managers are experts, their decisions can contain noise or suboptimal biases. A method is needed to distill their collective wisdom while filtering out individual errors.
- *IRL–RL approach:* Halperin, Liu, and Zhang (2022) proposed a practical two-step framework that combines IRL and RL to learn from and improve on the investment practices of a group of fund managers.

  Step 1: Infer collective intent (IRL): The framework first takes the historical trading data from a group of fund managers with similar investment mandates (e.g., large-cap growth funds). The historical performance of these funds is used to rank their trajectories. Using the T-REX algorithm, the system learns a single, shared reward function whose parameters are optimized to be consistent with these performance rankings. This step infers the *collective intent* of the group—what objectives, on average, lead to better performance within this peer group.

  Step 2: Optimize policy (RL): The collective reward function learned in the IRL step is then passed to a direct RL agent (the G-learner). This agent solves for the optimal asset allocation policy that maximizes this reward function. Because the reward function is based on the distilled wisdom of the entire group, the resulting policy is often superior to the strategies of the individual managers it learned from.
- *Application as an assistant:* This framework is designed not to replace portfolio managers but to *assist* them. The output of the RL step is a set of recommended asset allocation changes (e.g., reweighting portfolio exposure across industry sectors). Managers can use these recommendations as a data-driven input to refine their own decisions, leveraging the collective intelligence of their peers to improve performance. This demonstrates a practical

human–machine interaction loop, where IRL learns from human experts and RL provides optimized suggestions to them.

High-Frequency Market Making with Imitation Learning

- *Problem:* Traditional HFT models for market making, such as the Avellaneda–Stoikov model, are often calibrated on historical data and make strict assumptions about market dynamics (e.g., stable order flow, specific price processes). These models struggle to adapt when real-world market conditions diverge from these assumptions. Standard RL approaches, on the other hand, can be sample-inefficient and often optimize myopically for single-step actions, which can lead to compounding errors and poor inventory management in HFT.
- *Imitation learning approach (FlowHFT):* To address these challenges, Li, Chen, and Yang (2025) proposed FlowHFT, a novel framework based on imitation learning. Instead of assuming a single expert model is best for all conditions, FlowHFT learns from a diverse set of expert demonstrations. It simulates various market scenarios (e.g., high/low volatility, trending/mean-reverting) and identifies the best-performing traditional model (e.g., AS, GLFT) for each specific scenario.
- *Flow-matching policy:* The core of the framework is a flow-matching policy. This is a sophisticated generative model that learns to map a market state to a sequence of optimal trading actions. It does so by learning a "flow" that transforms a simple noise distribution into the complex distribution of expert actions observed across all market scenarios. This process allows a single, adaptive model to integrate the knowledge of many specialized experts. Crucially, it learns to generate entire action *sequences* over a planning horizon, which inherently considers the near-term consequences of actions and helps mitigate the compounding errors seen in single-step RL.
- *Application:* The trained FlowHFT model can adaptively generate trading decisions suitable for the prevailing market state, effectively leveraging the best strategy from its library of learned experts. Li et al. (2025) showed that their single framework can consistently outperform the best individual expert model in each tested market condition, demonstrating a powerful application of imitation learning to build robust, adaptive HFT agents.

Computational Requirements and Infrastructure: Hardware and Software Ecosystem

Implementing RL for financial applications requires careful consideration of computational infrastructure:

Hardware requirements

- *Development phase:* GPU-enabled workstations (NVIDIA RTX 3090 or better) for deep RL algorithms
- *Production phase:* Low-latency inference servers for real-time decision making
- *Memory requirements:* 32GB+ RAM for experience replay buffers in high-frequency applications

Software stack

- *RL frameworks:*
  - Stable Baselines3: Production-ready implementations of standard algorithms
  - RLlib: Distributed training for large-scale applications
  - TF-Agents/PyTorch RL: For custom algorithm development
- *Market simulators:*
  - ABIDES: Agent-based market simulator for microstructure research
  - FinRL: Integrated environment for financial RL applications
  - Custom simulators using historical tick data

Challenges and Frontiers

Although RL and IRL hold immense promise for finance, practitioners must navigate a set of significant challenges before these methods can be widely and reliably deployed.

Key Challenges

The following are some of the main challenges for RL and IRL:

- *Sample efficiency and data requirements:* RL agents, particularly model-free ones, often learn through extensive trial and error. Financial data, although vast, can be noisy, and the number of truly independent historical scenarios is limited. This situation makes it difficult to train agents that are robust to rare but critical market events, such as financial crises.
- *Nonstationarity of financial markets:* The core assumption of a stationary MDP is often violated in finance. Market dynamics, volatility regimes, and correlation structures evolve over time. A policy learned on historical data may become suboptimal or even detrimental when market conditions change. This dynamic necessitates continuous learning or adaptive models that can detect and adjust to regime shifts.
- *Fidelity of simulation environments:* Training and validating RL agents, especially for high-stakes applications, require a realistic market simulator. For example, for applications for trading in the limit order book, building a simulator that accurately captures market micro-structure, order flow dynamics, latency, and the feedback loop of market impact is an extremely challenging problem in itself. An agent that performs well in a flawed simulation may fail spectacularly in a live market.
- *Reward specification and risk sensitivity:* For direct RL, defining a reward function that perfectly aligns with a long-term financial objective is nontrivial. A myopic reward (e.g., single-period profit) can lead to undesirable behavior, such as excessive risk taking. As discussed, standard RL optimizes for expected returns (risk neutrality), whereas financial applications almost always demand explicit management of risk (e.g., variance, tail risk, drawdown). Risk-sensitive and distributional RL can therefore present a particular interest for financial applications.
- *Interpretability and trust:* Many modern RL agents, especially those using deep neural networks, function as "black boxes." This lack of transparency is a major hurdle for adoption in a highly regulated industry where portfolio managers and risk officers need to understand and justify investment decisions.

The Research Frontiers

Addressing these challenges is the focus of ongoing research. Several exciting frontiers are emerging that are particularly relevant for finance:

- *Model-based RL and learned simulators:* To improve sample efficiency, researchers are developing agents that simultaneously learn a policy and a model of the environment. A learned world model can be used to generate simulated experiences, allowing the agent to "plan" and learn much faster than by relying solely on real market data.
- *Explainable AI (XAI) for RL/IRL:* A critical area of research is the development of methods to make the decisions of RL/IRL agents more transparent. Such techniques as sensitivity analysis and feature attribution can help practitioners understand which market signals are driving an agent's actions, fostering greater trust and facilitating model risk management.
- *Multi-agent RL (MARL):* Financial markets are inherently multi-agent systems. MARL moves beyond the single-agent paradigm to model the strategic interactions between multiple learning agents (e.g., competing high-frequency traders, interacting institutional investors, networks of broker/dealers in OTC markets). Similarly, multi-agent IRL aims to deconvolve the observed market dynamics into the behaviors and objectives of distinct classes of agents, which is a significant step beyond "single representative agent" models.
- *RL with large language models (LLMs):* The intersection of RL and LLMs is a rapidly developing area with significant implications for finance. LLMs can process and synthesize vast amounts of unstructured text data, such as news, filings, and social media, creating a richer, more nuanced state representation for an RL agent. Beyond simply enhancing the state, the methods used to align LLMs with human intent are directly applicable to financial decision making.
  - *Reinforcement learning from human feedback (RLHF):* This is the established technique used to fine-tune models such as ChatGPT. It involves a multistage process: First, a reward model is trained on human preference data (e.g., a human ranks several model-generated responses). Then, this reward model is used to fine-tune the LLM's policy using an RL algorithm (such as PPO). In a financial context, this approach could be adapted to align a trading or allocation agent with the complex, hard-to-specify intuition of an expert portfolio manager. A manager could provide qualitative feedback by ranking several trade proposals generated by the agent, allowing the system to learn the manager's implicit risk preferences and market views without the manager having to articulate an explicit utility function.
  - *Direct policy optimization:* Although powerful, RLHF is a complex and potentially unstable multistage process. Rafailov, Sharma, Mitchell, Ermon, Manning, and Finn (2023) introduced a more elegant and direct method called direct preference optimization (DPO), which bypasses the need for an explicit reward model. It uses a clever mathematical re-parameterization to show that the reward-modeling-plus-RL objective can be optimized directly on the preference data with a single, stable loss function. This approach significantly simplifies the training process. For finance, DPO offers a more robust and efficient way to fine-tune an RL agent's policy based on direct human preference data.
  - *Group preference optimization:* A limitation of both RLHF and DPO is their focus on a single preference provider. In many financial settings, decisions must satisfy multiple stakeholders with potentially conflicting objectives (e.g., a pension fund manager

balancing the needs of different beneficiary groups or a team of traders with diverse market views). Zeng, Zhang, Yang, and Chen (2024) formalized this problem as group preference optimization (GPO), a multi-agent extension of DPO. GPO learns a single policy that represents a social welfare optimum—a Condorcet winner—that is most preferred by the group as a whole. This approach provides a principled framework for building RL agents that can learn from and make decisions for a group of diverse financial experts or stakeholders.

Case Study: Trading and Option Hedging

In this section, we apply RL to building trading and option hedging strategies. We begin by exploring how RL can be used in trading, focusing on the ability of AI to discover optimal strategies despite market frictions, the definition of optimality in finance, and ways to encode risk preferences within the RL framework.

Next, we examine how various reward functions and state representations arise naturally in trading problems. We discuss how to formulate RL problems to reflect real financial objectives, such as maximizing expected utility, managing risk, or hedging derivatives.

We then introduce simple illustrative models—mean-reversion trading and option hedging—that demonstrate the mechanics of RL in finance and reveal key practical challenges. These foundational examples set the stage for more-advanced RL methods, including policy-gradient and deep RL approaches. As a practical complement to the discussion, we provide an open-source code example implementing the mean-reversion trading model, allowing readers to experiment with RL in a real trading context.

By the end of the case study, you will have a practical understanding of how to frame trading problems within the RL paradigm, design reward and state structures aligned with investment objectives, and interpret the results and limitations of RL-based trading strategies. This foundation prepares you for further study of deep RL, risk-sensitive methods, and real-world deployment in quantitative finance.

Defining Optimality and the Role of the Reward Function in Trading

We begin this section by discussing some worthwhile questions that motivated the first investigations into the application of RL to optimal trading strategies.

Question 1. *Can an artificial intelligence autonomously identify an optimal dynamic trading strategy, accounting for transaction costs, without prior knowledge of the strategy's structure?*

AlphaGo Zero (Silver, Schrittwieser, Simonyan, Antonoglou, Huang, Guez, Hubert, et al. 2017) is a historically important RL system that learned to play with "zero" human guidance, given only the rules of the game and the chance to play against a simulator. Question 1 asks, What are the various financial analogues of AlphaGo Zero—where an RL agent learns trading or investment strategies from first principles, with minimal human guidance?

Question 2. *In the context of Question 1, how should we define an optimal strategy? Is optimality inherently subjective, or can we rigorously quantify the strategy that a rational decision maker would employ?*

In finance, we define a strategy as optimal if it maximizes the expected utility of terminal wealth. This notion is grounded in the seminal work of von Neumann and Morgenstern (1944), who showed that if a decision maker (a) faces uncertain (probabilistic) outcomes and (b) has preferences satisfying four axioms of rational behavior, then the decision maker will behave as if maximizing the expected value of a utility function u, defined over possible outcomes. For further details, see Benveniste, Kolm, and Ritter (2024). When applied to trading or investment management, the relevant outcome is typically terminal wealth w_T, so the rational agent's objective becomes

$$E[u(w_T)]. \tag{1}$$

This principle underpins modern portfolio theory (Markowitz 1952; Merton 1969; Merton 1971) and provides the foundation for quantitative models of optimal investment and trading. We express terminal wealth as the sum of initial wealth w_0 and the cumulative increments in wealth over time:

$$w_T = w_0 + \sum_{t=1}^{T} \delta w_t, \tag{2}$$

where

$$\delta w_t := w_t - w_{t-1} \tag{3}$$

denotes the wealth increment at time t and T is the final time.

Although it might be tempting to maximize $E[w_T]$ directly, doing so ignores risk and can lead to paradoxes or undesirable outcomes. In contrast, maximizing expected utility $E[u(w_T)]$ incorporates risk preferences appropriately. Chamberlain (1983) and Benveniste et al. (2024) showed that the equivalence between maximizing expected utility and using mean–variance optimization is much broader than most practitioners realize. The classic mean–variance approach remains theoretically justified across a wide range of realistic return distributions—not just under the normality assumption. Specifically, this result applies whenever the distribution of terminal wealth is mean–variance equivalent (MVE), a broad class that includes all elliptical distributions, such as the normal and multivariate t-distributions, as well as a family of asymmetric distributions with well-defined first and second moments. In such cases, a rational agent's preferences over risky outcomes can be fully characterized by the mean and variance of terminal wealth, and the optimal strategy reduces to maximizing expected return penalized by risk, as in the classical Markowitz mean–variance framework (Markowitz 1952). We clarify those assumptions next.

Assumption 1 (Discreteness). *Trading occurs at discrete times* ($t = 1,\ldots,T$)*, and final wealth is given by Equation* 2.

Assumption 2 (Portfolios). *There exists a set of portfolios* ($h_0,\ldots,h_{T-1}$) *known at* $t = 0$ *such that*

$$\delta w_t = h_{t-1} \cdot r_t, \tag{4}$$

where h_t *is the dollar holdings vector at time* t *and* r_t *is the random vector of asset returns over* $[t-1,t]$.

Assumption 3 (Independence). *If* $t \neq s$*, then* r_t *and* r_s *are independent.*

Assumption 4 (Mean–Variance Equivalence). *For each t, the distribution of r_t is mean–variance equivalent.*

Assumption 5 (Utility). *The utility function is increasing and concave, and it has continuous derivatives up to second order.*

The assumption that the distribution is MVE is perhaps the most interesting one. Although the mean–variance optimization framework dates back to Markowitz's pioneering work in the early 1950s, the precise conditions under which mean–variance optimization is truly equivalent to expected utility maximization were only recently clarified by Benveniste et al. (2024). This distinction is central in our context, because a von Neumann–Morgenstern rational investor aims to maximize $E[u(w_T)]$, whereas the reward signal in Equation 11 takes a mean–variance form. Under the MVE condition, these objectives are, in fact, equivalent. Specifically, there exists some constant

$$\kappa > 0, \tag{5}$$

that depends on initial wealth w_0 and the investor's utility function, such that maximizing

$$E[u(w_T)] \tag{6}$$

is equivalent to maximizing

$$E[w_T] - \frac{1}{2}\kappa \mathbb{V}[w_T]. \tag{7}$$

In the following, we focus on

$$\text{maximize}\left\{\mathbb{E}[w_T] - \frac{\kappa}{2}\mathbb{V}[w_T]\right\}. \tag{8}$$

The first example of a reward signal appropriate for mean–variance utility in the context of RL was provided by Ritter (2017), which we now describe (see also Ritter 2018). Suppose we could invent some definition of reward (R_t) such that

$$E[w_T] - \frac{\kappa}{2}V[w_T] \approx \sum_{t=1}^{T} R_t. \tag{9}$$

Then, the optimization problem (Equation 8) looks like the kind of "cumulative reward over time" problem that is typical in RL. RL searches for policies that maximize

$$E[G_t] = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots], \tag{10}$$

which by Equation 9 would then maximize expected utility as long as $\gamma \approx 1$.

We consider the reward function proposed by Ritter (2017):

$$R_t := \delta w_t - \frac{\kappa}{2}\left(\delta w_t - \hat{\mu}\right)^2, \tag{11}$$

where $\hat{\mu}$ is an estimate of a parameter representing the mean wealth increment over one period, ($\mu := E[\delta w_t]$). Then, averaging over T periods yields

$$\frac{1}{T}\sum_{t=1}^{T} R_t = \underbrace{\frac{1}{T}\sum_{t=1}^{T}\delta w_t}_{\to \mathbb{E}[\delta w_t]} - \frac{\kappa}{2}\underbrace{\frac{1}{T}\sum_{t=1}^{T}\left(\delta w_t - \hat{\mu}\right)^2}_{\to \mathbb{V}[\delta w_t]}, \tag{12}$$

where for large T, the first term approaches the sample mean and the second approaches the sample variance of the wealth increments. Thus, with the reward function (Equation 11), maximizing cumulative reward implies maximizing the mean–variance form of expected utility.

State Variables for Trading Problems

The state variable (s_t) is a data structure that, simply put, should contain everything the agent needs to make an informed trading decision and nothing else. Variables that are natural candidates for inclusion in the state are

- the current position or holding in the asset,
- the values of any signals that are believed to be predictive,
- the current state of the market, including current price and any relevant microstructure/limit order book details, and
- if the portfolio includes contracts with expirations, such as futures or options, the time remaining until expiry.

In trading problems, the most natural choice for an action is the number of shares to trade, δn_t. This choice identifies the action space ($\mathcal{A} \subset Z$). When market microstructure effects are significant, the action space may need to be expanded. For example, the agent could decide which execution algorithm to use, choose between crossing the spread or submitting a passive order, or set the target participation rate. If one of the assets is an option, the agent may also have the ability to take additional actions, such as early exercise.

Trading Examples

In this section, we present two simple examples—mean-reversion trading and option hedging—that highlight key ideas and challenges in applying RL in trading.

Mean Reversion

We assume there exists a tradable asset with a strictly positive price process $p_t > 0$. This "asset" could be a portfolio of other assets, such as an exchange-traded fund or a hedged relative value trade. Further suppose that there exists an "equilibrium price" p_e such that $x_t = \log(p_t/p_e)$ has dynamics

$$dx_t = -\lambda x_t + \sigma\xi_t, \tag{13}$$

where $\xi_t \sim N(0,1)$ and ξ_t, ξ_s are independent when $t \neq s$. This means that p_t tends to revert to its long-run equilibrium level, p_e, with mean-reversion rate λ.

These assumptions describe a scenario that is close to an arbitrage: Positions established far from equilibrium have a very small probability of loss and highly asymmetric gain/loss profiles. Initially, we do not allow the agent to know anything about the dynamics. That is, the agent does not know λ, σ, or even that some dynamics of Equation 13 are valid. The agent also does not know there are trading costs. We impose a spread cost of one tick per trade. If the bid–offer

spread were equal to two ticks, then this proportional cost would correspond to the slippage incurred by an aggressive fill that crosses the spread to execute. Hence,

$$\text{SpreadCost}(\delta n) = \text{TickSize} \cdot |\delta n|. \tag{14}$$

Additionally, we assume there is a temporary price impact with a linear functional form: Each round lot traded is assumed to move the price one tick, hence leading to a dollar cost

$$\delta n_t \times \text{TickSize/LotSize} \tag{15}$$

per share traded, for a total dollar cost for all shares

$$\text{ImpactCost}(\delta n) = (\delta n)^2 \times \text{TickSize/LotSize}. \tag{16}$$

Together, the total trading cost is given by

$$\text{Cost}(\delta n) = \text{Multiplier} \times [\text{SpreadCost}(\delta n) + \text{ImpactCost}(\delta n)]. \tag{17}$$

This functional form matches the *t*-cost assumptions of Almgren and Chriss (1999). In fact, the Almgren–Chriss model of optimal execution can be learned by an RL agent, similar to the Ornstein–Uhlenbeck trading model discussed here.

The state variable at time *t*,

$$s_t = (p_t, n_{t-1}), \tag{18}$$

consists of the current asset price, p_t, and the agent's position in shares, n_{t-1}, at the start of the period.

As a proof of concept and in the spirit of exhausting the simplest method first, Ritter (2017) trained a tabular Q-learner with $n_{train} = 10^7$ training steps and then evaluated the system on 5,000 new samples of the stochastic process; see **Exhibit 1**.

These results look encouraging. We simulated a simple dynamic wherein we know there is an arbitrage, analogous to a game where it is actually possible to win. The machine then learns to play this game and develops a profitable strategy. How well did the agent really learn? To find the answer, we examine a cross-section of its learned action-value function $\hat{q}$, a diagnostic that reveals not only whether the agent profits but also how it chooses actions across different states. This deeper analysis helps us move beyond performance to evaluate the quality and reliability of the learned policy (see **Exhibit 2**).

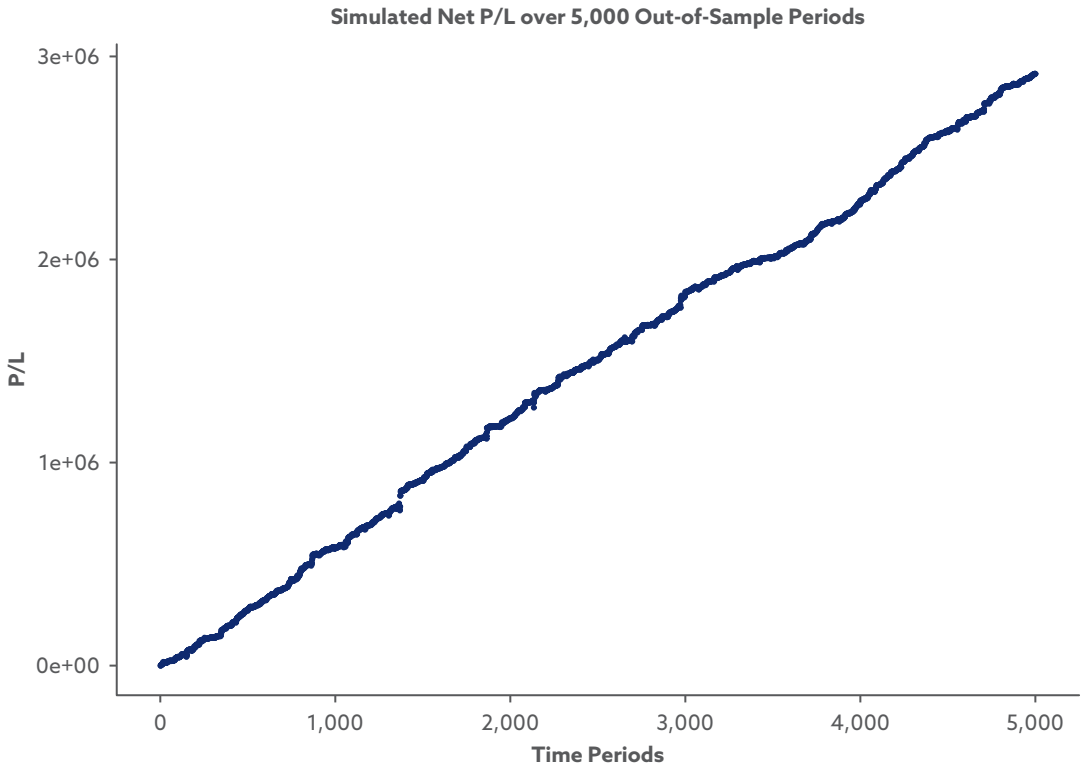
The main weakness of the tabular method is that it estimates each element

$$\hat{q}(s,a) \tag{19}$$

in isolation, without any smoothing across neighboring states or any inherent continuity. In this example, the optimal action choice exhibits a natural monotonicity, which we now describe intuitively.

If the current holding is $h = 0$ and for some price $p < p_e$ the optimal action is to buy 100 shares, then for any lower price $p' < p$, the optimal action should be to buy at least 100 shares. As Exhibit 2 illustrates, however, for large prices, the tabular value function often oscillates between different decisions, violating this monotonicity. This behavior reflects estimation

Exhibit 1. Tabular Q-Learner with 10^7 Training Steps, Evaluated on 5,000 New Samples



Notes: This exhibit uses simulated data. P/L stands for profit/loss.

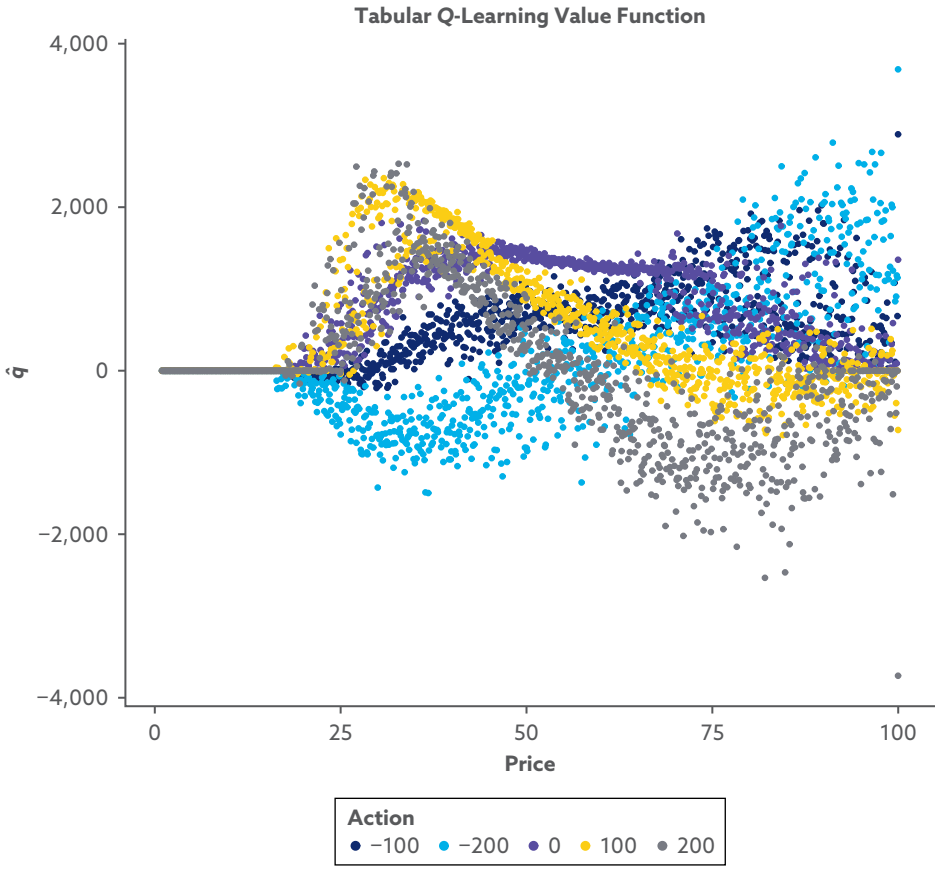
error and incomplete convergence, even after millions of iterations. The tabular value function also tends to collapse to a trivial form in the far-left tail because those states are rarely visited during training.

All these issues stem from the same fundamental limitation: the use of a finite, tabular state space. Methods that rely on discretizing the state space inevitably break down as dimensionality increases. For example, if the state vector included 10 variables, each taking 100 possible values, the number of parameters to estimate would reach into the billions. The solution to this curse of dimensionality is to move beyond tabular methods and adopt continuous state spaces with function approximation in RL. In the next section, we illustrate this approach with a concrete example.

Option Hedging with Transaction Costs

We now consider another problem of interest to traders: hedging an option position. For clarity, we focus on the simplest possible case—a European call option with strike K and expiry T on a non-dividend-paying stock. We set the strike and maturity as exogenous constants and, for

Exhibit 2. Value Function $p \to \hat{q}\,[(0, p), a]$, Where $\hat{q}$ Is Estimated by the Tabular Method



simplicity, assume a risk-free rate of zero. We train the agent to hedge this specific option with its given strike and maturity, rather than teaching it to hedge options with arbitrary parameters.

To hedge a European option, the state must at least contain the current price, S_t, of the underlying; the time remaining to expiry,

$$\tau := T - t > 0; \tag{20}$$

and our current position of n shares in the underlier. The state is thus naturally an element of

$$\mathcal{S} := \mathbb{R}_+^2 \times \mathbb{Z} = \{(S, \tau, n) \,|\, S > 0, \tau > 0, n \in \mathbb{Z}\}. \tag{21}$$

We emphasize that the state does not need to contain the option Greeks, because these quantities are nonlinear functions of the state variables already available to the agent. We expect agents to learn such nonlinear functions on their own as needed. This approach has the advantage of not requiring any model-based calculations. First, we consider a "frictionless" world (i.e., without trading costs) and ask whether it is possible for a machine to learn what we teach

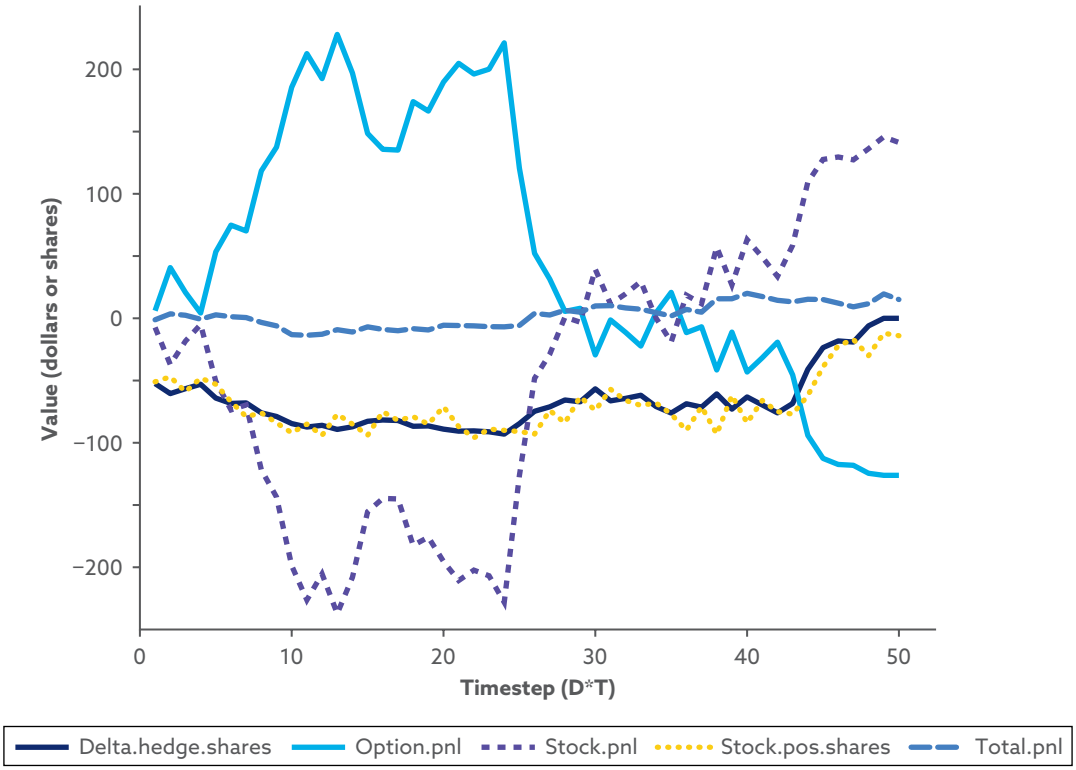
students in their first semester of business school: the dynamic replicating portfolio strategy. Unlike our students, the machine can learn only by observing and interacting with the environment, without any explicit instruction.

The RL agent is initially at a disadvantage. Recall that it does not know any of the following pertinent pieces of information:

- The strike price, K
- The fact that the stock price process is a geometric Brownian motion
- The volatility of the price process
- The Black–Scholes–Merton formula
- The payoff function, $(S - K)_+$, at maturity
- Any of the Greeks

It must infer the relevant information from the state variables, insofar as it affects the value function, by interacting with the environment. We present the results in **Exhibit 3**. Notably, the cumulative stock and option P&L largely offset each other, resulting in a total P&L with relatively

Exhibit 3. Out-of-Sample Simulation of a Trained Agent



Note: We depict cumulative stock, option, and total P&L; the RL agent's position in shares (Stock.pos.shares); and $-100 \cdot \Delta$ (Delta.hedge.shares).

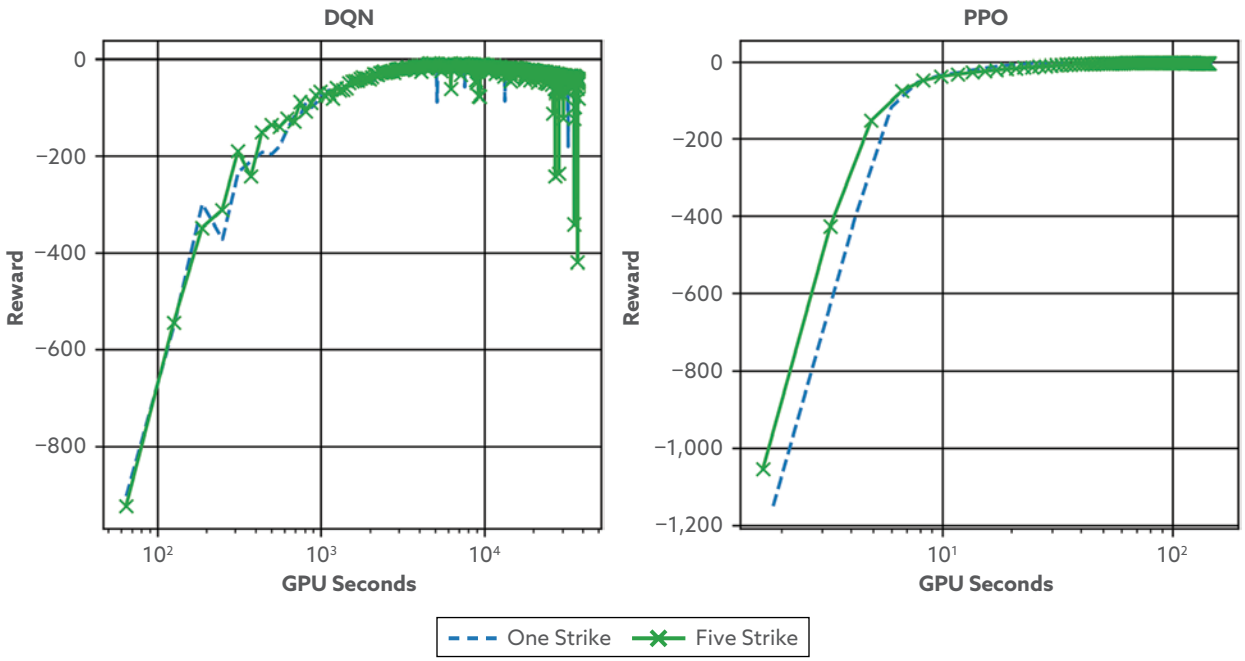
low variance. We also see that the RL agent's position closely tracks the delta position, despite having no direct access to it. For a more detailed discussion, see Kolm and Ritter (2019).

Policy Gradient Methods

When working in continuous state spaces, policy gradient methods frequently outperform value-based approaches. Du, Jin, Kolm, Ritter, Wang, and Zhang (2020) were the first to apply proximal policy optimization (PPO) to the problem of option hedging with transaction costs. The authors developed models that replicate options across a wide range of strikes while incorporating discrete trading, round lots, and nonlinear transaction costs. These models use deep RL techniques—including deep Q-learning and PPO—and are built to interface easily with any option pricing and simulation library, enabling users to train them on arbitrary option portfolios without further modification. Our empirical studies demonstrate that deep RL models can match or surpass traditional delta hedging, with PPO delivering superior results in terms of profit and loss, training speed, and data efficiency; see **Exhibit 4**.

Note that policy gradient methods, including PPO and asynchronous actor-critic (A2C), can be applied to the mean-reversion trading model discussed earlier and, in practice, yield significantly better results than the simple tabular Q-learning approach used by Ritter (2017).

Exhibit 4. Average Reward vs. GPU Seconds for the DQN and PPO Agents in the One and Five Strike Scenarios



Computational Implementations

The state of the art for computational AI is evolving at an extraordinary pace. Nevertheless, we believe that providing readers with a short, self-contained program that implements the mean-reversion example we showed will contribute to practical understanding and further progress in this area.

The code for this guide was developed with the goal of prioritizing simplicity, brevity, and reproducibility, using only open-source frameworks. Computational efficiency was intentionally not the aim, in the spirit of Knuth's well-known advice that premature optimization is the root of all (programming) evil.

For our implementation, we chose Stable Baselines 3 (SB3), a set of reliable, well-tested, and standardized RL algorithms built on PyTorch. SB3 offers a clean, modular codebase and a consistent API for training, evaluating, and deploying RL agents. It is compatible with gymnasium environments. These features make SB3 an ideal computational engine for our examples.

For clarity, our code sample consists of just two short files: reversion.py defines the custom environment for the mean-reversion trading problem described previously, and main.py is a main script that orchestrates the training and performance evaluation. The primary outputs of the main script are a set of image files that summarize the learning curve, the behavior of the trained agent, and the key performance metrics, such as P/L and the Sharpe ratio. The trained model is also saved as a file that can be reloaded for later use. Setting the Boolean variable discrete to true switches the model to a discrete action space; by default, the model operates with a continuous action space.

In summary, machines can identify arbitrage opportunities in data when such opportunities exist and can learn to optimize over long horizons in the presence of transaction costs. Despite these advances, the field of RL in finance remains in its early stages, and we are still far from a fully autonomous "Skynet" for trading. In practice, working in continuous state spaces is most effective because most optimal value functions in finance are continuous—and often monotonic—functions of the state, with properties grounded in economic intuition. Many optimal value functions are smooth or piecewise smooth, as seen in such classic problems as the linear-quadratic regulator. These same RL methods can be applied to derivative hedging in illiquid markets, where trading costs play a crucial role. RL agents can learn to price and hedge derivatives in environments where perfect replication is either impossible or prohibitively expensive.

Conclusion and Outlook

Reinforcement learning and inverse reinforcement learning mark a paradigm shift in quantitative finance, transitioning from static predictive models to dynamic, adaptive decision-making systems. RL excels in optimizing sequential decisions under uncertainty, making it ideal for such tasks as asset allocation, trade execution, and risk management. IRL, although less mature, enables the inference of preferences or objectives from observed behaviors, offering novel insights into individual trader strategies or market dynamics.

For practitioners aiming to adopt these technologies, a disciplined and strategic approach is essential. The path to real-world deployment is complex, but the benefits—improved performance and deeper market understanding—are significant. We provide the following key recommendations:

- *Target well-defined problems:* Choose applications with clearly defined states, actions, and rewards to ensure effective RL implementation. Identify problems aligned with your specific objectives, where the decision-making process can be modeled and optimized systematically.
- *Integrate risk management early:* Incorporate risk considerations directly into the reward function, using risk-averse or distributional RL frameworks. Doing so ensures alignment with the desired risk–return profile, avoiding the pitfalls of retrofitting risk controls.
- *Prioritize robust simulation:* Success hinges on high-quality data and realistic simulation environments. Rigorous backtesting, out-of-sample validation, and stress testing across diverse market scenarios are critical before deployment.
- *Leverage hybrid intelligence:* Combine RL and IRL with human expertise for optimal results. Use IRL to codify successful human strategies and RL to refine them, creating systems that augment human decision making while addressing ethical considerations.
- *Address practical constraints:* Design frameworks to meet computational, latency, and regulatory requirements. Interpretability is essential in regulated environments and should be a core design principle.

The future of quantitative finance lies in adaptive, real-time learning systems. RL and IRL provide the theoretical and practical tools to realize this vision. Despite challenges, their ability to enhance performance and uncover new market insights makes them critical for practitioners to master. As computational power grows and algorithms advance, RL and IRL are poised to become indispensable components of the quantitative finance toolkit, driving widespread adoption across the industry.

References

Almgren, Robert, and Neil Chriss. 1999. "Value under Liquidation." *Risk* (December): 61–63.

Benveniste, Jerome, Petter N. Kolm, and Gordon Ritter. 2024. "Untangling Universality and Dispelling Myths in Mean–Variance Optimization." *Journal of Portfolio Management* 50 (8): 90–116. doi:10.3905/jpm.2024.50.8.090.

Black, Fischer, and Myron Scholes. 1973. "The Pricing of Options and Corporate Liabilities." *Journal of Political Economy* 81 (3): 637–54. doi:10.1086/260062.

Brown, Daniel S., Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. 2019. "Extrapolating beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations." In *International Conference on Machine Learning*, 783–792. doi:10.48550/arXiv.1904.06387.

Buehler, Hans, Lukas Gonon, Josef Teichmann, and Ben Wood. 2019. "Deep Hedging." *Quantitative Finance* 19 (8): 1271–91. doi:10.1080/14697688.2019.1571683.

Chamberlain, Gary. 1983. "A Characterization of the Distributions That Imply Mean–Variance Utility Functions." *Journal of Economic Theory* 29 (1): 185–201. doi:10.1016/0022-0531(83)90129-1.

Dixon, Matthew Francis, and Igor Halperin. 2020. "G-Learner and GIRL: Goal Based Wealth Management with Reinforcement Learning." Working paper (25 February). doi:10.2139/ssrn.3543852.

Dixon, Matthew F., Igor Halperin, and Paul Bilokon. 2020. *Machine Learning in Finance: From Theory to Practice*. Cham, Switzerland: Springer. doi:10.1007/978-3-030-41068-1.

Du, Jiayi, Muyang Jin, Petter N. Kolm, Gordon Ritter, Yixuan Wang, and Bofei Zhang. 2020. "Deep Reinforcement Learning for Option Replication and Hedging." *Journal of Financial Data Science* 2 (4): 44–57. doi:10.3905/jfds.2020.1.045.

Ernst, Damien, Pierre Geurts, and Louis Wehenkel. 2005. "Tree-Based Batch Mode Reinforcement Learning." *Journal of Machine Learning Research* 6 (18): 503–56.

Finn, Chelsea, Paul Christiano, Pieter Abbeel, and Sergey Levine. 2016. "A Connection Between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models." Working paper (25 November). doi:10.48550/arXiv.1611.03852.

Halperin, Igor. 2020. "QLBS: Q-Learner in the Black-Scholes(-Merton) Worlds." *Journal of Derivatives* 28 (1): 99–122. doi:10.3905/jod.2020.1.108.

Halperin, Igor. 2024. "Distributional Offline Continuous-Time Reinforcement Learning with Neural Physics-Informed PDEs (SciPhy RL for DOCTR-L)." *Neural Computing & Applications* 36: 4643–59. doi:10.1007/s00521-023-09300-7.

Halperin, Igor, Jiayo Liu, and Xiao Zhang. 2022. "Asset Allocation with Inverse Reinforcement Learning." *Risk.net* (30 November). doi:10.2139/ssrn.4002715.

Ho, Jonathan, and Stefano Ermon. 2016. "Generative Adversarial Imitation Learning." In *Advances in Neural Information Processing Systems 29*. doi:10.48550/arXiv.1606.03476.

Kolm, Petter N., and Gordon Ritter. 2019. "Dynamic Replication and Hedging: A Reinforcement Learning Approach." *Journal of Financial Data Science* 1 (1): 159–71. doi:10.3905/jfds.2019.1.1.159.

Levine, Sergey, Zoran Popović, and Vladlen Koltun. 2011. "Nonlinear Inverse Reinforcement Learning with Gaussian Processes." In *NIPS'11: Proceedings of the 25th International Conference on Neural Information Processing Systems*, 19–27. https://dl.acm.org/doi/abs/10.5555/2986459.2986462.

Li, Yang, Zhi Chen, and Steve Yang. 2025. "FlowHFT: Flow Policy Induced Optimal High-Frequency Trading under Diverse Market Conditions." Working paper (22 May). doi:10.48550/arXiv.2505.05784.

Markowitz, Harry. 1952. "Portfolio Selection." *Journal of Finance* 7 (1): 77–91.

Merton, Robert C. 1969. "Lifetime Portfolio Selection under Uncertainty: The Continuous-Time Case." *Review of Economics and Statistics* 51 (3): 247–57. doi:10.2307/1926560.

Merton, Robert C. 1971. "Optimum Consumption and Portfolio Rules in a Continuous-Time Model." *Journal of Economic Theory* 3 (4): 373–413. doi:10.1016/0022-0531(71)90038-X.

Merton, Robert C. 1974. "On the Pricing of Corporate Debt: The Risk Structure of Interest Rates." *Journal of Finance* 29 (2): 449–70. doi:10.2307/2978814.

Ng, Andrew Y., Daishi Harada, and Stuart J. Russell. 1999. "Policy Invariance under Reward Transformations: Theory and Application to Reward Shaping." In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, 278–87.

Noguer i Alonso, Miquel, and Sonam Srivastava. 2020. "Deep Reinforcement Learning for Asset Allocation in US Equities." Working paper (9 October). doi:10.2139/ssrn.3711487.

Pricope, Tidor-Vlad. 2021. "Deep Reinforcement Learning in Quantitative Algorithmic Trading: A Review." Working paper (31 May). doi:10.48550/arXiv.2106.00123.

Rafailov, Rafael, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. "Direct Preference Optimization: Your Language Model Is Secretly a Reward Model." Working paper (13 December). doi:10.48550/arXiv.2305.18290.

Ritter, Gordon. 2017. "Machine Learning for Trading." Working paper (8 August). doi:10.2139/ssrn.3015609.

Ritter, Gordon. 2018. "Reinforcement Learning in Finance." In *Big Data and Machine Learning in Quantitative Investment*, edited by Tony Guida, 225–250. Hoboken, NJ: John Wiley & Sons.

Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, et al. 2017. "Mastering the Game of Go without Human Knowledge." *Nature* 550 (7676): 354–59. doi:10.1038/nature24270.

Spooner, Thomas, John Fearnley, Rahul Savani, and Andreas Koukorinis. 2018. "Market Making via Reinforcement Learning." Working paper (11 April). doi:10.48550/arXiv.1804.04216.

Sutton, Richard S., and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press.

von Neumann, John, and Oskar Morgenstern. 1944. *Theory of Games and Economic Behavior*. Princeton, NJ: Princeton University Press.

Yang, Steve Y., Qifeng Qiao, Peter A. Beling, William T. Scherer, and Andrei A. Kirilenko. 2015. "Gaussian Process-Based Algorithmic Trading Strategy Identification." *Quantitative Finance* 15 (10): 1683–703. doi:10.1080/14697688.2015.1011684.

Yang, Steve Y., Yangyang Yu, and Saud Almahdi. 2018. "An Investor Sentiment Reward-Based Trading System Using Gaussian Inverse Reinforcement Learning Algorithm." *Expert Systems with Applications* 114 (30 December): 388–401. doi:10.1016/j.eswa.2018.07.056.

Zeng, J.-C., Z. Zhang, Y. Yang, and J. Chen. 2024. "Group Preference Optimization: A Multi-Agent Formalization of RLHF for Scenarios with Multiple Stakeholders."

Ziebart, Brian D., Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. 2008. "Maximum Entropy Inverse Reinforcement Learning." In *Twenty-Third AAAI Conference on Artificial Intelligence*, 1433–38.

NATURAL LANGUAGE PROCESSING

Francesco A. Fabozzi, PhD
Research Director, Yale International Center for Finance

Introduction

Natural language processing (NLP) has become a foundational technology in financial analysis and investment decision making. As modern financial markets are increasingly shaped by the flow of unstructured textual information—from earnings calls and regulatory filings to news articles and social media—investors and analysts face growing pressure to systematically process, interpret, and act on these data.

At the same time, large language models (LLMs) have ushered in a paradigm shift in how textual data can be analyzed. These models—trained on vast corpora of text—can perform a wide range of tasks with minimal supervision, including question answering, summarization, classification, and entity extraction. Their ability to generalize across tasks and domains has made them particularly attractive for financial applications, where bespoke data and limited training labels often hinder the use of conventional supervised learning approaches.

This chapter explores the evolving role of NLP in finance, with a particular emphasis on the transformative impact of LLMs. I argue that the flexibility, scalability, and adaptability of LLMs have opened up new frontiers for analyzing financial text, enabling both discretionary and systematic investors to extract insights that were previously inaccessible. The integration of these models into financial workflows, however, also introduces new challenges related to trust, evaluation, infrastructure, and regulation. The sections that follow trace the technical evolution of NLP methods, examine the capabilities and limitations of generative models, and highlight the most relevant applications, risks, and future directions for financial professionals.

Evolution of NLP Techniques

Although today's LLMs offer flexible and high-performing solutions for a wide range of text-based tasks, their development has been built upon decades of progress in the field of NLP. Understanding the historical evolution of NLP methods is valuable not only for appreciating how the field arrived at its current capabilities but also for identifying use cases where simpler, more computationally efficient techniques may still be appropriate. In many financial applications, classical NLP tools remain relevant and cost-effective when high predictive accuracy or linguistic nuance is not essential.

Computers cannot naturally interpret language in the same way they handle structured, numerical data. As a result, early NLP efforts focused on converting textual information into numerical representations suitable for modeling. Over time, methods have evolved from treating language as unordered collections of words to more sophisticated approaches capable of capturing syntax, semantics, and even pragmatic meaning.

Rule-Based and Dictionary Approaches

The earliest NLP techniques, dating back to the 1950s, were based on rule-based systems and hand-crafted dictionaries. These approaches involved predefined word lists associated with particular categories, such as sentiment, emotion, or subject matter. For example, a sentiment dictionary might classify "excellent," "profit," or "growth" as positive terms, while "loss," "decline," or "risk" would be labeled negative. An analyst could then quantify the sentiment of a document by counting the number of positive and negative words it contained.

To illustrate, consider a simple example. Suppose a sentence reads, "The company reported strong revenue growth but noted increased supply chain risks." Using a basic sentiment dictionary, this sentence might register both positive (e.g., "strong," "growth") and negative (e.g., "risks") words, with the final classification depending on their relative counts. These methods were intuitive and easy to implement, but they came with substantial limitations:

- *Lack of context:* Dictionary methods ignore the surrounding context of words. For instance, they typically fail to account for negations ("not profitable") or modifiers ("barely profitable"), leading to misclassification.
- *Polysemy and ambiguity:* Words with multiple meanings, such as "interest" (loan interest versus personal interest), are treated uniformly, often introducing noise into analysis.
- *Equal weighting:* All words in the dictionary are treated as equally informative, failing to capture intensity differences—for example, "good" and "great" may both be classified as positive, despite differing in strength.
- *Subjectivity in word selection:* Manually assigning words to categories can reflect human bias and domain insensitivity.

These limitations became particularly apparent in financial applications. General-purpose sentiment dictionaries developed for consumer product reviews or news articles often performed poorly when applied to financial documents. For instance, the word "liability" may be negative in everyday language, but in accounting, it is a neutral technical term. Recognizing this issue, Loughran and McDonald (2011) analyzed common financial texts and found that approximately two-thirds of words classified as negative by standard dictionaries were not actually negative in a financial context. To address this, they introduced the "Loughran–McDonald Master Dictionary w/Sentiment Word Lists," specifically tailored for financial analysis. This development marked a turning point in domain-specific NLP and highlighted the importance of context-aware tools.

Statistical and Count-Based Methods

To address the limitations of rule-based and dictionary approaches, the next phase in NLP involved representing textual data in structured, statistical formats that could be analyzed using traditional machine learning models. This approach centers on the *document–term matrix*, where each row represents a document (such as a news article or earnings call) and each column corresponds to a unique word, or *term*, in the corpus. The entries in this matrix indicate the frequency with which each word appears in each document.

This representation enables words to be assigned weights by a statistical model, rather than relying on predefined dictionary labels. In effect, the model learns the relationship between

words and an outcome variable—such as stock returns or sentiment labels—based on observed co-occurrence patterns, allowing for a more data-driven understanding of language.

To illustrate, consider two documents:

- Document A: "The company reported strong earnings."
- Document B: "The company faced weak earnings."

A document–term matrix would represent these as follows (simplified for clarity):

	Company	Reported	Strong	Earnings	Faced	Weak
Document A	1	1	1	1	0	0
Document B	1	0	0	1	1	1

These *count vectors* can then be used as inputs to standard classifiers, such as logistic regression or naive Bayes, enabling empirical estimation of which words are associated with positive or negative outcomes.

This approach introduces new challenges, however, including dimensionality and sparsity: As the vocabulary size increases—particularly in financial text where domain-specific terminology is abundant—the document–term matrix becomes high-dimensional and sparse. That is, most entries are zero because any given document contains only a small subset of the total vocabulary. This sparsity can reduce model performance and increase computational burden.

To mitigate this drawback, practitioners often apply *text preprocessing* techniques, including the following:

- *Stopword removal:* Common, noninformative words, such as "the," "and," and "of," are removed.
- *Stemming and lemmatization:* Words are reduced to their base or root forms. For example, "running," "ran," and "runs" may all be reduced to "run," improving consistency and reducing dimensionality.
- *TF–IDF weighting:* In raw count-based matrices, common words dominate. To address this problem, *term frequency–inverse document frequency* (TF–IDF) weighting is used. TF–IDF downweights words that appear frequently across all documents (such as "company") and upweights words that are more unique to a specific document, helping highlight discriminative terms.

Despite these refinements, count vector–based methods still have significant limitations:

- *No context or ordering:* Word order is ignored, making it impossible to distinguish between "not profitable" and "profitable." This feature limits the model's ability to capture negation and other contextual modifiers.
- *No semantic relationships:* Each word is treated as an independent token, with no recognition of synonymy or antonymy. For instance, "good" and "great" are no more similar than "good" and "terrible" in this representation.

These drawbacks motivated the development of more-advanced models that can capture contextual and semantic meaning in language, which are discussed in the following sections.

Neural Models and Word Embeddings

The techniques discussed thus far focus on representing text in structured formats that can be used as inputs to traditional machine learning models. These representations—whether based on word counts or term frequency weighting—treat words as independent and do not attempt to capture the underlying relationships among them. Although useful for many tasks, these models fall short in their ability to *understand* language in a deeper, semantic sense.

As machine learning models and computational power advanced, NLP shifted from merely *representing* text to *learning from* text. Rather than manually crafting features or counting word occurrences, models began using neural networks to learn relationships between words directly from large corpora of text. This process is typically done through semisupervised learning: The model is trained on a simple prediction task, such as predicting a missing word based on its surrounding context. In doing so, the model develops an internal representation of word meaning—referred to as *word embeddings*.

A word embedding is a high-dimensional vector that encodes semantic information about a word. Words that appear in similar contexts tend to have similar embeddings. One of the most well-known approaches for generating word embeddings is *Word2Vec*, introduced by Mikolov, Sutskever, Chen, Corrado, and Dean (2013). Word2Vec is trained on such tasks as the following:

- *Continuous bag of words:* Predict a word based on its surrounding context (e.g., given "The company [...] strong earnings," predict "reported").
- *Skip-gram:* Predict surrounding words given a central word.

Through training, the model learns to associate each word in the vocabulary with a dense vector of real numbers. These vectors capture meaningful linguistic relationships. For example, consider the famous analogy from Mikolov et al. (2013):

$$\text{vector}(\text{"king"}) - \text{vector}(\text{"man"}) + \text{vector}(\text{"woman"}) \approx \text{vector}(\text{"queen"}).$$

It demonstrates how arithmetic operations on word embeddings can reveal latent semantic structure. What this says is that if we take the word embedding from "king," subtract that vector with the vector of the word embedding of "man," and add the vector for "woman," it will be more similar to the word embedding for "queen," illustrating the semantic meaning embedded in these word vectors.

The output of a model such as Word2Vec is an *embedding matrix*, where each row corresponds to a word in the vocabulary and each column represents a learned feature of the word. These embeddings can then be used as inputs to downstream models (i.e., models specializing in a specific task) for classification, clustering, or regression tasks—offering a more compact and semantically rich representation relative to the sparse count vectors discussed in the previous section.

These methods still have limitations, however, such as *static embeddings*: Once trained, each word has a single fixed embedding, regardless of context. This situation creates

problems for words with multiple meanings. For instance, consider the word "position" in two financial contexts:

- "The fund increased its position in Apple."
- "The trader accepted a new position at the firm."

In both cases, the same word vector would be used, even though "position" refers to an investment in one case and a job title in the other.

To address this issue, researchers introduced *recurrent neural networks* (RNNs) that process sequences of words in order, allowing models to generate *contextual embeddings*—word representations that change depending on surrounding words. Two widely used architectures in this family are the *long short-term memory* (LSTM) network and the *gated recurrent unit*. These models read a sentence one word at a time and retain a memory of the preceding words, enabling them to generate richer, context-aware representations.

Although sequential models such as LSTM networks improved performance on many NLP tasks, they also introduced new challenges:

- *Computational inefficiency:* Processing words one at a time limits parallelization and increases inference time (i.e., how long it takes a model to output predictions or embeddings).
- *Unidirectional context:* Standard LSTM networks typically read text in one direction—usually left to right—so they can condition only on previous words and not on future context.

These limitations paved the way for the next major breakthrough in NLP: the transformer architecture, which enables more efficient and flexible modeling of text with full context awareness. This development is explored in the following section.

The Transformer Revolution

The introduction of the *transformer architecture*, proposed by Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin (2017), in the landmark paper "Attention Is All You Need," marked a turning point in NLP development. The transformer not only outperformed prior models across a wide range of NLP benchmarks but also became the foundational architecture for all modern LLMs, including BERT, GPT, and their successors.

Although the transformer is a neural network like those discussed in the previous section, it departs significantly from earlier sequential models such as RNNs and LSTM networks. To understand its impact and how it differs from past architectures, this section presents a high-level conceptual overview of the transformer architecture and its two key innovations: *self-attention* and *encoder–decoder separation*.

Self-Attention: Capturing Relationships Across Text

At the core of the transformer is the *self-attention mechanism*, which allows the model to consider the relationship between all words (or *tokens*) in a given input sequence, regardless of their position. In contrast to LSTM networks, which process sequences word by word and typically look only at previous words (i.e., left to right), self-attention enables the model to compute

relevance scores across the entire sequence in parallel. This approach means that each word can attend to every other word, with the model weighting them based on learned relevance.

For example, in the sentence, "The investor sold the stock because it was overvalued," the word "it" could plausibly refer to either "stock" or "investor." A model equipped with self-attention can learn to focus on the correct referent—"stock"—by assigning higher attention weight to it. This capability to model long-range dependencies and ambiguity is one of the reasons why transformer-based models have proven so effective in capturing the complexity of human language.

The Encoder–Decoder Architecture

The original transformer architecture consists of two major components:

- *Encoder:* processes the input text and converts it into a contextualized vector representation
- *Decoder:* uses this representation to generate an output sequence, such as translated text, a summary, or a response

This encoder–decoder configuration was designed for such tasks as *machine translation*, *summarization*, and *question answering*—where the model must first understand the full input before producing a meaningful output. The encoder portion of the model captures the semantic content of the source text, while the decoder portion generates new text conditioned on that information captured by the encoder.

In practice, however, encoder and decoder components are often used *independently*, depending on the task:

- *Encoder-only models* (e.g., BERT)[1] are designed to create high-quality, contextualized embeddings of the input text. These embeddings can then be used as features for downstream tasks, such as sentiment analysis, classification, or information retrieval. Encoder-only models are especially valuable in settings where the goal is to interpret or score a piece of text rather than generate new text.
- *Decoder-only models* (e.g., GPT)[2] are optimized for *language generation*. These models predict the next token in a sequence given all previous tokens—a formulation known as *causal* or *autoregressive language modeling*. This approach enables the model to generate coherent text, answer questions, and engage in dialogue. Decoder-only models form the backbone of modern chat-based systems and instruction-following agents.

The transformer architecture introduced two breakthroughs: the ability to process all tokens in parallel through self-attention and the modular encoder–decoder framework that enables both understanding and generation of new text. These innovations not only improved performance

[1]BERT refers to the bidirectional encoder representations from transformers model introduced by Devlin, Chang, Lee, and Toutanova (2019).

[2]GPT refers to the generative pretrained transformer model introduced by OpenAI through a series of breakthrough papers (Radford, Narasimhan, Salimans, and Sutskever 2018; Radford, Wu, Child, Luan, Amodei, and Sutskever 2019; Brown, Mann, Ryder, Subbiah, Kaplan, Dhariwal, Neelakantan, et al. 2020). GPT also refers to the class of decoder-only models popularized through modern LLMs.

across a wide range of NLP tasks but also made it feasible to scale models to unprecedented size, giving rise to today's LLMs.

The next section explores how these architectural advances, coupled with massive pretraining, have enabled a new paradigm of generative modeling that can solve a broad range of tasks with little or no task-specific data.

Pretraining, Fine-Tuning, and the Modular Nature of LLMs

To understand how LLMs work—and why they have become so powerful—it is helpful to break down their training into distinct phases: *pretraining* and *fine-tuning*. These phases reflect a modular and scalable approach to training LLMs that separates the acquisition of general linguistic and world knowledge from the adaptation of the model to specific tasks. Although the underlying model architecture (encoder, decoder, or both) plays an important role in shaping the model's capabilities, it is ultimately the training process that determines what the model can do in practice.

Pretraining: Building General Linguistic and World Knowledge

Pretraining is the foundational phase of LLM development. In this stage, the model is exposed to a massive corpus of raw textbooks, articles, websites, and other publicly available documents—and trained to predict missing or future words. This process does not require labeled data, making it highly scalable. The goal is to instill the model with a statistical understanding of language and a broad (though implicit) knowledge of the world.

The specific training objective depends on the architecture:

- *Masked language modeling (MLM):* Used primarily for training *encoder-only* models (e.g., BERT), MLM involves hiding or "masking" certain words in a sentence and training the model to predict the masked word(s) based on surrounding context. For example, given the input "The company reported a [MASK] gain in profits," the model learns to infer that "strong" might be an appropriate word. Because encoder models attend to the entire input at once, they generate rich contextual embeddings useful for classification and retrieval tasks.
- *Causal language modeling (CLM):* Used to train *decoder-only* models (e.g., GPT), CLM trains the model to predict the next word in a sequence given all previous words. For instance, given "The company reported a strong gain in," the model learns to predict "profits." This autoregressive objective aligns naturally with generative tasks and enables the model to produce coherent, fluent text. Radford, Narasimhan, Salimans, and Sutskever (2018) demonstrated that performance improves substantially with model scale, leading to the development of highly parameterized models capable of general-purpose text generation.

This pretraining phase is computationally intensive and typically carried out by large technology firms with access to significant compute infrastructure and massive datasets. The resulting pretrained models can be distributed and reused, however, providing a strong foundation for downstream customization.

Fine-Tuning: Adapting Models to Tasks and Domains

Although pretraining equips LLMs with general capabilities, it does not tailor them to specific tasks. Pretrained models may be able to generate fluent text or encode sentences into embeddings, but they often lack the domain-specific behavior needed for practical applications in such areas as finance. This is where *fine-tuning* comes in.

Fine-tuning involves continuing the training of a pretrained model on a smaller, domain-specific or task-specific dataset. This process allows users to adapt general-purpose models for their own use cases without starting from scratch.

For *encoder-only models*, fine-tuning typically involves supervised learning. For example, to build a sentiment classifier, a practitioner would start with a pretrained BERT model and fine-tune it on labeled data consisting of text samples paired with sentiment labels. The resulting model becomes highly specialized for sentiment classification. The resulting model will be able to produce meaningful outputs only for that specific task, however, which is why fine-tuned encoder-only models are also referred to as "narrow" or "task-specific" models.

For *decoder-only models*, fine-tuning can be more general. After pretraining on internet-scale data, these models are often fine-tuned on curated datasets of human interactions—such as question-and-answer pairs or chat logs—to teach the model how to be helpful, safe, and responsive in a conversational setting. Without this additional tuning, a model might respond to a prompt such as "What is your name?" with a story or unrelated prose. With fine-tuning, it learns to respond appropriately to user queries.

One of the reasons decoder-only models have become so widely used is their *flexibility*. Rather than needing a new model for each task (e.g., sentiment classification, summarization, ESG scoring), a single decoder-only model can be fine-tuned to perform a wide range of tasks simply by changing the *prompt*—the input instruction or context provided to the model. For instance:

- "Classify the sentiment of this text: 'Markets rallied after the rate cut.'"
- "Summarize the following earnings report."
- "List three ESG risks mentioned in this document."

This ability makes decoder-only models attractive in such domains as finance, where labeled data are often scarce but a wide range of tasks need to be performed on unstructured text. By shifting the complexity to the prompt design, practitioners can avoid the costly process of training narrow models for every new application.

Prompting vs. Fine-Tuning

Although *prompting* allows users to extract useful behavior from a pretrained or instruction-tuned model without any additional training, it has its limits. In cases where the model fails to follow instructions, struggles with domain-specific jargon, or underperforms on niche tasks, *fine-tuning* can still play an important role.

Fine-tuning can be used for the following:

- *Task-specific adaptation:* training the model for a narrow task, such as legal clause extraction or credit rating prediction

- *Domain adaptation:* exposing the model to domain-specific data (e.g., financial filings, regulatory texts) to improve its familiarity with specialized vocabulary and formats
- *Knowledge injection:* teaching the model facts or behaviors that are underrepresented in the pretraining corpus

In practice, many financial workflows now combine these approaches, using off-the-shelf LLMs where prompt engineering suffices and fine-tuning smaller versions of the model when reliability and domain expertise are critical.

Computational Efficiency and Fine-tuning

Although LLMs with billions of parameters tend to exhibit superior performance across a range of tasks, this ability comes at a significant computational cost. Both training and deploying such models require high-end hardware—typically graphics processing units (GPUs) or tensor processing units (TPUs)—and scale roughly with model size. As a result, the use of very large models can be prohibitive in environments with limited compute resources or strict latency constraints, such as real-time financial systems.

One solution is to *fine-tune smaller models to replicate the behavior of larger ones*, commonly done through a process known as *knowledge distillation*. In this setup, a high-performing, large model (the *teacher*) generates high-quality outputs for a given task and a smaller model (the *student*) is trained to imitate these outputs. For instance, if a large model excels at summarizing long financial documents, its summaries can serve as labeled training data to fine-tune a smaller model to perform the same task. This process allows practitioners to deploy a compact model that behaves similarly to a much larger one but with far lower computational requirements.

In addition to distillation, another strategy for improving efficiency is *quantization*. Quantization reduces the numerical precision of a model's parameters—from 32-bit floating point numbers (FP32) to lower-precision formats, such as 8-bit integers (INT8) or 4-bit floats. Although this approach can slightly reduce model performance, it dramatically reduces the memory footprint and speeds up inference, particularly on GPU hardware. Because inference cost is one of the most significant factors in large-scale deployment, quantization plays a key role in bringing LLM capabilities to environments with limited resources.

Taken together, these techniques—fine-tuning, distillation, and quantization—offer powerful levers for balancing performance and efficiency. They enable organizations to deploy LLMs that are adapted to specific tasks and constraints, without bearing the full costs of operating state-of-the-art, billion-parameter models. In the context of finance, where tasks are often repeated at scale and latency can be critical, these efficiency gains are not just convenient; they are essential.

Fine-Tuning Techniques

The specific strategy used to fine-tune a language model depends largely on its architecture—whether the model is *encoder only* or *decoder only*. These two types of models serve different purposes and therefore require distinct approaches when adapting them to downstream tasks.

Fine-Tuning Encoder-Only Models

Encoder-only models, such as *BERT*, are designed to generate rich, contextual embeddings of input text sequences. These embeddings are high-dimensional vectors that capture the semantic meaning of the text, incorporating the full context of surrounding words through self-attention.

Fine-tuning these models generally involves one of two approaches:

- *Feature extraction:* In this setup, the pretrained model is used to generate embeddings for each input document, and these embeddings are then used as input features to a separate, downstream machine learning model (e.g., logistic regression, random forest). The weights of the language model itself remain fixed.
- *End-to-end fine-tuning:* A classification or regression head (i.e., an additional fully connected feedforward layer) is added on top of the encoder model, and the entire architecture—including the weights of the pretrained model—is trained jointly on labeled data. For example, if a BERT model outputs a 768-dimensional embedding vector, a single-layer neural network can be added to map this vector to a three-class output for sentiment classification (e.g., positive, neutral, negative). This process fine-tunes the entire network for the target task.

End-to-end fine-tuning often leads to better performance but is more computationally demanding. Variants of this approach include *partial fine-tuning*, where only the top few layers of the model are updated while the rest are frozen (i.e., unchanged), and *layer-wise freezing*, where layers are gradually unfrozen during training. These techniques reduce computational cost and help prevent overfitting, especially when labeled data are scarce.

Fine-Tuning Decoder-Only Models

Decoder-only models, such as *GPT*, are designed for text generation and contain significantly more parameters than encoder-only models. Fine-tuning these models in a traditional end-to-end fashion is often prohibitively expensive in terms of memory and compute resources, especially when dealing with billions of parameters. To address this problem, researchers have developed more efficient fine-tuning methods, the most prominent of which is *low-rank adaptation* (LoRA).

LoRA was introduced by Hu, Shen, Wallis, Allen-Zhu, Li, Wang, and Chen (2021) as a *parameter-efficient fine-tuning* method for large models. Instead of updating all weights of the pretrained model, LoRA inserts small, trainable weight matrices into the existing architecture while freezing the original weights. These low-rank matrices capture the necessary task-specific adjustments without needing to train the entire model. As a result, LoRA significantly reduces the number of trainable parameters, lowers GPU memory usage, and allows for faster training.

Importantly, LoRA also helps mitigate *catastrophic forgetting*—a phenomenon where a model forgets previously learned knowledge when fine-tuned on a narrow task—because the original pretrained weights remain intact. This feature offers a form of regularization when training highly parameterized models and makes LoRA particularly attractive for adapting general-purpose LLMs to specialized domains, such as finance, without compromising their broader language capabilities.

Deployment Strategies and Infrastructure Considerations for LLMs in Finance

The Challenge of Real-Time Information in Finance

One of the most significant challenges in applying LLMs in financial contexts is the pace at which information changes. News articles, social media posts, SEC filings, and earnings call transcripts can alter the investment landscape in real time. Traditional LLMs, however, are static once trained; they can make use only of the information available at the time of their pretraining or fine-tuning. In high-stakes, time-sensitive domains, such as finance, this fact poses a fundamental limitation: Retraining a large model every time new information becomes available is both impractical and cost prohibitive.

To overcome this constraint, many LLM-based systems now use a design pattern known as *retrieval-augmented generation* (RAG). RAG is not a model itself but an architectural framework that allows LLMs to dynamically incorporate external information at inference time. This ability enables a form of real-time updating without retraining the model.

RAG systems typically consist of two main components:

- An *encoder-based retriever* processes the user's query and searches an external document repository (such as a database of recent filings or news articles) for relevant information. This component is typically built using an encoder-only model, which generates embeddings for both the query and the documents and then identifies the most semantically similar content through measuring *cosine similarity* of the embedding of the user's query and embeddings of the document repository.
- A *decoder-only language model* receives the retrieved documents as context and generates a coherent answer to the user's query. This model does not need to have seen the documents during training—it uses the retrieved information as context provided via the prompt.

This modular approach solves a key problem in financial NLP: incorporating *new and proprietary data* on the fly. Rather than training the model to "know everything," encoders are used to structure and fetch the relevant information, and decoders are used to synthesize and reason over that information. This process is particularly powerful in such workflows as

- summarizing recent earnings call transcripts,
- responding to ESG-related inquiries with up-to-date regulatory documents, and
- generating investment commentary based on newly filed 10-Ks or 8-Ks.

By separating retrieval from generation, RAG frameworks enable dynamic, domain-aware responses without compromising latency or requiring model retraining.

Agentic Frameworks and Autonomous Systems

Although retrieval-augmented systems allow LLMs to access up-to-date information, many real-world financial workflows require more than just a single question-and-answer interaction. Such tasks as portfolio monitoring, document triage, regulatory surveillance,

and risk assessment often require *multistep reasoning*, *conditional execution*, and *coordination across multiple tools or models*. This is where agentic frameworks come into play.

Agentic Frameworks

An *agentic framework* refers to an architectural approach in which LLMs are embedded within a broader decision-making system that can sequence multiple steps to achieve a complex objective. Rather than providing a single response to a single prompt, the LLM acts as a "thinking component" that can

- break down a goal into sub-tasks,
- decide what action to take at each step,
- retrieve or generate intermediate information,
- use external tools (e.g., calculators, databases, search APIs), and
- determine when the task is complete.

In financial applications, this might involve chaining together such tasks as

- reading and extracting risk factors from a 10-K filing,
- identifying which factors are new or material relative to prior filings,
- summarizing their potential impact on valuation, and
- flagging the result for human review if certain thresholds are exceeded.

Rather than relying on a single LLM to do all this, the system orchestrates multiple steps, often using different models or tools tailored to the specific subtasks. This modular and compositional approach enables more interpretable, reliable, and extensible pipelines—crucial attributes in high-stakes domains such as finance.

Autonomous Agents

Agentic systems can be taken a step further with *autonomous agents*: LLM-based systems that are goal directed and capable of operating with *minimal human oversight*. These agents maintain a persistent objective (e.g., "monitor the top 500 firms for changes in litigation risk"), autonomously initiate actions to gather information, and adapt their behavior according to the results.

What distinguishes autonomous agents from simpler pipelines is their ability to

- formulate plans based on their objective,
- monitor their own progress and iterate,
- communicate with multiple systems or data sources, and
- run continuously or on demand.

Examples in finance might include

- an *autonomous compliance monitor* that scans regulatory filings, compares them with internal policies, and flags discrepancies;

- a *market intelligence agent* that compiles real-time information across sources to maintain an up-to-date dashboard for analysts; or
- a *due diligence assistant* that autonomously reads through multiple data sources and summarizes potential red flags during deal evaluation.

These systems remain experimental in many financial institutions, but early results suggest that autonomy can reduce human workload on repetitive tasks while increasing responsiveness to new developments. They also introduce new risks, however, particularly around reliability, traceability, and control. Because the system is capable of executing its own plan, rigorous evaluation and monitoring infrastructure are needed to ensure safety, especially in regulated environments.

Deployment Models: API Access vs. Self-Hosting

As LLMs become increasingly integral to financial workflows, institutions must carefully consider *how* these models are deployed. Deployment choices involve trade-offs across cost, control, compliance, performance, and security—particularly when dealing with sensitive or proprietary data. Broadly, there are two primary approaches to deploying LLMs: *commercial API access* and *self-hosting open-source models*.

Commercial API Access

The most accessible way to use LLMs is through cloud-based APIs provided by such companies as OpenAI, Anthropic, Google, or Cohere. These models are hosted on proprietary infrastructure and accessed via subscription or usage-based pricing. They offer the following advantages:

- *Ease of integration:* These models allow for quick setup with standard APIs.
- *Performance:* They provide access to state-of-the-art models trained on massive datasets and enhancing models with additional tools, such as internet search or code execution.
- *Scalability:* Providers manage infrastructure, allowing dynamic scaling of workloads.

The models have limitations as well, however:

- *Data privacy concerns:* Sensitive or proprietary information must be transmitted to third-party servers, raising concerns about confidentiality, data retention, and compliance with data protection regulations.
- *Lack of control:* The model architecture, parameters, training data, and update schedule are fully managed by the provider, limiting transparency and customization.
- *Ongoing cost exposure:* Pay-as-you-go pricing can become expensive at scale, especially for high-frequency or latency-sensitive applications.

API-based access is suitable for exploratory research, public-facing applications, and tasks that do not involve proprietary financial data. It is often insufficient, however, for enterprise-grade use cases that require full data custody and customized model behavior.

Self-Hosting and Open-Source Models

An alternative is to deploy open-source LLMs, such as *LLaMA*, *Mistral*, *Falcon*, or *GPT-J*, on private infrastructure. This approach allows organizations to run inference and fine-tuning on their own servers—either on premises or in private cloud environments—and provides the following advantages:

- *Data security and compliance:* No data can leave the organization's environment, which is critical when handling client information, trading strategies, or internal documents.
- *Customizability:* Models can be fine-tuned, distilled, or augmented to meet domain-specific requirements or to improve performance on financial texts.
- *Cost control over time:* Although upfront compute and staffing costs are high, long-term operational costs may be lower than continued API usage at scale.

This approach has two main limitations:

- *Infrastructure complexity:* Running large models requires specialized hardware (e.g., GPUs), as well as specialized human capital for deployment, scaling, and monitoring.
- *Lag behind frontier models:* Open-source models often trail commercial models in absolute performance, although the gap has narrowed considerably in recent releases.

Hybrid and Strategic Considerations

Many institutions adopt *hybrid strategies*, using commercial APIs for low-risk or generic use cases and deploying internal models for high-sensitivity or proprietary tasks. Two examples follow:

- Use ChatGPT to summarize public market news.
- Use an in-house model to analyze internal credit memos or regulatory filings.

Ultimately, the choice of deployment model must align with the organization's data governance policies, use-case sensitivity, latency requirements, and long-term cost structure. In finance—where intellectual property, compliance, and risk management are paramount—these considerations are not peripheral but central to the successful deployment of LLMs.

Applications of LLMs in Finance

Sentiment analysis and topic modeling have long been central to the application of NLP in finance. Seminal works, such as Tetlock's (2007) study on the predictive power of media pessimism and Loughran and McDonald's (2011) research on domain-specific sentiment dictionaries, showed that textual sentiment contains information relevant to returns, volatility, and trading behavior. Early methods relied on lexicons, dictionaries, or simple machine learning classifiers, but these techniques have evolved alongside the growing sophistication of NLP models, as discussed previously.

In this section, I outline how LLMs have opened new frontiers in financial NLP—moving beyond traditional tasks toward applications in compliance, ESG monitoring, risk surveillance, and financial summarization. I then present a practical tutorial demonstrating how LLMs

can be implemented for sentiment analysis, focusing on open-source models accessible to practitioners and researchers. Although LLM applications extend much further, this hands-on example provides a foundational approach for integrating LLM techniques into financial workflows.

Classic NLP Tasks and How LLMs Changed Them

Before the rise of LLMs, many core financial NLP tasks—such as named entity recognition (NER), event extraction, and relation extraction—were tackled using rule-based methods, feature engineering, or early machine learning models, such as conditional random fields (CRFs). For example, Wang, Xu, Liu, Gui, and Zhou (2014) applied CRFs augmented with domain-specific dictionaries to identify company names, tickers, and monetary amounts in financial news, and Yang, Chen, Liu, Xiao, and Zhao (2018) developed document-level event extraction systems for the Chinese stock market.

With the advent of LLMs, many of these tasks have become almost trivial: Modern models can perform zero-shot or few-shot NER, relation extraction, or event detection without task-specific architectures or large, labeled datasets. Recent evaluations (e.g., Lu and Huo 2025) have shown, however, that although general-purpose LLMs perform reasonably well, fine-tuned domain-specific models still outperform them in precision-critical extraction tasks, especially when dealing with subtle distinctions in entity types or regulatory language.

New Frontiers Enabled by LLMs

LLMs have significantly expanded the scope of NLP applications in finance.

- *Compliance monitoring and regulatory reporting:* Hillebrand, Berger, Deußer, Dilmaghani, Khaled, Kliem, Loitz, et al. (2023) introduced ZeroShotALI, demonstrating that LLMs can match financial documents against complex regulatory requirements in a zero-shot setting, potentially automating parts of compliance checks and audits.
- *ESG analysis:* Mehra, Louka, and Zhang (2022) fine-tuned BERT to develop ESGBERT, a model capable of classifying and extracting ESG-related content from sustainability and CSR reports, improving the automated detection of environmental, social, and governance themes.
- *Risk monitoring via news and events:* Guo, Jamet, Betrix, Piquet, and Hauptmann (2020) built ESG2Risk, a pipeline using LLM-based models to track ESG-related news and predict abnormal stock volatility, illustrating how LLMs can detect material nonfinancial risks from unstructured text.
- *Financial document summarization:* Kim, Muhn, and Nikolaev (2023) showed that ChatGPT can effectively summarize long, complex 10-K filings, helping investors distill key risks and opportunities. Their results suggest that LLM summarization improves market efficiency by reducing information asymmetry.

These examples illustrate that LLMs are no longer limited to traditional classification or extraction. They now support more complex workflows involving multistep reasoning, summarization, and regulatory interpretation.

Sentiment Analysis, Topic Modeling, and Quantitative Investing

Sentiment analysis has historically been one of the most widely used NLP applications in quantitative finance. Early approaches, such as FinBERT (Araci 2019), fine-tuned BERT models on financial news and earnings calls, offering more accurate sentiment classifications than traditional dictionary-based methods. These sentiment signals became important features in quantitative models, helping improve predictions of stock returns or volatility.

Recent advances in LLMs have expanded how text-derived signals are used in investing—moving beyond simple sentiment scores to richer representations that directly predict returns or explain price movements. For example, Lopez-Lira and Tang (2023) tested whether prompting generative LLMs, such as GPT-4, on company news headlines could predict next-day stock returns. They found that even without fine-tuning, the model's outputs contained predictive signals, suggesting that LLMs implicitly capture financial sentiment and context well enough to inform trading strategies. This line of research connects closely with the work of Chen, Kelly, and Xiu (2024), who used LLM-derived *embeddings* as features for predicting returns across 16 international equity markets. They found that these embedding-based features, paired with a simple penalized linear model, improve the risk-adjusted returns of the resulting portfolios over traditional approaches.

Another emerging approach focuses on organizing and clustering the patterns found in textual data. Cong, Liang, Zhang, and Zhu (2024) introduced the idea of *textual factors*—interpretable clusters of themes or topics identified by applying clustering techniques to LLM-derived embeddings. These textual factors can be incorporated into economic and financial models, providing scalable, data-driven ways to analyze large sets of unstructured information, from macroeconomic news to corporate filings.

Wang, Izumi, and Sakaji (2024) combined generative prompting and quantitative modeling by using LLMs to produce explainable summaries of news, corporate events, or leadership tone, which were then transformed into structured *factors* for predicting stock movements. Together, these advances highlight the versatility of modern LLMs: They can be used to generate direct predictions, to build structured thematic clusters, or to extract interpretable features through carefully designed prompts.

LLMs in Practice

In this section, I demonstrate how to apply LLMs for sentiment classification using Python. To follow along with this tutorial, I recommend using Google Colab because it provides easy access to GPUs (i.e., speeding up model outputs) and a reproducible environment for testing.

When working with LLMs, it is important to be familiar with Hugging Face,[3] the platform where open-source datasets and language models are shared. The models here range from pretrained LLMs to models fine-tuned from those pretrained LLMs. Hugging Face has two Python packages that can be used to easily load models and datasets: *transformers* and *datasets*.

In this exercise, we will use the Financial Phrasebank dataset introduced by Malo, Sinha, Korhonen, Wallenius, and Takala (2014). This dataset consists of 4,840 sentences randomly

[3]For more information, visit https://huggingface.co/.

sampled from financial news, with each sentence being labeled by a set of annotators as "positive," "negative," or "neutral."

I will demonstrate two methods for sentiment analysis using LLMs: (1) a fine-tuned feature extraction model using SBERT[4] and (2) a pretrained language model using OpenAI's API.

SBERT Feature Extraction Model

SBERT (Reimers and Gurevych 2019) is a BERT-based model that specializes in sentence-level embeddings. That is, text is input to the model and a vector is returned. Because sentiment scores are not directly output by the model, we need to train the model to predict sentiment scores. To do so, we create a training and test set (in practice, we would use a validation set for hyperparameter tuning).

```
>> !pip install -U datasets

>> from datasets import load_dataset
>> data = load_dataset("financial_phrasebank","sentences_50agree")
>> data = data['train'].to_pandas()

>> train = data.sample(frac=0.8, random_state=200)
>> test = data.drop(train.index)
```

SBERT can be loaded through the *sentence-transformers* Python package, which handles much of the preprocessing required to extract vector embeddings. Next, I show how to install the package, load the model, and extract embeddings for each sentence in the training and test datasets. Note that it is necessary to split the text into chunks or "batches" because computational restraints prohibit evaluating all sentences at once.

```
>> !pip install sentence-transformers

>> from sentence_transformers import SentenceTransformer
>> model = SentenceTransformer('all-MiniLM-L6-v2')

>> def batch_sentences(sentences,batch_size=100):
    batch_sentences = []
    for i in range(0,len(sentences),batch_size):
        batch_sentences.append(sentences[i:i+batch_size])
    return batch_sentences

train_batches = batch_sentences(train['sentence'].tolist())
test_batches = batch_sentences(test['sentence'].tolist())

X_train = [model.encode(batch) for batch in train_batches]
X_test = [model.encode(batch) for batch in test_batches]
```

[4]Available at https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.

Once the embeddings are loaded, they can be concatenated to formulate a matrix of size $n \times 384$. You can then fit a ridge classification model on those embeddings to create a mapping between embedding vectors to sentiment labels. Predictions on the test set are obtained by passing the test embeddings through the fitted model.

```
>> import numpy as np
>> from sklearn.linear_model import RidgeClassifier

>> # Train model
>> clf = RidgeClassifier()
>> clf.fit(np.vstack(X_train),train['label'])

>> # Predictions: 0 = Neg, 1 = Neut, 2 = Pos
>> preds = clf.predict(np.vstack(X_test))
```

In practice, you would want to create a validation set to fit the model with different penalization terms (i.e., alpha terms) in the ridge regression to determine the parameter that produces the best fit. For this example, I simply demonstrate obtaining predictions.

OpenAI ChatGPT

Although open-source LLMs can be run using the transformers package in Python, I will focus on an easy-to-implement pipeline using OpenAI's API. Instead of sourcing computational resources or getting caught up in the technical aspects of running LLMs, you can instead rely on an API-based approach to do this for you. The idea is that you can send your prompted text to the API, and the output from the model will be returned.

Because I am using a sophisticated, pretrained LLM like GPT-3.5, I do not need to train the model but instead simply prompt the text for the model to respond with a sentiment score:

```
>> prompt = """Evaluate the sentence below and determine the sentiment
score.
Respond with either positive, neutral, or negative.

Sentence: %sentence

Response:"""

>> prompted_sentences = [
prompt.replace("%sentence",i) for i in data.sentence
]
```

I use this prompt because it assists in (1) reducing the number of output tokens that I need to obtain from the model (i.e., output tokens are more expensive to produce) and (2) easy parsing of the response because the model will respond immediately with solely "positive," "neutral," or "negative." If you do not prompt it to provide a "Response:" at the end, the model typically provides some preamble to its actual response, making the response more costly as well as harder to parse.

To obtain responses from the API, the *openai* Python package can be used. To obtain responses from the API, go to OpenAI's API platform[5] and generate an API key, which associates your input with your account for billing. Then, obtaining sentiment scores is as easy as looping through the list of prompted sentences and parsing the response.

```
>> !pip install openai
>> from openai import OpenAI
>> client = OpenAI(api_key="ENTER_API_KEY_HERE")
>> # Define function to query model
>> def get_gpt_sentiment(sentence,model_name="gpt-3.5-turbo"):
      response = client.responses.create(
          model="gpt-3.5-turbo",
          input=sentence
      )
      return response.output_text
>> responses = [get_gpt_sentiment(i) for i in prompted_sentences]
>> responses[0] # = "Neutral"
```

Risks, Challenges, and Considerations

As financial institutions adopt LLMs into their workflows, it is important to be aware of the risks and challenges of doing so. This section outlines several of the most pressing challenges: output reliability, evaluation standards, model leakage over time, and legal uncertainty.

Hallucinations and Output Reliability

LLMs are probabilistic models designed to predict the next most likely word given the previous sequence of words. Although this ability makes them flexible and powerful, it also means they are prone to producing *hallucinations*—text that is fluent, plausible, and confidently stated but factually incorrect or fabricated.

A now-famous example involves a lawyer who used ChatGPT to draft a legal filing. The model generated convincing citations to court cases that, upon inspection, did not exist. This episode underscores the difficulty in relying on LLMs in domains where factual precision is essential—such as regulatory interpretation, earnings analysis, or compliance reporting.

One solution is the use of RAG frameworks, discussed earlier, which allow the model to consult an external knowledge base and leverage the relevant information in the knowledge base

[5]Visit https://openai.com/api/.

in crafting its response. Even with RAG, however, verifying that the model used the information appropriately remains a challenge.

In practice, evaluating output reliability often requires *secondary evaluation mechanisms* such as the following:

- *Human review*, particularly for complex or high-impact use cases
- *Secondary LLMs*, which can be prompted to judge whether the model or workflow being evaluated is generating the expected output and/or sourcing the correct information based on the prompt

Evaluating output becomes even more challenging in *multi-agent pipelines*—systems where different LLMs or modules hand off intermediate outputs. If one model produces a hallucinated output, that output may silently propagate through the system, breaking task logic or triggering unintended behaviors. Thus, output validation and fallback systems are essential components in production-grade deployments.

Evaluation in Domain-Specific Contexts

In general NLP research, model capabilities are evaluated using benchmark datasets, which are labeled datasets used to determine LLM performance in different contexts or tasks. Domain-specific evaluation in finance remains an underdeveloped area, however. Most financial tasks—such as interpreting a 10-K filing, generating risk summaries, or classifying ESG disclosures—lack standardized ground truth labels.

This situation creates several challenges:

- *No universal "correct" answer* exists for such tasks as summarization or narrative analysis.
- *Crafting an evaluation metric* is not as straightforward as evaluating classification performance. Often, one must be creative in constructing a single metric to determine performance or reliability in a pipeline.
- *Task-specific benchmarks* must often be constructed in house, with significant human effort.

For such applications as return prediction or sentiment classification, evaluation may be grounded in market response (e.g., future returns). But for more qualitative tasks—such as policy interpretation or stock recommendations—quantifying model performance requires human expertise and domain fluency. Without rigorous evaluation, firms risk deploying models whose performance may degrade silently under new conditions or inputs.

Forward-Looking Contamination in Backtests

Another risk in using LLMs for alpha signal generation is the issue of *forward-looking bias*. Most large LLMs have been pretrained on internet-scale text up to a certain cutoff date. If that training corpus includes data from the backtest period, then the model may "know" future outcomes during historical simulations, even if inadvertently.

For instance, if a model is used to predict the sentiment or return of stocks using news headlines from 2010, the model may implicitly encode information that was not available at the time. That is, if a news article has the headline "NVIDIA Introduces X Chip" and the model implicitly knows that this chip was very popular from forward-looking training data, this will inherently bias the model's response to be more positive.

Some studies have noted, however, that this forward-looking information can sometimes negatively influence out-of-sample performance because the model is distracted by the future narratives and information, not focusing on the current information available. Glasserman and Lin (2024) referred to this phenomenon as the *distraction effect*.

Compliance, Data Security, and IP Risk

As discussed earlier in the context of deployment models, the use of third-party APIs for LLM inference raises serious compliance concerns. But beyond data privacy, a new class of legal and intellectual property (IP) issues is emerging around ownership and liability.

Key concerns include the following:

- *Ownership of model outputs:* Does the firm own the generated text? What if that text closely resembles a copyrighted source seen during pretraining?
- *Liability for misinformation:* If a model generates a misleading investment recommendation or incorrect regulatory interpretation, who is responsible?
- *Internal data leakage:* If proprietary data are used in prompts or fine-tuning and processed by a third-party model, the data may be inadvertently retained or exposed, depending on provider policies.

These concerns are amplified in finance, where data governance and regulatory compliance are not optional. Firms should establish clear policies around

- internal versus external LLM usage,
- data anonymization and minimization practices, and
- logging, audit trails, and human-in-the-loop reviews.

Conclusion

The advances in NLP have revolutionized financial workflows through the development of LLMs. Although traditional, task-specific NLP models found their way into earlier quantitative models, the flexibility granted by conversational models has made LLMs accessible to nearly every facet of the investment process. As the technology continues to advance, it is important to understand the different types of models, their applications, and implementation frameworks.

The power of these models comes with important caveats, however. Effective deployment requires an understanding of their architectural foundations, training processes, and the trade-offs involved in different implementation strategies. Moreover, issues of reliability,

interpretability, computational cost, compliance, and domain-specific evaluation demand careful consideration, particularly in high-stakes financial contexts.

The future of NLP in finance will likely be shaped by hybrid approaches that combine the generative power of LLMs with structured data pipelines and traditional time-series models. As models continue to evolve—integrating multimodal inputs, improving reasoning abilities, and adapting to specialized domains—success will depend not only on technical sophistication but also on rigorous evaluation, sound governance, and a deep understanding of the financial domain.

In short, LLMs are a powerful technology that will make information assimilation in financial markets much faster and efficient. Like any powerful tool, however, their value lies not in raw capability but in careful, context-aware application.

References

Araci, Dogu. 2019. "FinBERT: Financial Sentiment Analysis with Pre-Trained Language Models." Working paper (27 August). doi:10.48550/arXiv.1908.10063.

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. 2020. "Language Models are Few-Shot Learners." In *NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems*, 1877–901. doi:10.48550/arXiv.2005.14165.

Chen, Yifei, Bryan T. Kelly, and Dacheng Xiu. 2024. "Expected Returns and Large Language Models." Working paper (23 August). https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4416687.

Cong, Lin William, Tengyuan Liang, Xiao Zhang, and Wu Zhu. 2024. "Textual Factors: A Scalable, Interpretable, and Data-Driven Approach to Analyzing Unstructured Information." NBER Working Paper 33168 (November). doi:10.3386/w33168.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–86. doi:10.48550/arXiv.1810.04805.

Glasserman, Paul, and Caden Lin. 2024. "Assessing Look-Ahead Bias in Stock Return Predictions Generated by GPT Sentiment Analysis." *Journal of Financial Data Science* 6 (1): 25–42. doi:10.3905/jfds.2023.1.143.

Guo, Tian, Nicolas Jamet, Valentin Betrix, Louis-Alexandre Piquet, and Emmanuel Hauptmann. 2020. "ESG2Risk: A Deep Learning Framework from ESG News to Stock Volatility Prediction." Working paper (5 May). doi:10.48550/arXiv.2005.02527.

Hillebrand, Lars, Armin Berger, Tobias Deußer, Tim Dilmaghani, Mohamed Khaled, Bernd Kliem, Rüdiger Loitz, et al. 2023. "Improving Zero-Shot Text Matching for Financial Auditing with Large Language Models." In *DocEng '23: Proceedings of the ACM Symposium on Document Engineering 2023*, 1–4. doi:10.1145/3573128.3609344.

Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzi Li, Shean Wang, and Weizhu Chen. 2021. "LoRA: Low-Rank Adaptation of Large Language Models." Working paper (16 October). doi:10.48550/arXiv.2106.09685.

Kim, Alex G., Maximilian Muhn, and Valeri Nikolaev. 2023. "Bloated Disclosures: Can ChatGPT Help Investors Process Financial Information?" Working paper, Becker Friedman Institute for Economics (26 April). doi:10.48550/arXiv.2306.10224.

Lopez-Lira, Alejandro, and Yueha Tang. 2023. "Can ChatGPT Forecast Stock Price Movements? Return Predictability and Large Language Models." Working paper, University of Florida (6 April). doi:10.2139/ssrn.4412788.

Loughran, Tim, and Bill McDonald. 2011. "When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks." *Journal of Finance* 65 (4): 35–65.

Lu, Yi-Te, and Yintong Huo. 2025. "Financial Named Entity Recognition: How Far Can LLM Go?" Working paper (4 January). doi:10.48550/arXiv.2501.02237.

Malo, P., A. Sinha, P. Korhonen, J. Wallenius, and P. Takala. 2014. "Good Debt or Bad Debt: Detecting Semantic Orientations in Economic Texts." *Journal of the American Society for Information Science and Technology* 65 (4): 782–96. https://doi.org/10.1002/asi.23062.

Mehra, Srishti, Robert Louka, and Yixun Zhang. 2022. "ESGBERT: Language Model to Help with Classification Tasks Related to Companies' Environmental, Social, and Governance Practices." Working paper (31 March).

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. "Distributed Representations of Words and Phrases and Their Compositionality." Working paper (16 October). doi:10.48550/arXiv.1310.4546.

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. "Improving Language Understanding by Generative Pre-Training." OpenAI. www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. "Language Models Are Unsupervised Multitask Learners." OpenAI. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Reimers, Nils, and Iryna Gurevych. 2019. "Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks." Working paper (27 August). doi:10.48550/arXiv.1908.10084.

Tetlock, Paul C. 2007. "Giving Content to Investor Sentiment: The Role of Media in the Stock Market." *Journal of Finance* 62 (3): 1139–68. doi:10.1111/j.1540-6261.2007.01232.x.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need." In *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6000–10.

Wang, Meiyun, Kiyoshi Izumi, and Hiroki Sakaji. 2024. "LLMFactor: Extracting Profitable Factors Through Prompts for Explainable Stock Movement Prediction." Working paper (16 June). doi:10.48550/arXiv.2406.10811.

Wang, Shuwei, Ruifeng Xu, Bin Liu, Lin Gui, and Yu Zhou. 2014. "Financial Named Entity Recognition Based on Conditional Random Fields and Information Entropy." In *2014 International Conference on Machine Learning and Cybernetics*, 838–43. doi:10.1109/ICMLC.2014.7009718.

Yang, Hang, Yubo Chen, Kang Liu, Yang Xiao, and Jun Zhao. 2018. "DCFEE: A Document-Level Chinese Financial Event Extraction System Based on Automatically Labeled Training Data." In *Proceedings of ACL 2018, System Demonstrations*, 50–55.

MACHINE LEARNING IN COMMODITY FUTURES: BRIDGING DATA, THEORY, AND RETURN PREDICTABILITY

Tony Guida
Senior Quantitative Analyst, R&D, Atonra

During the past decade, machine learning (ML) methods have become foundational in quantitative equity investing, where datasets spanning traditional fundamentals to alternative signals have enabled the modeling of nonlinear relationships and cross-sectional return drivers. In contrast, commodity futures remain an underexplored frontier for ML applications, despite the growing maturity of commodities as a financialized asset class and their rising prominence in institutional portfolios.

This asymmetry arises from multiple structural frictions. First, commodities are not capital assets in the traditional sense. They are tangible goods, often classified as "consumption assets" (Lucas 1978), and lack a balance sheet or accounting disclosure framework comparable to that of stocks. Their value is shaped by physical supply chains, geopolitical disruptions, seasonality, and idiosyncratic features, such as storage costs or delivery mechanisms—factors that are harder to quantify and less standardized than firm-level financial fundamentals. Second, the academic literature in commodities (Fama and French 2015; Gorton and Rouwenhorst 2006, among others) has long been dominated by reduced-form econometric models, largely relying on price-based inputs for one or a specific group of contracts (e.g., metals, agricultural). These models are often constrained by past returns and/or technical indicators derived from the futures curve.

And yet, commodity futures offer a highly attractive laboratory for modern predictive modeling. Unlike equities, commodity prices are tightly linked to macroeconomic conditions, exhibit abrupt regime shifts, and are influenced by distinct classes of agents, hedgers, producers, and speculators, each with different risk preferences and trading constraints. The pricing literature has already surfaced a range of theoretical factors, such as momentum, basis, hedging pressure, open interest, and skewness, which could be embedded into ML pipelines to forecast returns. Despite the diverse range of potential signals stemming from the literature on factors, most ML studies on commodities remain narrow in scope, focusing either on a single commodity group or on technical-only signals without theoretical foundations.

This chapter seeks to correct that imbalance. I follow a supervised learning framework familiar in equity ML research, where cross-sectional returns are predicted using engineered features grounded in theory and empirical evidence. I construct a wide signal database combining both theoretical premiums and traditional technical trading predictors variables. The implementation leverages a boosted tree ensemble to learn from these signals and generate daily predictions over eight target horizons across 41 commodity futures over a 30-year horizon. I then analyze the ML term structure, as shown in Blitz, Hanauer, Hoogteijling, and Howard (2023) for equity, and create a meta ensemble over the eight target horizons to mitigate target variable model

risk and reduce turnover, which is usually high with machine learning models applied to financial markets.

This chapter addresses three key gaps in the financial literature. I begin by taking an expansive view of feature engineering, grounding input construction in the academic commodity factor investing literature rather than relying exclusively on off-the-shelf technical indicators. Next, I extend the equity-based machine learning methodology to the construction of long–short signals in commodity futures. Finally, I introduce the concept of ensemble portfolios by predicting over multiple target horizons, effectively characterizing the pseudo-term structure of ML signals in commodities before aggregating them into a unified strategy.

The main findings can be summarized as follows. By training gradient boosted trees in an expanding window framework, I replicate the empirical asset pricing approach seen in equities, extracting feature importance scores as pseudo-betas that quantify the relative value of each signal through time. Momentum-based indicators, particularly time-series momentum, consistently emerge as dominant features, while skewness-based signals are more prominent at shorter horizons, supporting the relevance of both trend and reversal effects. Portfolio results exhibit a mirror J-shaped pattern in annualized returns, with low correlation between ultra-short-term and long-term models. This J-shaped configuration highlights the value of horizon diversification and justifies an ensemble approach, which delivers smoother returns, lower volatility, and improved drawdown control.

This chapter is organized as follows. I begin with a review of the theoretical and empirical foundations of commodity return predictability, covering key pricing models and documented anomalies, and assess the recent literature in ML commodities. I then describe the dataset, detailing the construction of input features and the machine learning methodology used throughout the analysis. The core empirical results follow, including model-level and ensemble performance, feature relevance over time, and portfolio-level attribution. The chapter concludes with a summary of the main insights and implications for machine learning–based commodity investing.

Seeking Commodity Features: Foundations of Commodity Factor Investing

Despite their narrower universe and shorter history of financialization, commodities benefit from a surprisingly rich theoretical and empirical foundation. Early work in the economics of commodity markets introduced two core theories that remain essential today: the theory of storage (Kaldor 1939; Working 1949) and the hedging pressure hypothesis (Keynes 1930; Hirshleifer 1988). These frameworks are not academic artifacts; they provide operational foundations for return predictability in futures contracts and have shaped the classification of commodity anomalies, such as basis, term structure slope, and inventory cycles.

The theory of storage ties the slope of the futures curve to physical inventories. High inventory levels imply low convenience yields and upward-sloping curves (contango), whereas low inventories lead to backwardation, offering a reward to those willing to hold scarce assets. In contrast, the hedging pressure hypothesis focuses on the positioning of commercial hedgers and speculative traders. When commercial players dominate one side of the market, risk

premiums tend to compensate the speculators taking the opposite position. These economic frictions manifest in observable features that can be engineered into factor models.

What makes commodities distinct and appealing from a quantitative perspective is their structural heterogeneity, showing even negative pairwise correlations. Energy, metals, and agricultural products are driven by different microeconomics and subject to disjoint supply shocks. And yet, empirical work has shown that a coherent set of predictors can explain return differentials across commodities. Fama and French (2015) and Gorton and Rouwenhorst (2006) established that traditional factors, such as basis and momentum, carry explanatory power in a cross-sectional framework. More importantly, these predictors are grounded in the economics of storage, risk transfer, and price discovery. The financialization of commodity markets, reflected in the growth of commodity index funds and increased institutional flows, cemented their status as an investable asset class. Part of this appeal stemmed from diversification. Commodities have historically exhibited low correlations with stocks and bonds, especially in inflationary environments. Precious metals in particular show positive co-movement with unexpected inflation (Bhardwaj, Gorton, and Rouwenhorst 2015).

Commodity Anomalies, Liquidity, and Technical Indicators

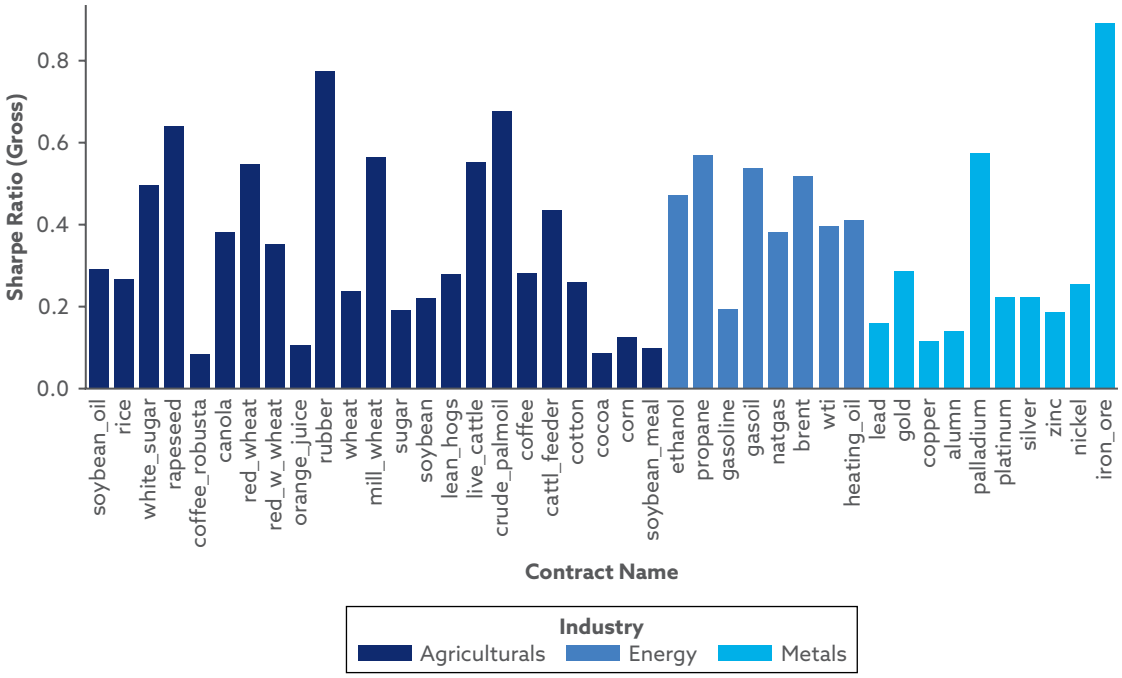
Although commodities are widely recognized for their diversification benefits, a growing body of empirical research highlights persistent return anomalies in futures markets that go beyond passive exposure. These anomalies, grounded in momentum, carry, skewness, and basis, reflect systematic inefficiencies that can be exploited through well-structured, rule-based strategies and that have been harvested as standalone quantitative investment strategies products. Building on this foundation, I constructed nine families of predictive features that span both price-based dynamics and market microstructure signals.

Momentum and time-series momentum capture recent return behavior and trend persistence. Carry and basis signals extract information from the slope and curvature of the futures curve, while basis momentum extends this insight by comparing relative performance across nearby maturities. Skewness proxies for asymmetries in return distributions that may reflect sentiment extremes or positioning risk. Additional signals include idiosyncratic volatility, which isolates contract-specific risks, and liquidity (open interest), which measures market participation and capital flows. Finally, classical technical rules, such as moving averages and relative strength index (RSI) filters, are included to benchmark modern signals against widely used technical heuristics.

Momentum: The Trend Is Your Friend

Momentum is arguably the most robust and widely validated anomaly across asset classes, and commodities are no exception. Two forms are documented: cross-sectional and time-series momentum. Cross-sectional momentum looks at how commodities perform relative to each other. The idea is simple: Go long the winners and short the losers. Miffre and Rallis (2007) showed that sorting commodities according to their past 12-month returns produces a long–short portfolio with strong and statistically significant excess returns. This effect survives standard risk adjustments, indicating that it is not only compensation for known exposures.

Exhibit 1. Time-Series Momentum for Each Futures Contract, January 1994–June 2024



Source: Bloomberg LLC.

In contrast, time-series momentum, shown in **Exhibit 1**, treats each commodity on its own. If a given contract has risen during the past year, it is likely to continue to rise, and if it has been falling, one can expect more of the same. Moskowitz, Ooi, and Pedersen (2012) implemented this logic by going long commodities with positive 12-month returns and short those with negative ones. The result was highly consistent profits across dozens of markets. This approach is the backbone of many trend-following commodity trading advisers (CTAs) and managed futures funds, proving its relevance in live strategies.

Why does momentum work? The explanations for this phenomenon range from behavioral biases, such as herding and anchoring, to structural frictions that slow information diffusion. Importantly, momentum in commodities is not a market-specific quirk. Rather, it is part of a global anomaly observed in equities, bonds, and currencies alike (Asness, Moskowitz, and Pedersen 2013).

Carry and Basis: The Term Structure as a Predictor

The term structure of futures prices offers valuable insight into the expectations and positioning of various economic agents. By examining the relationship between a commodity's futures price and its spot price or by comparing prices across different contract maturities, it is often possible to infer directional signals about future returns. This information is encapsulated

in the carry or basis, which serves as a key predictive signal in many empirical asset pricing frameworks.

When futures are in backwardation (near-term futures priced below spot), it often signals supply stress and low inventories. According to the theory of storage (Kaldor 1939; Working 1949), backwardation reflects a high convenience yield, an implicit benefit to holding the physical commodity. In this context, going long earns you a premium for providing liquidity or inventory services.

Conversely, contango (when futures are above spot) reflects surplus supply and lack of urgency to hold the good. Long positions in such markets often lose value as contracts roll forward. Gorton, Hayashi, and Rouwenhorst (2013) confirmed with decades of data that commodities with tight inventories and high basis tend to earn higher returns. The hedging pressure hypothesis offers a complementary view. Commercial players, such as producers, hedge their exposure by shorting futures. Speculators willing to take the long side demand a risk premium. This dynamic leads to predictable drift in futures prices over time, typically upward, benefiting long holders. Empirical studies (e.g., de Roon, Nijman, and Veld 2000; Basu and Miffre 2013) have shown that long–short strategies based on hedging pressure or carry signals capture substantial excess returns. In practice, a simple carry strategy involves ranking commodities by their basis and going long the top quantile and short the bottom.

Skewness: When Asymmetry Matters

A more recent and nuanced anomaly is return skewness. Commodity prices are prone to spikes from geopolitical events, weather, or supply chain shocks. This tendency creates asymmetry in returns, with some markets showing fat right tails (lottery-like upside) and others, left tails (sudden crashes). Fernandez-Perez, Frijns, Fuertes, and Miffre (2018) found that this asymmetry contains predictive value. Commodities that have experienced negative skewness—frequent small gains with rare large losses—tend to deliver higher future returns. The intuition is risk based: Investors require a premium to bear downside tail risk. In contrast, positively skewed commodities (a small chance of windfall gains) attract overconfident or risk-seeking buyers, pushing prices up and expected returns down.

A skewness-based strategy, long on negatively skewed and short on positively skewed futures, earns abnormal returns not explained by momentum or carry. Although harder to implement because of higher-moment estimation and signal instability, this anomaly highlights that tail risks matter—that financial markets price the full shape of the return distribution, not only variance.

Idiosyncratic Volatility: The Role of Unexplained Risk

Idiosyncratic volatility (IVOL) refers to the volatility of a commodity's returns that is not explained by broad market or factor movements. In equity markets, high idiosyncratic volatility is often associated with lower subsequent returns (the IVOL anomaly), and a similar phenomenon has been explored in commodities. Early studies found that commodity futures with high past idiosyncratic volatility tend to underperform, suggesting a negative relation between IVOL and future returns analogous to equities' low-volatility anomaly. In the context of commodity futures, IVOL captures risks that are market specific, such as supply shocks, seasonal patterns, or logistical disruptions, that do not co-move with the broader commodity complex.

Fuertes, Miffre, and Fernandez-Perez (2015) explored the role of IVOL in commodity markets, showing that its integration into multisignal frameworks can improve portfolio diversification. For instance, combining IVOL with momentum and term structure signals allows for the construction of more robust long–short strategies by exploiting complementary sources of information. Fernandez-Perez et al. (2018) further analyzed IVOL in the presence of established commodity factors, highlighting its relevance for understanding commodity-specific risk beyond systematic drivers. Although IVOL use as a standalone signal is less straightforward, its interaction with other structural features of the market is worth investigating in an ML research context.

Basis Momentum: Riding the Curve's Slope

Basis momentum is a recently identified predictor that exploits information in the shape of the futures curve. It is defined as the difference between the momentum of near-term futures and the momentum of deferred (second-nearby) futures. This factor effectively captures the slope and curvature dynamics of the term structure: A strategy going long commodities whose nearby contracts have outperformed their second-nearby contract (and shorting those with the opposite pattern) earns significant returns. Boons and Prado (2019) introduced basis momentum and showed it strongly outperforms traditional signals, such as simple momentum or carry (basis) in predicting commodity returns. In their findings, a portfolio sorted on basis momentum produced large spread returns both in cross-section and time series, indicating that this maturity-specific momentum effect is a powerful anomaly. The basis momentum premium appears to be related to volatility and segmentation across contract maturities. For example, when speculators' risk-bearing capacity is strained, mispricing can occur between near and far contracts, which basis momentum strategies exploit. Follow-up studies have confirmed the robustness of this factor. Paschke, Prokopczuk, and Simen (2020), for instance, documented a similar "curve momentum" strategy (operating on the first two contract maturities) that achieves high Sharpe ratios and positive alpha after controlling for other factors.

Open Interest: Reading the Crowd

Open interest represents the total number of outstanding contracts (long or short) in a futures market and is often viewed as a gauge of market participation and liquidity. Uniquely, open interest is not a price-based signal but a quantity-based measure that can reflect the ease with which risk is absorbed by the market. Research by Hong and Yogo (2012) showed that movements in open interest contain valuable information about commodity risk premia. Specifically, they found that increases in open interest tend to predict lower future returns for commodity contracts, whereas declining open interest predicts higher future returns. This pattern is consistent with the idea that when more investors (particularly speculators) enter a market, driving open interest up, they provide risk-bearing capacity and push risk premiums down. Conversely, when open interest dries up, remaining hedgers must offer higher expected returns to entice speculators to take the other side. Related work also has shown that open interest is highly procyclical and correlated with macroeconomic activity, reinforcing the interpretation that it signals the broad flow of investment into commodity markets. Moreover, the predictive power of open interest holds even alongside other predictors. Cheng, Kirilenko, and Xiong (2015) found that surges in participation (often captured by open interest or investor positions) can lead to "convective risk flows," affecting pricing beyond what fundamentals

alone would suggest. In practice, low-open-interest environments have been associated with subsequent positive commodity returns, as risk premiums rise to attract capital, whereas high-open-interest periods coincide with compressed returns.

Relative Strength Index: A Technical Oscillator

The relative strength index is a bounded oscillator designed to capture the velocity of recent price movements, typically using a 14-day window. In commodity futures, it is often used as a contrarian tool, with threshold levels above 70 interpreted as overbought and below 30 as oversold. Although RSI-based strategies are intuitive and widely adopted by practitioners, early academic evaluations found that the approach produced less reliable results compared with trend-following rules. Lukac, Brorsen, and Irwin (1988) reported limited profitability from standard RSI implementations across a broad set of commodity markets. More-recent studies, however, suggest that its performance can improve when combined with volume indicators or modified threshold levels. Yen and Hsu (2010) showed that hybrid strategies incorporating RSI and money flow signals perform competitively in certain commodity sectors, and Anderson and Li (2015) demonstrated that tuning RSI parameters enhances returns in energy and agricultural markets. Although the standalone signal may exhibit limited forecasting power in isolation, RSI remains a useful timing tool within multisignal frameworks, particularly for refining entries and exits in mean-reverting environments.

Moving-Average Crossovers: Trend Following in Practice

Moving-average crossover systems are among the most enduring and empirically supported technical rules in commodity futures markets. These strategies involve comparing a short-term moving average, commonly a 50-day window, with a longer-term average, such as 200 days. A buy signal is generated when the short-term average crosses above the long-term average, and a sell signal occurs on a downward crossover. This mechanism seeks to exploit persistent trends while avoiding short-term noise. Lukac et al. (1988) found that dual moving-average strategies produced statistically significant risk-adjusted returns across multiple commodity contracts. Park and Irwin (2007), in a comprehensive review of more than 90 studies, concluded that moving-average systems were consistently among the most robust technical rules in futures markets. Szakmary, Shen, and Sharma (2010) further validated the effectiveness of these rules across 28 commodity markets, showing that excess returns persist even after accounting for transaction costs and data-snooping concerns. Narayan, Ahmed, and Narayan (2015) also confirmed the signal's predictive value in a cross-sectional portfolio context, although they noted sensitivity to parameter tuning. These findings align with the broader time-series momentum literature, including the framework of Moskowitz et al. (2012), whose one-year trend filters echo the logic of long-horizon moving-average systems. Taken together, this body of work underscores the continued relevance of moving-average crossovers as reliable building blocks for systematic commodity trading strategies.

Existing Relevant Machine Learning in Commodity Markets

In this subsection, I review the most relevant recent developments in applying machine learning to commodity markets. Two key strands of literature are particularly relevant to the scope of this study.

- First, the integration of macroeconomic variables into ML-based models for commodity price forecasting has received growing attention, with numerous studies (Ben Jabeur, Khalfaoui, and Ben Arfi 2021; Wang and Zhang 2024) demonstrating the predictive value of macro-financial indicators.
- Second, a methodological distinction has emerged between univariate models, applied in isolation to individual commodities or subsets, such as metals, agricultural products, or energy, and cross-sectional (panel) approaches that model multiple commodities jointly.

I examine the main features and signals used in each stream, highlight the respective methodological trade-offs, and contrast their empirical findings. This review serves to position my own contribution, which builds long–short commodity portfolios using machine learning, in relation to these established approaches.

The Role of Macroeconomic Variables in ML-Based Commodity Forecasting Literature

Macroeconomic fundamentals are known to influence commodity markets through demand, supply, and investment channels. ML models often integrate these variables as predictors to capture broader economic signals that drive commodity price movements. Common macroeconomic features include indicators of global economic activity (e.g., industrial production growth or GDP), inflation rates, interest rates, exchange rates (especially for commodity-exporting countries' currencies), equity market indexes, and measures of liquidity or risk appetite (Gargano and Timmermann 2014; Costa, Ferreira, Gaglianone, Guillén, Issler, and Lin 2021). By including such variables, ML models aim to account for shifts in the business cycle and financial conditions that affect commodity prices. Early work by Gargano and Timmermann (2014) demonstrated the value of macro variables in commodity forecasting using traditional models. They found that certain predictors, such as commodity currency exchange rates (currencies of major commodity exporters), have significant short-horizon predictive power for commodity price indexes, while industrial production growth and investment ratios matter at longer horizons. Moreover, their results indicated that the predictive power of macroeconomic variables is regime dependent: Commodity return predictability was strongest during global recessionary periods, when macro conditions undergo substantial shift. This finding suggests that macro indicators help capture low-frequency economic trends and structural breaks.

Building on such insights, recent ML studies have incorporated large sets of macroeconomic features to improve forecast accuracy. Costa et al. (2021), for example, explored oil price forecasting with an extensive "big data" macro-financial dataset of 315 variables, combined with a suite of 23 modeling approaches, including tree-based ML (random forests, gradient boosting), regularized regressions, and classical econometric benchmarks. Their comprehensive pseudo-out-of-sample study showed that machine learning models leveraging rich macroeconomic information can significantly outperform naive benchmarks in the short run. In particular, at forecast horizons up to six months, such models as LASSO regression and tree-based ensembles (applied to macro and financial predictors alongside oil futures prices) yielded the most accurate forecasts, often achieving substantial gains in out-of-sample R^2 relative to a random walk. These improvements, on the order of two-digit percentage reductions in forecast error in some cases, underscore that ML algorithms can effectively exploit the predictive content of a broad array of macroeconomic indicators. At longer horizons (one to five years ahead), Costa et al. (2021) noted that no single ML method dominates but combinations of forecasts

and structural models become more relevant, implying macro variables still contribute when used in ensemble approaches.

Other studies have confirmed that adding macroeconomic features boosts commodity price forecasts in ML frameworks. Wang and Zhang (2024) examined 22 commodity futures and showed that including both commodity-specific variables and macroeconomic factors as inputs to ML algorithms improves out-of-sample performance for the majority of those commodities. In their experiments, a gradient boosting model (LightGBM) augmented with such features as global economic indexes, interest rates, and financial market variables produced lower prediction errors than simple autoregressive benchmarks—AR(1) models—in most cases. The authors used SHAP (SHapley Additive exPlanations) values to interpret the importance of features in the ML forecasts, finding that the most influential predictors vary across commodities. For instance, some commodities are driven strongly by general macro variables (such as an aggregate demand index or currency values), whereas others are more influenced by idiosyncratic factors (such as inventory levels or past price momentum). This heterogeneity highlights that macroeconomic inputs have significant predictive value overall, but their impact can be commodity specific.

In commodity markets that are highly sensitive to policy and global conditions, macroeconomic predictors may be especially critical. For example, Ben Jabeur et al. (2021) used an explainable ML approach to predict crude oil price crashes by integrating such variables as stock market indexes, currency exchange rates, and green energy indexes. Their findings indicated that macro-financial indicators improve the early warning signals for oil market downturns by capturing broad market sentiment and structural shifts. Likewise, Ampountolas and AlGharbi (2025), using a hybrid ML model for orange juice commodity prices, reported that including financial market indexes (e.g., the S&P 500 Index) alongside macro factors improved forecast accuracy. This finding aligns with Gargano and Timmermann's (2014) emphasis on macro predictors: Broad indexes likely capture global economic conditions and investor risk appetite, which translates into better predictions of commodity demand and pricing pressures.

Univariate vs. Cross-Sectional in ML Commodity Research: Global vs. Local

A second key dimension in the literature is whether researchers model commodity prices in isolation (univariate time-series approach) or leverage information across a cross-section or panel of commodities. The univariate approach entails building a separate predictive model for each commodity's price or return, using that commodity's own lagged values and potentially some exogenous features (which could include macro variables or that commodity's specific fundamentals). In contrast, cross-sectional (or panel) approaches involve modeling multiple commodities jointly—for example, by pooling data across commodities to train a single model or by predicting the entire cross-section of commodity return at each point in time using common predictors (similar to factor investing models).

Each approach has distinct methodological trade-offs, and has been explored in recent ML-based commodity research. Univariate ML models are common in studies focusing on a particular commodity or commodity index group. Many studies take this route for major commodities, such as crude oil or gold, often comparing various ML algorithms to find the best performer for that single series. For instance, Foroutan and Lahmiri (2024) implemented 16 different machine learning and deep learning models to forecast prices of four individual markets (West Texas intermediate crude oil, Brent oil, gold, and silver), essentially treating each market as a separate

forecasting task. They found that advanced neural networks, such as temporal convolutional networks or Bi-GRU networks, could outperform other models for certain commodities, while tree-based models, such as LightGBM, also performed strongly as a baseline.

The point of the univariate approach is that it allows specialized modeling of each commodity's unique dynamics, capturing specific supply shocks, seasonal patterns, or specific economic linkages (such as oil's linkage to oil inventory levels or gold's inverse relationship with interest rates) without being "diluted" by data from other commodities. Researchers can include commodity-specific features (such as metal inventories, energy rig counts, and weather variables for crops) alongside macro variables tailored to that market. This approach often improves interpretability for that commodity. One drawback, however, is that the data for each commodity are limited, which can constrain complex ML models—and most of the time, those data (like the macro variables) are lagged and their granularity is at best monthly. Overfitting is a risk, and the model might fail to generalize if the training sample (sample of one commodity) is small or if structural changes occur.

In contrast, cross-sectional and panel ML approaches attempt to "learn" across multiple commodities simultaneously, leveraging the idea that different commodities may share common patterns or factors. Cross-sectional modeling for forecasting is widespread in equities. In this approach, one might use characteristics or factors of each commodity (such as the family of factors previously introduced) at time t to predict the cross-section of returns in the next period. ML algorithms can be used to combine these commodity-specific characteristics in a nonlinear way.

Angelidis, Sakkas, and Tessaromatis (2025) followed this approach by comparing time-series models for individual commodities versus cross-sectional models that predict all commodities' returns jointly. They examined 37 commodity futures and found that cross-sectional models produce superior forecasts compared with time-series models for commodity returns. In other words, a model that pools information across commodities (and exploits cross-sectional predictors, such as each commodity's basis, momentum, or other characteristics at a given time) achieved lower overall prediction error than modeling each commodity in isolation. The authors reported that combining forecasts and using modern ML methods further enhanced performance, suggesting that the cross-sectional signals contain exploitable structure that ML methods can capture more effectively relative to traditional approaches. Intuitively, cross-sectional models benefit from wider information sets. They can detect, for example, that commodities with certain traits (say, a high prior-year return or a negative skewness) tend to mean-revert or continue trending, based on patterns learned from the full panel of commodities. This dynamic can improve generalization because the model draws on a larger sample of observations (multiple commodities over time) to learn predictor–response relationships. There are several methodological trade-offs between univariate and cross-sectional approaches. Generalization versus specialization is a key consideration.

Cross-sectional ML models effectively assume that different commodities' price dynamics share some underlying functional form or factors, which the model can learn. This pooling can dramatically increase the effective sample size for training, aiding generalization and reducing estimation noise. Indeed, the superiority of the cross-sectional ML in Angelidis et al. (2025) echoes findings in other asset classes (e.g., equities) that "global" models using panel data can outperform many separate "local" models by borrowing strength across series. Nonetheless, the optimal approach may depend on the context: If the forecasting goal is to rank or allocate

across commodities (long–short portfolio construction), panel ML models are naturally suited. If the goal is to predict exact price movements of a particular commodity (for hedging purposes), however, a univariate model with that commodity's key drivers might be better suited.

Empirical Results

This section presents the empirical evaluation of the proposed framework. I begin by describing the dataset, feature construction, and methodological setup used for forecasting across multiple horizons. Next, I analyze the information content and relative importance of each feature within boosted tree–based models to understand how predictive drivers vary with horizon length. Finally, I assess the out-of-sample performance of individual models and ensemble approaches through long–short portfolio simulations, comparing their risk-adjusted returns and analytics across forecast horizons.

Data and Methodology

The sample universe consists of 41 daily continuous futures contracts: 23 agricultural commodities, 8 energy products, and 10 metals, as shown in **Exhibit 2**. I sourced prices and open interest data from Bloomberg for the first and second contracts along the term structure, which were used to construct the features. For each contract, I computed a continuous price series using the back-adjusted ratio methodology. This approach applies a multiplicative adjustment five days prior to each contract roll, using the ratio between the next and previous contract prices to rescale all preceding prices. This process ensures that returns and all price-based features remain consistent across the series, avoiding artificial jumps caused by roll effects. For return-based features, all prices not denominated in US dollars were converted to US dollars. When I computed price ratios, however, contracts were maintained in their local currency to minimize the influence of exchange rate movements on daily price dynamics.

As outlined in the previous section, I constructed the feature set on the basis of both theoretical and practical insights from the literature. Theoretically motivated features include carry, time-series momentum, cross-sectional momentum, basis momentum, and idiosyncratic volatility, which are well documented in empirical asset pricing studies. Practically motivated signals were drawn from widely used technical analysis heuristics, such as moving average crossovers, the relative strength index, and moving averages of open interest, reflecting strategies often implemented by systematic traders. This process resulted in a total of nine distinct feature families, each aiming to capture different structural aspects of commodity price behavior.

In line with standard practices in equity-based machine learning research and inspired by the lookback window parameter (J) in Jegadeesh and Titman (2001), I applied 14 different lookback periods to each of the nine feature types. The lookbacks, expressed in trading days, are 10, 22, 44, 66, 88, 130, 200, 252, 300, 382, 400, 504, 600, and 756. These windows are designed to span ultra-short-term, short-term, medium-term, long-term, and ultra-long-term horizons, allowing the models to learn from trend dynamics and seasonal patterns. This approach yielded a total of 126 distinct daily features per contract.

The full dataset consists of approximately 40 million data points across contracts, dates, and features. For supervised machine learning, I computed daily returns from continuous futures

Exhibit 2. Investment Universe

Group Level 1	Group Level 2	Contract/Exchange Name	Group Level 1	Group Level 2	Contract/Exchange Name
Agriculturals	grains_oilseeds	Canola (ICE Canada)	Energy_fut	gas	Natural Gas (NYMEX – CME Group)
Agriculturals	grains_oilseeds	Corn (CBOT – CME Group)	Energy_fut	gas	Propane – Mt Belvieu LDH (NYMEX – CME Group)
Agriculturals	grains_oilseeds	Crude Palm Oil (Bursa Malaysia Derivatives – BMD)	Energy_fut	oil	Brent Crude Oil (ICE Europe)
Agriculturals	grains_oilseeds	Milling Wheat (Euronext – MATIF)	Energy_fut	oil	Ethanol (CBOT – CME Group)
Agriculturals	grains_oilseeds	Rapeseed (Euronext)	Energy_fut	oil	Low Sulfur Gasoil (ICE Europe)
Agriculturals	grains_oilseeds	Rough Rice (CBOT – CME Group)	Energy_fut	oil	NY Harbor ULSD – Ultra Low Sulfur Diesel (NYMEX – CME Group)
Agriculturals	grains_oilseeds	Soybean Meal (CBOT – CME Group)	Energy_fut	oil	RBOB Gasoline (NYMEX – CME Group)
Agriculturals	grains_oilseeds	Soybean Oil (CBOT – CME Group)	Energy_fut	oil	WTI Crude Oil (NYMEX – CME Group)
Agriculturals	grains_oilseeds	Soybeans (CBOT – CME Group)	Metals	base_metals	Copper (COMEX – CME Group)
Agriculturals	grains_oilseeds	Wheat – Hard Red Winter (KCBT – CME Group)	Metals	base_metals	Iron Ore 62% Fe CFR China (SGX)
Agriculturals	grains_oilseeds	Wheat – Red Spring (MGEX – CME Group)	Metals	base_metals	Lead (LME)
Agriculturals	grains_oilseeds	Wheat – Soft Red Winter (CBOT – CME Group)	Metals	base_metals	Nickel (LME)

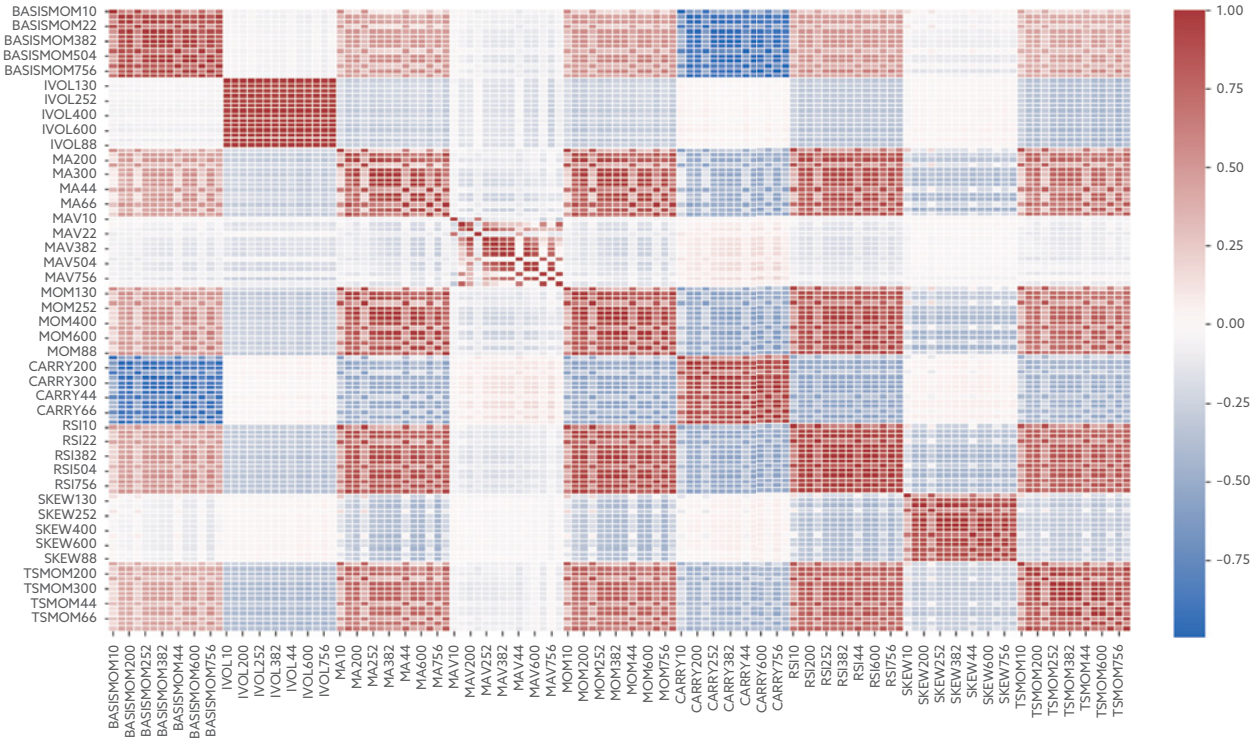
Group Level 1	Group Level 2	Contract/Exchange Name
Agriculturals	livestock	Feeder Cattle (CME)
Agriculturals	livestock	Lean Hogs (CME)
Agriculturals	livestock	Live Cattle (CME)
Agriculturals	softs	Cocoa (ICE US)
Agriculturals	softs	Coffee Arabica (ICE US)
Agriculturals	softs	Coffee Robusta (ICE Europe)
Agriculturals	softs	Cotton No.2 (ICE US)
Agriculturals	softs	Frozen Concentrated Orange Juice (ICE US)
Agriculturals	softs	Rubber (OSE – Osaka Exchange)
Agriculturals	softs	Sugar #11 – Raw Sugar (ICE US)
Agriculturals	softs	Sugar #5 – White Sugar (ICE Europe)
Metals	base_metals	Primary Aluminum (LME)
Metals	base_metals	Zinc (LME)
Metals	precious_metals	Gold 100 oz (COMEX – CME Group)
Metals	precious_metals	Palladium (NYMEX – CME Group)
Metals	precious_metals	Platinum (NYMEX – CME Group)
Metals	precious_metals	Silver (COMEX – CME Group)

Source: Bloomberg LLC.

price series over various prediction horizons: 5, 10, 22, 44, 66, 80, 100, and 200 days. Each prediction horizon corresponds to a separate supervised learning model, enabling the analysis of time-varying predictability across horizons. All features and target variables are normalized into percentiles to ensure comparability across commodities and over time, mitigating scale differences and enhancing model stability during training. **Exhibit 3** shows the correlation heatmap of percentile-transformed features. It reveals that several feature families, such as BASISMOM, CARRY, and SKEW, are largely uncorrelated with each other, indicating they capture distinct dimensions of return predictability. Within individual families, the impact of lookback parameters varies: IVOL features are highly correlated across different horizons, suggesting that the choice of lookback has limited influence on their information content, while such features as MOM, TSMOM, and RSI show greater variation with horizon length, highlighting their sensitivity to the lookback parameter.

Because the scope of this chapter is not to compare different supervised models but rather to focus on the application of ML to commodities, I selected a single type of ML model—a regression boosted tree model—to train over all the various prediction horizons. Boosted trees are a powerful and regularized implementation of gradient boosting, widely used in cross-sectional finance (see Guida and Coqueret 2019). Instead of fitting one large tree, the algorithm builds an ensemble of shallow trees, where each new tree focuses on correcting the residuals of the

Exhibit 3. Correlation Heatmap of Percentile-Transformed Features over the Full Sample



Source: Bloomberg LLC.

previous ones. This sequential learning process enables the model to capture complex, nonlinear interactions between predictors and the target variable. At each iteration, boosted trees minimize a regularized objective function that combines a differentiable loss, such as squared error or logistic loss, with penalties on model complexity, including the number of leaves and the magnitude of leaf weights. The algorithm leverages both the gradient and the Hessian of the loss function to construct an accurate approximation of the optimal tree structure. Split quality is evaluated using a second-order Taylor expansion, which guides the model toward partitions that effectively reduce the loss while maintaining parsimony. These technical refinements make boosted trees especially effective in financial settings, where predictive signals are often subtle and embedded in noisy data.

Each boosted tree model is trained using an expanding window approach, beginning with a minimum of three years of historical data. For the initial training window of five years, 90% of the observations are used for training and the remaining 10%, corresponding to approximately six months, are reserved for out-of-sample testing. In subsequent iterations, the test set is kept fixed to ensure that the training data do not become too temporally distant from the prediction period. Models are retrained annually by extending the training window to incorporate the most recent data, while maintaining a fixed-length, non-overlapping six-month test set. To avoid look-ahead bias, particularly in models that predict long-term returns (such as the 200-day forward horizon), a temporal buffer is introduced. Specifically, the end date of the test features is shifted to ensure that the return label window does not extend into the subsequent prediction period. For example, if the next prediction period begins in January 2023, the test features for a 200-day horizon model would end in January 2022. This adjustment isolates the training and testing phases and prevents any leakage of future information.

Because some hyperparameters introduce randomness, each model is run 20 times with different random seeds for each target variable. The results are then averaged to reduce statistical noise and improve robustness. Consistent with standard practice in financial machine learning, I treat the model's output not as a point forecast but as a ranking signal. Each supervised model produces daily predictions across assets, which I interpret as cross-sectional scores. These scores are then normalized into quantiles and used to sort assets into a long–short portfolio, in a manner closely aligned with traditional factor-based investing. The same score drives both portfolio construction and weighting: Assets with the highest predicted returns are assigned to the long leg, and those with the lowest scores populate the short leg. Within each leg, position sizes are proportional to the scores themselves, giving more weight to the most confident predictions. This setup ensures that the model's relative convictions are fully reflected in the portfolio. I fix the gross exposure of each leg at 1, resulting in a total gross exposure of 2. The process is repeated independently for each prediction horizon, enabling horizon-specific portfolio views.

Model hyperparameters are tuned separately for each target horizon and rebalancing point via a light grid search (for each hyperparameter, I selected three variants), choosing the best parameter set that minimizes the mean squared error. The grid search explores tree depth (from 1 to 3), learning rate (from 0.05 to 0.005), number of trees (from 50 to 150), feature subsampling per tree (from 0.55 to 0.85), and feature subsampling per level (from 0.55 to 0.85), balancing model complexity with out-of-sample robustness. The L2 regularization parameter is held at its default value of 1.

Results: Interpreting Feature Importance across Horizons and Models

In linear models, coefficients represent the marginal contribution of each predictor to the target variable. Although tree-based models, such as gradient boosting, do not yield coefficients in the classical sense, they do provide feature importance scores that reflect the relative contribution of each feature to the model's predictions. These scores can be derived using various metrics, such as the following:

- *Gain:* the improvement in the model's loss function brought by a feature when it is used to split the data
- *Cover:* the number of samples affected by splits on that feature
- *Frequency:* how often a feature is used in splits across all trees

Alternatively, SHAP (SHapley Additive exPlanations) values (Shapley 1953) offer a game-theoretic approach to feature attribution. SHAP values assign each feature a contribution value for individual predictions, enabling both local (per-sample) and global (aggregated) interpretability. Unlike gain or frequency, SHAP values are consistent and provide additive explanations across features. As Simonian, Wu, Itano, and Narayanam (2019) proposed, feature importance can be interpreted as pseudo-betas, offering a directionless yet informative measure of marginal predictive power. This concept plays a central role in interpreting machine learning models, often (wrongly) seen as opaque, and offers insights into which signals the model relies on and how this reliance evolves through time. Understanding which characteristics influence model predictions directly relates to the interpretability challenges discussed in empirical asset pricing applications using machine learning (Gu, Kelly, and Xiu 2020).

In **Exhibit 4**, I report the top 10 normalized feature importance values for each model based on different target horizons, using a last-10-year window for clarity's sake. Each table is sorted by the cumulative importance of each feature across the last 10 years. On average, the top 10 features account for 20% of total importance, while the top 50 features explain roughly 66%. Because of regularization constraints, on average about 40 of the 126 input features are consistently unused by the models.

The models are trained in an expanding window framework, meaning that feature importance in a given year reflects cumulative learning up to that point. For instance, importance values observed for end-of-year 2020 models include data from January 1994 to May 2020, yet meaningful shifts occur in model behavior following the onset of the COVID-19 crisis. This effect is especially pronounced in short-horizon models, such as the 5-day and 10-day variants. As more postcrisis data enter the training set, these models adjust rapidly. In contrast, longer-horizon models, such as the 200-day version, exhibit greater stability, with little change observed between 2019 and 2020. This behavior reflects the model's regularization bias toward persistent signals and its aversion to unstable feature dynamics.

One of the most consistent findings across models and timeframes is the dominance of momentum-based signals, particularly time-series momentum. TSMOM252 appears in the top 10 features for every model, and TSMOM300 is ranked similarly in six out of eight models. Although validating the factor investing literature is not the primary objective of this chapter, the results, derived from a purely data-driven boosted tree model, confirm the predictive power

Exhibit 4. Yearly Normalized Features' Importance

Feature	Target_Variable	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
MOM252	*5d*	2.3%	2.3%	5.8%	4.8%	2.5%	9.3%	2.3%	6.0%	3.4%	7.0%	2.3%
SKEW130	*5d*	1.5%	1.5%	2.3%	2.1%	1.6%	4.4%	1.7%	6.2%	3.8%	7.9%	2.4%
TSMOM252	*5d*	1.9%	1.9%	3.8%	3.1%	1.8%	4.9%	1.9%	4.1%	2.5%	3.7%	1.6%
BASISMOM130	*5d*	1.5%	1.6%	2.5%	3.4%	1.8%	4.0%	2.0%	3.9%	2.4%	5.0%	1.7%
MOM88	*5d*	2.4%	2.0%	5.2%	1.8%	2.0%	4.5%	1.9%	3.7%	1.9%	2.4%	1.7%
SKEW200	*5d*	1.5%	1.5%	2.3%	2.1%	1.4%	3.9%	1.6%	4.4%	2.1%	5.3%	2.0%
TSMOM300	*5d*	1.6%	1.4%	4.2%	2.2%	1.5%	4.1%	1.6%	2.9%	1.6%	3.3%	1.4%
SKEW400	*5d*	1.4%	1.4%	2.3%	1.5%	1.2%	1.9%	1.1%	3.3%	2.4%	4.6%	1.5%
CARRY66	*5d*	1.3%	1.4%	2.4%	2.2%	1.5%	2.5%	1.3%	2.1%	2.0%	3.3%	1.3%
MOM756	*5d*	1.2%	1.3%	1.9%	1.9%	1.3%	2.9%	1.3%	2.5%	1.6%	3.0%	1.3%

Feature	Target_Variable	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
MOM252	*10d*	2.4%	2.4%	6.7%	5.4%	3.1%	3.3%	2.2%	4.0%	2.7%	5.5%	1.9%
TSMOM252	*10d*	1.9%	1.9%	3.8%	3.8%	3.4%	3.2%	2.3%	4.8%	2.8%	3.3%	1.7%
SKEW130	*10d*	1.2%	1.2%	2.5%	3.1%	1.8%	2.0%	1.6%	5.1%	2.9%	6.4%	2.1%
BASISMOM130	*10d*	1.6%	1.5%	3.6%	4.2%	2.6%	2.3%	2.0%	3.4%	2.4%	3.9%	1.5%
SKEW200	*10d*	1.3%	1.3%	2.0%	2.2%	1.8%	2.0%	1.6%	3.5%	2.7%	6.0%	1.9%
TSMOM300	*10d*	2.1%	2.1%	4.5%	2.9%	2.0%	2.4%	1.7%	2.3%	1.8%	2.4%	1.6%
SKEW382	*10d*	1.4%	1.4%	2.6%	1.9%	1.4%	1.3%	1.1%	2.7%	3.3%	6.6%	1.6%
BASISMOM66	*10d*	1.4%	1.4%	3.1%	2.9%	1.8%	1.8%	1.2%	2.0%	1.8%	3.8%	1.2%
SKEW756	*10d*	1.3%	1.5%	3.1%	2.5%	2.2%	2.1%	1.7%	2.6%	2.3%	0.0%	1.5%
SKEW600	*10d*	1.4%	1.0%	3.1%	2.4%	1.8%	1.5%	1.3%	2.5%	2.5%	2.1%	1.3%

Feature	Target_Variable	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
MOM252	*22d*	2.7%	2.7%	5.5%	5.6%	2.2%	2.9%	2.1%	3.5%	5.4%	6.3%	2.0%
TSMOM252	*22d*	2.4%	2.5%	6.5%	5.5%	2.4%	3.6%	2.5%	4.6%	4.2%	3.4%	2.2%
SKEW130	*22d*	1.6%	1.7%	3.4%	2.9%	1.7%	2.0%	2.0%	5.6%	7.5%	8.8%	2.3%
SKEW200	*22d*	1.6%	1.6%	1.7%	2.4%	1.5%	2.0%	1.8%	4.0%	5.8%	8.2%	2.2%
TSMOM300	*22d*	2.3%	2.8%	4.7%	3.9%	1.6%	2.9%	2.2%	2.0%	3.2%	3.9%	1.4%
SKEW252	*22d*	0.0%	0.9%	0.0%	0.0%	1.6%	1.9%	1.4%	4.7%	7.9%	8.3%	2.0%
BASISMOM130	*22d*	1.3%	1.5%	3.0%	3.1%	1.8%	2.0%	1.9%	2.7%	3.9%	3.6%	1.6%
MOM756	*22d*	1.3%	1.4%	2.7%	2.4%	1.5%	1.8%	1.7%	2.7%	3.9%	4.3%	1.8%
SKEW382	*22d*	1.8%	1.8%	2.2%	2.2%	1.5%	1.9%	1.7%	3.3%	5.1%	0.0%	1.2%
BASISMOM200	*22d*	1.4%	1.3%	0.0%	0.0%	1.4%	1.5%	1.5%	3.4%	4.5%	4.2%	1.4%

Feature	Target_Variable	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
TSMOM252	*44d*	2.6%	2.8%	5.6%	3.5%	3.8%	2.8%	2.9%	3.1%	2.2%	3.6%	2.0%
MOM252	*44d*	2.3%	2.6%	5.8%	3.4%	3.1%	2.5%	1.5%	2.0%	1.8%	5.3%	1.9%
BASISMOM200	*44d*	1.7%	1.5%	3.8%	2.8%	3.0%	2.4%	2.2%	2.9%	2.2%	5.8%	2.0%
SKEW252	*44d*	1.8%	0.9%	0.0%	2.6%	2.8%	1.7%	1.9%	3.4%	3.1%	9.4%	2.7%
BASISMOM130	*44d*	1.4%	1.3%	2.5%	2.2%	2.7%	2.2%	2.4%	2.5%	2.1%	5.5%	1.9%
TSMOM300	*44d*	2.0%	2.0%	5.0%	2.5%	2.5%	1.9%	2.3%	1.9%	1.6%	3.7%	1.3%
BASISMOM88	*44d*	1.6%	1.5%	3.5%	2.4%	3.0%	2.0%	2.3%	2.5%	1.8%	3.3%	1.9%
SKEW130	*44d*	1.3%	1.6%	2.5%	0.0%	2.2%	1.8%	2.0%	2.9%	2.3%	6.1%	2.5%
CARRY88	*44d*	1.7%	1.6%	2.8%	2.4%	3.0%	2.0%	2.2%	2.0%	1.1%	3.2%	1.9%
SKEW756	*44d*	1.6%	1.5%	3.8%	2.0%	2.3%	1.6%	1.7%	2.0%	2.1%	3.2%	1.9%

Feature	Target_Variable	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
TSMOM252	*66d*	2.6%	3.2%	5.5%	3.3%	3.3%	3.0%	2.7%	2.8%	2.3%	4.3%	2.0%
SKEW252	*66d*	1.4%	1.3%	0.0%	1.5%	2.2%	1.4%	1.8%	2.9%	3.8%	10.5%	3.1%
BASISMOM130	*66d*	1.1%	1.7%	2.8%	2.0%	3.0%	2.2%	2.3%	2.6%	2.5%	6.8%	2.2%
BASISMOM200	*66d*	1.6%	1.8%	3.6%	2.5%	2.7%	2.5%	2.1%	2.3%	2.0%	6.4%	1.8%
SKEW300	*66d*	1.3%	1.7%	2.4%	1.5%	1.8%	2.1%	2.0%	2.4%	2.4%	7.3%	1.9%
SKEW130	*66d*	1.2%	1.3%	3.2%	2.0%	1.8%	2.0%	1.9%	2.7%	2.7%	5.3%	2.2%
MOM252	*66d*	2.0%	2.9%	5.0%	2.9%	0.0%	2.0%	1.2%	1.6%	1.6%	4.6%	1.6%
TSMOM300	*66d*	1.9%	2.4%	4.1%	1.6%	1.7%	2.5%	3.0%	1.4%	1.6%	3.2%	1.6%
SKEW600	*66d*	1.6%	1.9%	3.7%	2.1%	2.4%	1.6%	1.9%	2.3%	2.1%	3.5%	1.6%
SKEW400	*66d*	2.1%	2.3%	3.9%	2.0%	1.8%	1.7%	1.8%	2.2%	2.4%	3.1%	1.3%

Feature	Target_Variable	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
TSMOM252	*88d*	3.1%	5.8%	6.5%	3.4%	4.1%	2.9%	2.8%	2.4%	1.8%	3.3%	1.6%
TSMOM200	*88d*	3.1%	5.2%	6.0%	3.4%	4.1%	2.3%	2.3%	2.3%	1.9%	3.8%	1.6%
SKEW130	*88d*	1.7%	2.3%	3.7%	1.9%	2.5%	2.0%	2.1%	3.3%	3.2%	7.9%	2.4%
BASISMOM130	*88d*	1.8%	2.3%	3.2%	2.1%	2.7%	2.0%	2.1%	3.2%	2.9%	5.8%	2.1%
BASISMOM200	*88d*	1.7%	2.5%	3.5%	2.7%	3.3%	2.1%	2.5%	2.6%	1.9%	5.3%	1.7%
SKEW200	*88d*	1.7%	2.2%	2.5%	2.2%	2.7%	2.0%	2.0%	2.1%	1.8%	8.2%	1.8%
SKEW252	*88d*	1.2%	0.0%	0.0%	0.0%	1.8%	1.6%	2.4%	3.1%	3.5%	9.1%	2.6%
IVOL400	*88d*	1.7%	2.1%	2.6%	1.9%	2.2%	1.9%	1.9%	2.0%	2.1%	4.3%	1.8%
TSMOM400	*88d*	2.2%	4.0%	5.7%	3.3%	3.3%	1.1%	0.0%	1.7%	1.5%	0.0%	1.0%
SKEW600	*88d*	2.5%	2.9%	3.9%	1.9%	2.3%	1.4%	1.7%	2.2%	2.3%	0.0%	1.4%

Exhibit 4. Yearly Normalized Features' Importance *(continued)*

Feature	Target_Variable	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
TSMOM252	*100d*	3.3%	4.5%	6.1%	3.4%	7.0%	2.5%	2.3%	1.9%	1.7%	3.2%	1.5%
TSMOM200	*100d*	3.2%	4.3%	5.6%	3.3%	5.9%	2.4%	2.2%	2.1%	1.8%	3.4%	1.8%
SKEW130	*100d*	2.0%	2.0%	3.2%	2.2%	3.7%	2.1%	2.2%	3.4%	3.2%	8.4%	2.7%
BASISMOM130	*100d*	2.1%	1.9%	3.3%	2.2%	4.6%	2.1%	2.4%	3.3%	2.6%	5.6%	2.3%
BASISMOM200	*100d*	1.9%	1.9%	3.1%	2.9%	5.2%	2.3%	2.4%	2.4%	1.9%	3.3%	1.9%
SKEW200	*100d*	1.7%	1.7%	2.2%	2.2%	3.7%	2.0%	2.0%	1.9%	1.8%	7.6%	2.0%
TSMOM400	*100d*	2.6%	2.8%	5.5%	3.6%	5.3%	1.0%	0.0%	1.2%	1.5%	2.4%	1.6%
IVOL400	*100d*	1.8%	2.0%	2.6%	1.9%	3.6%	1.7%	1.7%	1.9%	2.1%	3.9%	1.9%
SKEW600	*100d*	2.4%	2.9%	3.8%	2.1%	3.2%	1.6%	1.8%	2.2%	2.1%	0.0%	1.5%
SKEW400	*100d*	2.9%	2.4%	4.0%	1.9%	3.4%	1.8%	1.8%	2.2%	1.6%	0.0%	1.5%

Feature	Target_Variable	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
TSMOM200	*200d*	3.2%	6.2%	6.1%	6.9%	6.5%	3.0%	2.9%	2.6%	2.4%	6.3%	2.3%
TSMOM252	*200d*	2.5%	7.9%	6.6%	7.2%	5.4%	2.4%	2.6%	1.8%	1.6%	5.3%	1.9%
IVOL300	*200d*	1.6%	3.5%	3.7%	4.8%	3.6%	2.0%	2.1%	1.4%	1.2%	5.3%	1.7%
IVOL382	*200d*	1.6%	2.8%	3.6%	4.1%	3.9%	2.0%	2.2%	2.0%	1.6%	3.7%	1.5%
TSMOM300	*200d*	3.1%	7.0%	5.6%	5.6%	1.9%	1.3%	0.0%	1.4%	1.4%	0.0%	1.5%
SKEW600	*200d*	1.9%	3.3%	3.2%	3.3%	2.9%	1.6%	1.3%	1.4%	1.1%	4.1%	1.8%
IVOL252	*200d*	1.9%	3.9%	3.8%	3.9%	3.6%	1.7%	1.6%	1.3%	1.4%	0.0%	1.6%
MOM600	*200d*	1.5%	0.0%	3.0%	2.8%	2.7%	2.1%	1.9%	1.8%	1.8%	4.4%	2.2%
MA756	*200d*	1.6%	2.1%	2.1%	2.3%	2.2%	1.8%	1.5%	1.4%	1.8%	4.7%	2.2%
TSMOM382	*200d*	2.1%	4.5%	5.6%	5.6%	0.0%	1.2%	1.1%	1.1%	1.1%	0.0%	1.2%

Feature	Target_Variable	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
TSMOM252	*ens*	2.3%	3.4%	4.9%	3.7%	3.5%	2.8%	2.2%	2.8%	2.1%	3.3%	1.6%
SKEW130	*ens*	1.3%	1.3%	2.3%	1.6%	1.7%	2.0%	1.7%	3.4%	3.0%	6.1%	2.0%
BASISMOM130	*ens*	1.3%	1.6%	2.6%	2.5%	2.5%	2.1%	1.9%	2.6%	2.2%	4.0%	1.6%
MOM252	*ens*	1.7%	1.4%	3.5%	3.0%	1.2%	2.7%	1.2%	1.9%	1.7%	3.5%	1.3%
SKEW200	*ens*	1.5%	1.6%	1.8%	1.7%	1.8%	1.9%	1.5%	2.4%	2.2%	4.4%	1.6%
BASISMOM200	*ens*	1.3%	1.3%	2.1%	1.6%	2.4%	1.6%	1.6%	2.3%	2.0%	4.0%	1.5%
TSMOM300	*ens*	2.0%	2.3%	3.1%	2.3%	1.4%	2.0%	1.6%	1.6%	1.5%	2.5%	1.3%
SKEW600	*ens*	1.6%	1.8%	2.9%	1.9%	1.8%	1.6%	1.3%	1.7%	1.8%	1.3%	1.4%
TSMOM200	*ens*	1.3%	2.1%	2.4%	1.9%	2.4%	1.6%	1.5%	1.2%	1.1%	2.3%	1.2%
SKEW252	*ens*	0.7%	0.5%	0.0%	0.6%	1.3%	1.2%	1.5%	2.7%	3.1%	5.5%	1.9%

Source: Bloomberg LLC.

of trend signals in commodity markets. Cross-sectional momentum (MOM252) also appears frequently among the most important features, suggesting that both absolute and relative price dynamics are integral to model performance.

Another notable result is the prominence of skewness-based features, particularly in short-term models. For example, SKEW130 and SKEW200 display elevated importance in 2022, especially in models with shorter forecast horizons. This finding is consistent with the hypothesis that negative skewness can trigger short-term price overreactions, followed by reversals (Fernandez-Perez et al. 2018). In this sense, short-term models tend to load more on reversal-type signals, often combining skewness with momentum indicators, whereas longer-horizon models gravitate more heavily toward pure trend-following signals, such as TSMOM.

Finally, it is worth noting that classical technical indicators, such as RSI, MA, and MAV, rank lower in importance for almost all horizons. Regardless of lookback period, these features appear less frequently in the top 10, suggesting that when competing against more academically grounded signals, their marginal contribution to predictive accuracy is limited.

Performance Analysis Comparing Models and Ensemble

Exhibit 5 presents the detailed statistics of long–short portfolios constructed for each target horizon. Each leg of the long–short portfolio is weighted according to its normalized prediction score. Given the narrow nature of the commodity investment universe, limited to only

Exhibit 5. Performance and Analytics Table for Models and Ensemble

Model	Annualized Return Net of Transaction Costs	Annualized Volatility	Sharpe	Sortino	Downside Volatility	Value at Risk (95%)	Success Rate	Max. Drawdown	Daily Turnover
5d	49%	13.5%	3.6	5.3	9.2%	1.2%	60%	−17%	38%
10d	39%	13.5%	2.9	4.2	9.1%	1.2%	58%	−24%	37%
22d	26%	13.5%	1.9	2.7	9.7%	1.3%	56%	−28%	31%
44d	20%	13.8%	1.5	2.1	9.8%	1.3%	55%	−26%	29%
66d	18%	13.9%	1.3	1.8	9.9%	1.3%	55%	−34%	27%
88d	18%	13.7%	1.3	1.9	9.6%	1.3%	54%	−25%	28%
100d	20%	13.7%	1.4	2.0	9.8%	1.3%	55%	−25%	27%
200d	22%	13.1%	1.7	2.4	9.1%	1.2%	55%	−23%	27%
Ensemble	27%	10.9%	2.4	3.3	8.3%	1.0%	57%	−22%	17%

Source: Bloomberg LLC.

41 contracts, it would not be meaningful to form decile portfolios. Instead, the long portfolio includes contracts with normalized predictions above the cross-sectional median, and the short portfolio contains those below it. As explained previously, all models are retrained each year in December and new predictions are made every day for the following year that becomes the out-of-sample results. Hence, the models' portfolio would change daily according to new unseen features' input. I applied a generic prediction-level weighting scheme (comparable to a factor intensity) to accentuate the role of predictions in the performance of the models' long–short portfolio.

Results are shown for the full sample of out-of-sample predictions from January 2000 until June 2024, net of transaction costs, which are computed, for the sake of simplicity, as 5 bps per unit of turnover. The ensemble is created using the average of normalized predictions, which are normalized once more to filter and weight the assets in the same fashion as the other target horizon models. The gross market value (GMV) is kept at a constant level for each model because a new model weight is recomputed each day with new predictions between two yearly retrainings.

Annualized net performance by target horizon exhibits a nonmonotonic pattern: Performance is higher for shorter-term horizons—it decreases from 5 to 66 days—and it rises slightly after 88 days. Sharpe and Sortino ratios are generally high, especially for shorter-term models (Sharpe ratio of 3.6 and 2.9 for, respectively, 5d and 10d). The cost, however, is a high level of turnover. Even with conservative transaction costs of 5 bps, the capacity (maximum assets under management that is tradeable without impacting the market too much and leaving too visible a market footprint) is in the low hundreds of millions, especially for achieving the

required turnover on such contracts as orange juice, rubber, or rice. The success rate, defined as the number of positive returns over the total number of daily returns, is generally very good for all models and very high for shorter-term horizon strategies (60.0% for 5d, 58.3% for 10d).

The risk measures do not indicate large differences among the models, with volatility being around 13% and downside volatility being below 10%. There are some differences in maximum drawdown over the full period, part of which occurred during the COVID-19 pandemic; maximum drawdown is high in general and in particular for the 22d and 66d models. For these models, maximum drawdown exceeds the rule of thumb of 1.5 times the volatility of the strategy, which is commonly used as a "red flag" in multistrategy hedge funds pods. Regarding the ensemble, the results show that blending predictions together provides a risk diversification effect that smooths volatility and tail risk. While providing a mid-range net annualized return of 27.3%, the ensemble cuts annual volatility by 2 to 3 points relative to the best standalone models, lowers value at risk, and keeps the drawdowns moderate.

Exhibit 6 shows the full-period pairwise correlation matrix for the strategies, each defined by a specific target horizon, ranging from short term (5d) to long term (200d), including their relationship to an ensemble strategy that aggregates them.

First, the matrix reveals a high degree of correlation among the medium-term models (22d to 100d), with values typically exceeding 0.80. For example, the correlation between the 22d and 44d strategies is 0.84, while the 66d and 88d correlation reaches 0.88. These elevated correlations suggest that the corresponding models are identifying overlapping patterns in commodity returns, arising from important common features. Although this finding confirms the robustness of certain signals, it also implies a degree of informational redundancy among these models, which may limit the incremental benefit when combining them into the ensemble.

Conversely, the 200d strategy stands out because of its markedly lower correlations. For instance, its correlations with the 5d and 22d strategies are only 0.41 and 0.49, respectively. This divergence suggests that the 200d model is extracting fundamentally different signals, as depicted by the top 10 most important features table (Exhibit 4), potentially aligned with slow-moving structural or seasonal phenomena. In an ensemble framework, such low

Exhibit 6. Correlation Matrix for Long–Short Portfolios

Target Variable	5d	10d	22d	44d	66d	88d	100d	200d	Ensemble
5d	1.00	0.83	0.77	0.73	0.71	0.66	0.65	0.41	0.83
10d	0.83	1.00	0.83	0.78	0.76	0.71	0.69	0.50	0.88
22d	0.77	0.83	1.00	0.84	0.81	0.74	0.72	0.49	0.89
44d	0.73	0.78	0.84	1.00	0.86	0.79	0.77	0.51	0.91
66d	0.71	0.76	0.81	0.86	1.00	0.88	0.84	0.58	0.93
88d	0.66	0.71	0.74	0.79	0.88	1.00	0.92	0.63	0.91
100d	0.65	0.69	0.72	0.77	0.84	0.92	1.00	0.64	0.90
200d	0.41	0.50	0.49	0.51	0.58	0.63	0.64	1.00	0.68
Ensemble	0.83	0.88	0.89	0.91	0.93	0.91	0.90	0.68	1.00

Source: Bloomberg LLC.

correlation is valuable because it introduces more "orthogonal" information that can reduce portfolio variance and enhance risk-adjusted returns, even if the 200d model has a lower standalone Sharpe ratio.

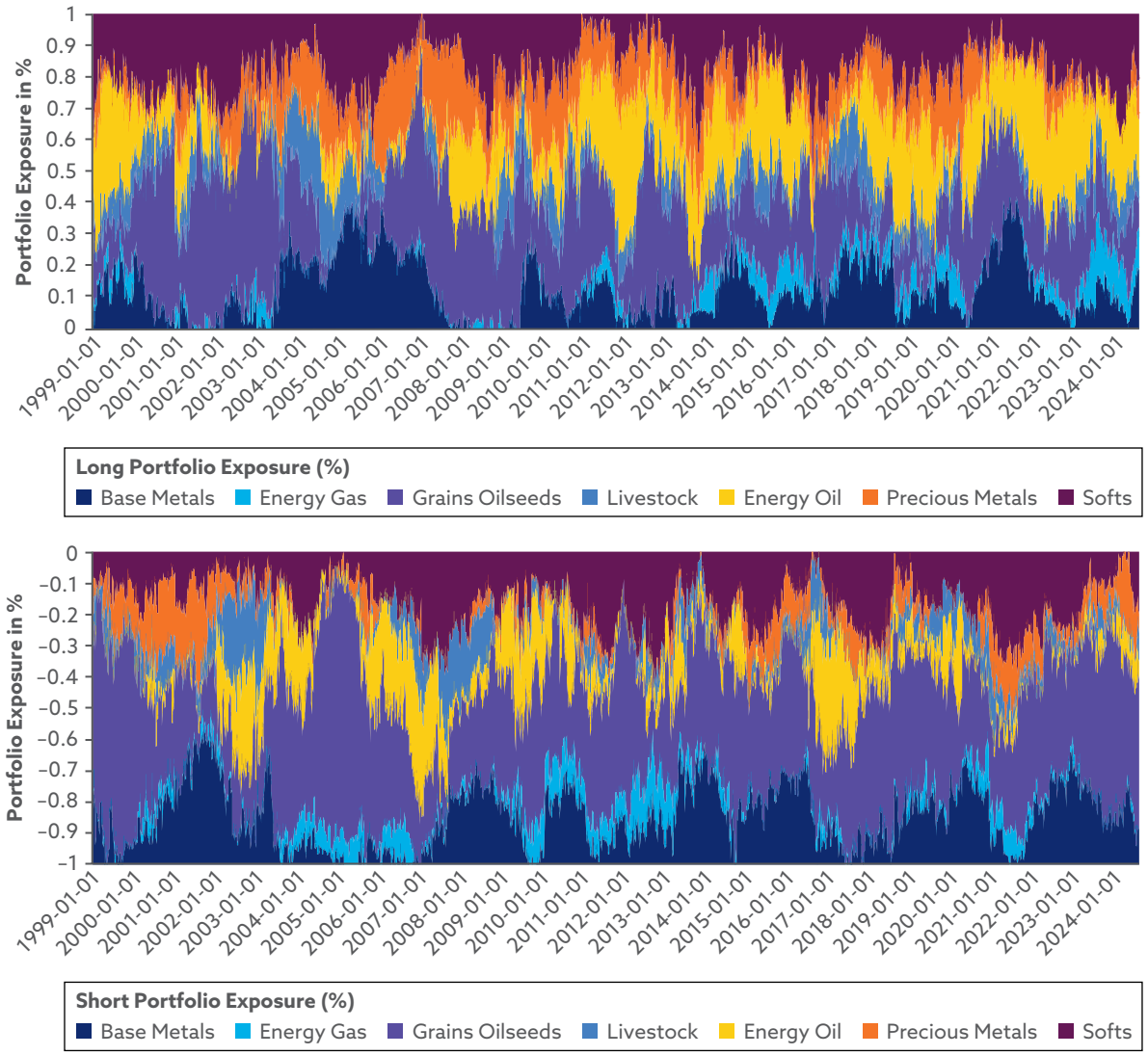
The ensemble model itself exhibits very high correlations with the base strategies, particularly with the mid-term ones, reaching 0.928 with the 66d model and 0.915 with the 88d model. This finding indicates that the ensemble is primarily shaped by these horizons, which may dominate because of superior performance or greater predictive stability. Nevertheless, the ensemble's correlation with the 200d model remains moderate at 0.684, showing that it still benefits from some degree of diversification. The inclusion of all horizons in the ensemble, even those with weaker performance or less commonality, supports the principle of model averaging, where aggregating diverse signals leads to smoother and more robust predictive outputs.

Exhibit 7 shows the long and short book group breakdown of the ensemble portfolio. I classified contracts into subsectors (base metals, precious metals, energy oil, energy gas, softs, grains and oilseeds, livestock) and show the exposure of the long leg and the short leg. Both legs are highly dynamic and invest in all subsectors but with substantial shift in time-varying exposure from net long to net short. When looking at each portfolio separately and focusing on the long leg, grains and oilseeds and precious metals consistently represent a significant portion of the long exposure across the entire sample period. Soft commodities and base metals exhibit elevated exposure during specific intervals, notably between 1999 and 2003, 2009 and 2012, and 2018 and 2021. Crude oil–related positions show pronounced spikes in exposure that coincide with major macroeconomic and geopolitical disruptions, such as those observed in 2008, 2014–2016, and 2022. Natural gas begins to play a more prominent role from 2010 onward, potentially reflecting increased market volatility. Livestock exposures display a more erratic pattern but maintain a persistent presence, with notable peaks occurring in the early 2000s and again in the post-2020 period.

Short exposures are predominantly concentrated in grains and oilseeds, particularly from 1998 to 2011 and after 2022. Natural gas exhibited substantial short allocations between 2006 and 2009 and again from 2019 to 2022, suggesting persistent bearish signals or mean-reverting behavior. Crude oil positions were heavily shorted during 2005–2008 and 2011–2015, coinciding with episodes of elevated volatility and shifting macro conditions. Precious metals and livestock also contribute meaningfully to the short book, with consistent allocations observed throughout 2004–2012. In contrast, base metals and soft commodities exhibit limited or intermittent short exposure, potentially reflecting a structural long bias or weaker signal strength in these sectors.

Finally, **Exhibit 8** shows the net exposure by group. As hinted by the long and short exposure, net market value (NMV) highlights significant temporal variation in directional positions across sectors. Grains and oilseeds show the most pronounced fluctuations, with large positive swings in the early 2000s, 2007–2008, and 2021 and deep negative exposures in 1999–2000, 2005, 2014, and after 2023. Energy oil displays sustained positive net exposure in multiple periods, notably in 2003–2006 and 2012–2015 and from 2020 through 2022. These periods coincide with major energy market cycles, indicating the model's persistent tilt toward oil-related trends during macro-driven dislocations. Livestock maintains consistent, though smaller, net long positions through most of the sample, with occasional reversals during 2008–2010

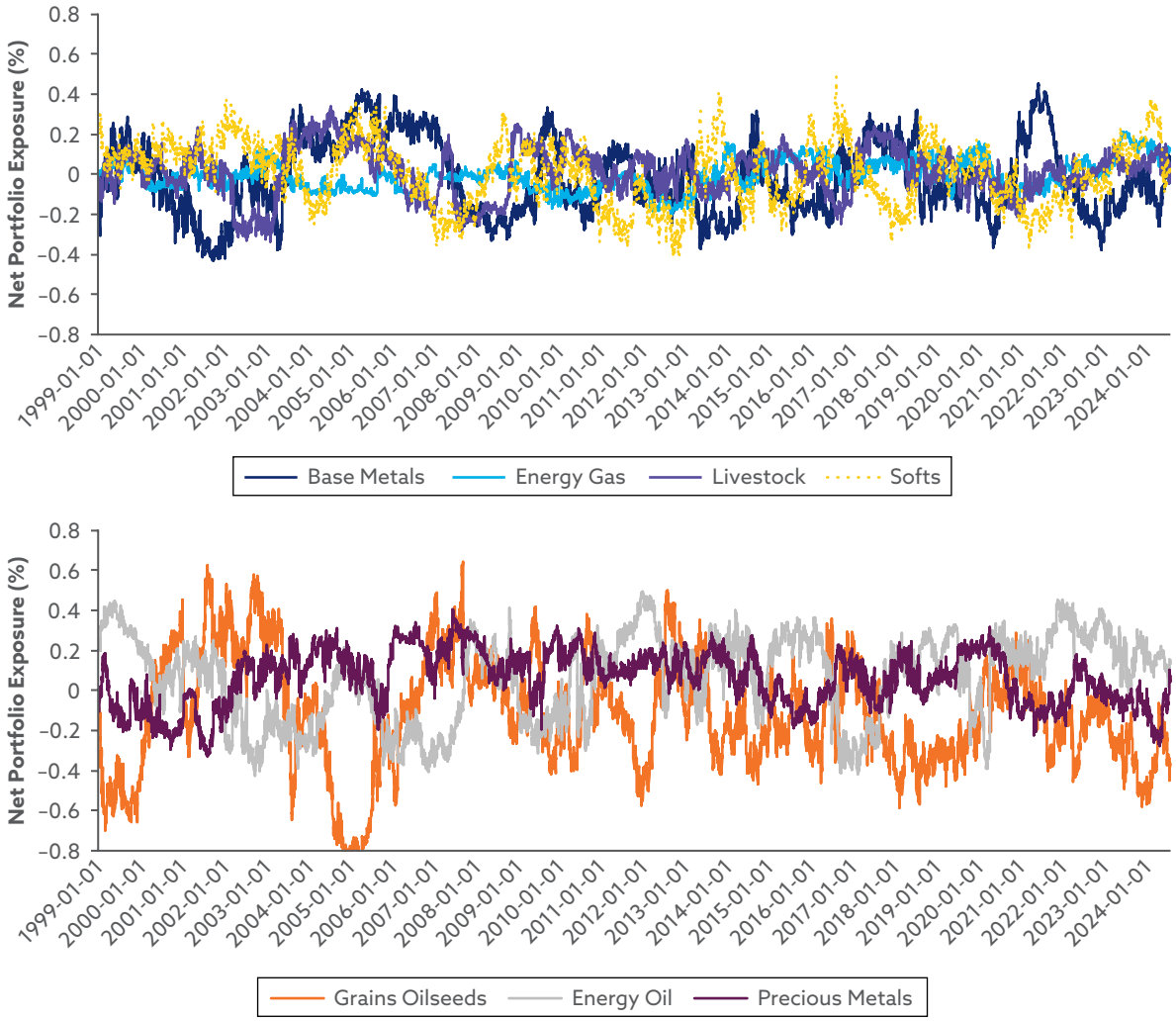
Exhibit 7. Gross Exposure (GMV) Breakdown for Ensemble Long and Short Book



Source: Bloomberg LLC.

and after 2015. Base metals and soft commodities demonstrate alternating positive and negative net exposures, with base metals notably net long during the post–Global Financial Crisis commodity supercycle (2009–2011) and again after the COVID-19 crisis with inflation rising, while softs show long tilts around 2003, 2017–2018, and 2021. Precious metals exhibit notable net short phases in 1999–2000, 2005–2006, and 2013–2015 and after 2022, interspersed with long phases, such as 2008–2009 and the early COVID-19 crisis. These shifts likely reflect both macro-hedging behavior and relative value signals.

Exhibit 8. Net Exposure (NMV) per Contract Type for Ensemble



Source: Bloomberg LLC.

Conclusion

This chapter began with a simple question: Can the rigor and structure of equity-based machine learning pipelines be transposed onto commodity futures markets, where economic intuition runs deep but modeling applications have remained largely siloed and segmented? The results of this study suggest the answer is a clear yes, although not without important structural adaptations relative to equities.

Commodity futures are not stocks. They lack balance sheets, are governed by supply chains rather than corporate disclosures, and move to the beat of exogenous cycles, weather, geopolitics, inventories, and institutional positioning. These characteristics, often framed as modeling

challenges, are in fact the very properties that make commodity markets an ideal testbed for modern supervised learning techniques. When organized through a cross-sectional lens, where predictors are derived from asset pricing theory and not just price patterns, commodity futures reveal a persistent and learnable structure that can be used for long–short portfolio construction.

My approach was built on three foundational choices. First, I rooted the features in academic theory, not technical folklore. Carry, basis momentum, skewness, time-series momentum, and so on are not ad hoc constructs but manifestations of storage constraints, hedging pressure, and market segmentation. Second, I extended the empirical asset pricing logic of equity factor models into the commodity space by constructing cross-sectional long–short portfolios guided by machine-learned scores. Finally, I introduced temporal diversity by predicting across multiple horizons, which is a realistic modern way of creating tradable portfolios, capturing both short-term reversals and long-term trend cycles by fusing them via ensemble averaging.

Across a 30-year window and 41 futures contracts, I showed that ML can not only forecast cross-sectional returns but also rank assets in a way that produces consistent economic value. The empirical results are instructive. Time-series momentum, the cornerstone of trend-following CTAs, dominates most models, particularly over intermediate to long horizons. Skewness, a higher-moment feature often ignored in equity ML, plays a meaningful role at short horizons, hinting at reversal tendencies and potential sentiment-driven mispricing. Importantly, the ensemble strategy does more than just blend signals; it reconciles the tension between short-term alpha and long-term robustness. Unlike individual models, which exhibit a mirror J-shaped performance curve across horizons, the ensemble smooths these extremes, offering better drawdown control and volatility reduction without materially sacrificing returns. This finding echoes a broader principle in financial ML: Diversity of signal horizons matters just as much as signal quality.

The correlation structure among horizons further reinforces the value of multiscale modeling. Mid-term models (22d to 100d) cluster tightly, suggesting that they capture variations on a shared theme, likely driven by overlapping features. But the 200-day model, with its low correlations and distinct top features, adds a unique longer-cycle perspective. This decorrelation, although dilutive on a standalone basis, enhances the ensemble's stability and reduces fragility to negative events, as depicted by the better risk profile of the ensemble.

At the portfolio level, the results reveal meaningful heterogeneity. Exposure patterns vary strongly across sectors and time, with pronounced tilts toward grains, energy, and metals at key macro inflection points. These shifts are not noise; they reflect the interaction between predictive signals and regime-specific fundamentals. For instance, long tilts in crude oil during geopolitical crises or short positions in grains after 2022 mirror real-world economic dynamics that models have learned to anticipate.

In conclusion, this chapter bridges the methodological divide between equity ML and commodity return forecasting. It demonstrates that with careful feature construction, robust out-of-sample validation, and thoughtful aggregation across time horizons, ML can serve not only as a modeling tool but as a research framework—one that makes commodities more interpretable, predictable, and, ultimately, investable. As systematic investing continues to evolve, the fusion of domain theory with modern ML holds the key to unlocking latent structure in every markets.

References

Ampountolas, Apostolos, and Sayed AlGharbi. 2025. "An Innovative Hybrid LightGBM-BPNN Model for Enhanced Commodity Forecasting Accuracy." *Finance Research Open* 1 (1). doi:10.1016/j.finr.2025.100004.

Anderson, Bing, and Shuyun May Li. 2015. "An Investigation of the Relative Strength Index." *Banks & Bank Systems* 10 (1): 92–96.

Angelidis, Timotheos, Athanasios Sakkas, and Nikalaos Tessaromatis. 2025. "Predicting Commodity Returns: Time Series vs. Cross Sectional Prediction Models." *Journal of Commodity Markets* 38 (June). doi:10.1016/j.jcomm.2025.100475.

Asness, Clifford S., Tobias J. Moskowitz, and Lasse Heje Pedersen. 2013. "Value and Momentum Everywhere." *Journal of Finance* 68 (3): 929–85. doi:10.1111/jofi.12021.

Basu, Devraj, and Joëlle Miffre. 2013. "Capturing the Risk Premium of Commodity Futures: The Role of Hedging Pressure." *Journal of Banking & Finance* 37 (7): 2652–64. doi:10.1016/j.jbankfin.2013.02.031.

Ben Jabeur, Sami, Rabeh Khalfaoui, and Wissal Ben Arfi. 2021. "The Effect of Green Energy, Global Environmental Indexes, and Stock Markets in Predicting Oil Price Crashes: Evidence from Explainable Machine Learning." *Journal of Environmental Management* 298 (November). doi:10.1016/j.jenvman.2021.113511.

Bhardwaj, Geetesh, Gary Gorton, and Geert Rouwenhorst. 2015. "Facts and Fantasies About Commodity Futures Ten Years Later." NBER Working Paper 21243 (June). doi:10.3386/w21243.

Blitz, David, Matthias X. Hanauer, Tobias Hoogteijling, and Clint Howard. 2023. "The Term Structure of Machine Learning Alpha." Working paper (19 July). doi:10.2139/ssrn.4474637.

Boons, Martijn, and Melissa Porras Prado. 2019. "Basis-Momentum." *Journal of Finance* 74 (1): 239–79. doi:10.1111/jofi.12738.

Cheng, Ing-Haw, Andrei Kirilenko, and Wei Xiong. 2015. "Convective Risk Flows in Commodity Futures Markets." *Review of Finance* 19 (5): 1733–81. doi:10.1093/rof/rfu043.

Costa, Alexandre Bonnet R., Pedro Calvacanti G. Ferreira, Wagner P. Gaglianone, Osmani Teixeira C. Guillén, João Victor Issler, and Yihao Lin. 2021. "Machine Learning and Oil Price Point and Density Forecasting." *Energy Economics* 102 (October). doi:10.1016/j.eneco.2021.105494.

de Roon, Frans A., Theo E. Nijman, and Chris Veld. 2000. "Hedging Pressure Effects in Futures Markets." *Journal of Finance* 55 (3): 1437–56. doi:10.1111/0022-1082.00253.

Fama, Eugene F., and Kenneth R. French. 2015. "Commodity Futures Prices: Some Evidence on Forecast Power, Premiums, and the Theory of Storage." In *The World Scientific Handbook of Futures Markets*, edited by Anastasio G. Malliaris and William T. Ziemba, 79–102. London: World Scientific.

Fernandez-Perez, Adrian, Bart Frijns, Ana-Maria Fuertes, and Joëlle Miffre. 2018. "The Skewness of Commodity Futures Returns." *Journal of Banking & Finance* 86 (January): 143–58. doi:10.1016/j.jbankfin.2017.06.015.

Foroutan, Parisa, and Salim Lahmiri. 2024. "Deep Learning Systems for Forecasting the Prices of Crude Oil and Precious Metals." *Financial Innovation* 10 (16 July). doi:10.1186/s40854-024-00637-z.

Fuertes, Ana-Maria, Joëlle Miffre, and Adrian Fernandez-Perez. 2015. "Commodity Strategies Based on Momentum, Term Structure, and Idiosyncratic Volatility." *Journal of Futures Markets* 35 (3): 274–97. doi:10.1002/fut.21656.

Gargano, Antonio, and Allan Timmermann. 2014. "Forecasting Commodity Price Indexes Using Macroeconomic and Financial Predictors." *International Journal of Forecasting* 30 (3): 825–43. doi:10.1016/j.ijforecast.2013.09.003.

Gorton, Gary B., Fumio Hayashi, and K. Geert Rouwenhorst. 2013. "The Fundamentals of Commodity Futures Returns." *Review of Finance* 17 (1): 35–105. doi:10.1093/rof/rfs019.

Gorton, Gary, and K. Geert Rouwenhorst. 2006. "Facts and Fantasies About Commodity Futures." *Financial Analysts Journal* 62 (2): 47–68. doi:10.2469/faj.v62.n2.4083.

Gu, Shihao, Bryan Kelly, and Dacheng Xiu. 2020. "Empirical Asset Pricing via Machine Learning." *Review of Financial Studies* 33 (5): 2223–73. doi:10.1093/rfs/hhaa009.

Guida, Tony, and Guillaume Coqueret. 2019. "Ensemble Learning Applied to Quant Equity: Gradient Boosting in a Multifactor Framework." In *Big Data and Machine Learning in Quantitative Investment*, edited by Tony Guida, 129–148. Hoboken, NJ: Wiley.

Hirshleifer, David. 1988. "Residual Risk, Trading Costs, and Commodity Futures Risk Premia." *Review of Financial Studies* 1 (2): 173–93. doi:10.1093/rfs/1.2.173.

Hong, Harrison, and Motohiro Yogo. 2012. "What Does Futures Market Interest Tell Us About the Macroeconomy and Asset Prices?" *Journal of Financial Economics* 105 (3): 473–90. doi:10.1016/j.jfineco.2012.04.005.

Jegadeesh, Narasimhan, and Sheridan Titman. 2001. "Profitability of Momentum Strategies: An Evaluation of Alternative Explanations." *Journal of Finance* 56 (2): 699–720. doi:10.1111/0022-1082.00342.

Kaldor, Nicholas. 1939. "Speculation and Economic Stability." *Review of Economic Studies* 7 (1): 1–27. doi:10.2307/2967593.

Keynes, John M. 1930. *A Treatise on Money, Volume 1: The Pure Theory of Money*. London: Harcourt, Brace & Co. doi:10.4324/9781003547150-7.

Lucas, Robert E., Jr. 1978. "Asset Prices in an Exchange Economy." *Econometrica* 46 (6): 1429–45. doi:10.2307/1913837.

Lukac, Louis P., B. Wade Brorsen, and Scott H. Irwin. 1988. "A Test of Futures Market Disequilibrium Using Twelve Different Technical Trading Systems." *Applied Economics* 20 (5): 623–39. doi:10.1080/00036848800000113.

Miffre, Joëlle, and Georgios Rallis. 2007. "Momentum Strategies in Commodity Futures Markets." *Journal of Banking & Finance* 31 (6): 1863–86. doi:10.1016/j.jbankfin.2006.12.005.

Moskowitz, Tobias J., Yao Hua Ooi, and Lasse Heje Pedersen. 2012. "Time Series Momentum." *Journal of Financial Economics* 104 (2): 228–50. doi:10.1016/j.jfineco.2011.11.003.

Narayan, Paresh Kumar, Hudson Ali Ahmed, and Seema Narayan. 2015. "Do Momentum-Based Trading Strategies Work in the Commodity Futures Markets?" *Journal of Futures Markets* 35 (9): 868–91. doi:10.1002/fut.21685.

Park, Cheol-Ho, and Scott H. Irwin. 2007. "What Do We Know About the Profitability of Technical Analysis?" *Journal of Economic Surveys* 21 (4): 786–826. doi:10.1111/j.1467-6419.2007.00519.x.

Paschke, Raphael, Marcel Prokopczuk, and Chardin Wese Simen. 2020. "Curve Momentum." *Journal of Banking & Finance* 113 (April). doi:10.1016/j.jbankfin.2019.105718.

Shapley, Lloyd S. 1953. "A Value for *n*-Person Games." In *Contributions to the Theory of Games II*, edited by Harold W. Kuhn and Albert William Tucker, 307–17. Princeton, NJ: Princeton University Press.

Simonian, Joseph, Chenwei Wu, Daniel Itano, and Vyshaal Narayanam. 2019. "A Machine Learning Approach to Risk Factors: A Case Study Using the Fama–French–Carhart Model." *Journal of Financial Data Science* 1 (1): 32–44. doi:10.3905/jfds.2019.1.032.

Szakmary, Andrew C., Qian Shen, and Subhash C. Sharma. 2010. "Trend-Following Trading Strategies in Commodity Futures: A Re-Examination." *Journal of Banking & Finance* 34 (2): 409–26. doi:10.1016/j.jbankfin.2009.08.004.

Wang, Shirui, and Tianyang Zhang. 2024. "Predictability of Commodity Futures Returns with Machine Learning Models." *Journal of Futures Markets* 44 (2): 302–22. doi:10.1002/fut.22471.

Working, Holbrook. 1949. "The Theory of Price of Storage." *American Economic Review* 39 (6): 1254–62. www.jstor.org/stable/1816601.

Yen, Stéphane Meng-Feng, and Ying-Lin Hsu. 2010. "Profitability of Technical Analysis in Financial and Commodity Futures Markets—A Reality Check." *Decision Support Systems* 50 (1): 128–39. doi:10.1016/j.dss.2010.07.008.

QUANTUM COMPUTING FOR FINANCE

Oswaldo Zapata, PhD
Cofounder, The Quantum Finance Boardroom

Introduction

This chapter aims to illustrate how quantum computing is expected to transform the future of finance. It provides a concise overview of the fundamental concepts underlying quantum computing, along with several of its most prominent applications in finance, such as portfolio optimization and quantum-enhanced machine learning.

Quantum computing encompasses three related but distinct quantum technologies: quantum computation, quantum communication, and quantum sensing. In this chapter, I primarily focus on quantum computation—the idea that quantum computers can solve certain problems more efficiently and accurately than classical computers. Quantum communication, although currently of interest to financial organizations, will be mentioned briefly. Quantum sensing, which has no application in finance, will not be discussed.

The chapter is organized as follows: First, I provide an overview of quantum computing, emphasizing current capabilities in light of the present state of hardware development. Next, I review applications of machine learning in finance and present the fundamental ideas behind applying quantum computing to financial problems. Then, I discuss quantum communication, and I conclude with a summary and a brief discussion of other topics of interest.

Quantum Computation

Quantum computation is a quantum-mechanical approach to solving computational problems. Instead of relying on traditional Boolean algebra, where information is represented in binary form—either 0 or 1—quantum computation uses the principles and mathematical formalism of quantum mechanics, such as state vectors, unitary operators, and measurements, to arrive at logical solutions to computational problems. A *quantum computer* is the physical device that implements the quantum computational processes of interest.

Quantum mechanics describes nature in a fundamentally different way from classical physics, which relies on such concepts as force, mass, velocity, and electrical currents. With this caveat in mind, however, drawing an analogy between classical and quantum circuits can still be useful.

A standard electronic circuit consists of electrical currents and a series of electronic components known as logic gates. A circuit can be designed specifically to solve a given problem. This arrangement of currents and gates—that is, the circuit—constitutes an algorithm. These circuits are deterministic in the sense that, given a set of input currents and a sequence of gates, the output is uniquely determined. The efficiency of the algorithm is evaluated on the basis of the number of gates and layers it uses—a field known as circuit complexity.

The basic elements of a quantum circuit are the quantum analogues of electric currents and electronic components. The "quantum electric currents" are called *qubits*, and the "quantum electronic components" are called *quantum gates*. An arrangement of quantum gates forms a *quantum circuit*. When a quantum circuit is designed to solve a particular computational problem, it is referred to as a *quantum algorithm*. In contrast to classical circuits, quantum circuits are generally nondeterministic because the *measurement* process—an integral part of a quantum circuit—typically produces a probabilistic output. For example, in a classical circuit, one always measures either the presence or absence of a current in a predictable way. In contrast, in a quantum circuit, measuring the same qubit can yield different outcomes—sometimes indicating the presence of a current and other times indicating its absence—even if the circuit is unchanged.

In mathematical terms, qubits are represented by vectors in a normed complex vector space, and quantum gates are realized by unitary transformations. The most basic qubit is a *1-qubit*. This quantum state is represented by a complex linear combination of two nonparallel vectors (usually taken as orthogonal). Quantum gates acting on single qubits change the configuration of the qubit by modifying the complex coefficients. For example, a quantum gate can exchange one of the basis vectors for the other or simply make it vanish. The measurement process at the end provides a probabilistic result based on the new combination created by the quantum gate, or gates, depending on the complexity of the problem. For more general qubits, we use the term *n-qubit*. The vector describing this quantum state is more complicated: It involves 2^n nonparallel vectors, usually orthogonal, and 2^n complex coefficients. Quantum gates are *unitary transformations* that change the coefficients of the input qubit. Quantum gates and circuits more generally leverage the *entanglement property* of quantum mechanics, which involves creating an output qubit that contains more information than the individual input qubits used to create it.

A final word about quantum circuits: It has been known since the early days of modern computing that any classical circuit can be constructed using a small set of electronic components. Such a set is known as a universal gate set. Similarly, any quantum operation can, in theory, be approximated using a finite set of basic gates—known as a *universal quantum gate set*.

The motivation for developing quantum computers stems from two central beliefs:

- They may significantly outperform classical devices on certain tasks, providing exponential or polynomial increases in speed.
- Quantum machines might be able to solve problems deemed intractable for classical computers—offering not only faster computation but also fundamentally new capabilities.

Although this potential remains largely theoretical at present, growing evidence and experimental progress suggest that quantum computation could revolutionize the current approach to complex problems across various scientific and technological fields, including finance. Significant challenges remain, however, and here I highlight two: (1) the difficulty of preserving the quantum properties of qubits and quantum gates and (2) the challenge of constructing larger, more complex quantum circuits.

If a qubit or quantum gate unpredictably changes its quantum properties, the theoretical predictions will no longer align with the experimental results. This discrepancy can compromise the entire quantum computation. Such unpredictability is referred to as an *error*. Fortunately, some

errors can be corrected. This challenge is not entirely new. In the early days of classical computing, digital components were also imperfect, and *error correction methods* were essential. Over time, classical hardware became so reliable that such corrections became largely unnecessary. In contrast, quantum systems remain highly sensitive and difficult to isolate from their environments. Even minor interactions—collectively known as *noise*—can disrupt quantum behavior. To tackle this problem, engineers work on techniques to shield quantum circuits from environmental noise, and theorists develop procedures to detect and correct various types of errors.

A common example is a bit-flip error, where the state of a qubit flips unexpectedly during transmission or processing. In classical computing, such errors are corrected by duplicating the bit and using majority voting; for example, instead of using 0, we use 000. If one copy is flipped, the system can identify and fix the mistake by comparing the values. This method assumes that the probability of an error occurring is extremely low; if multiple errors occur, the correction fails. Quantum computing applies a similar idea but with important differences. Instead of copying qubits directly (impossible because of quantum rules), the information is spread across multiple qubits in a way that allows for error detection and correction. These groups of qubits are called *logical qubits*, and each individual qubit in the group is referred to as a *physical qubit*.

Detecting errors in quantum systems is more delicate than in classical ones because directly measuring a qubit destroys its state. Instead, quantum computers use indirect methods—such as *parity checks*—to detect errors without collapsing the quantum information. Once an error is identified, correction techniques are applied to restore the original state. Beyond simple bit flips, other types of errors can affect qubits, including those caused by faulty gates. If a qubit enters a faulty gate, the error may propagate throughout the quantum circuit. Worse still, the process of correcting errors also involves quantum components—which means it can introduce new errors. This dynamic creates a paradox: Fixing errors can sometimes cause more of them. As a result, building reliable quantum computers requires scaling up the system, which further increases the chances of something going wrong. Fortunately, researchers have proven that if certain conditions are met, error correction codes can reduce error rates to very low levels. A *fault-tolerant quantum computer* is one that continually detects and corrects errors in its logical qubits throughout the computation, ensuring the final result is reliable.

In recent years, however, scientists have come to accept that fault-tolerant quantum computers will not be available anytime soon. As a result, they began looking for more realistic algorithms that could be implemented on near-term quantum computers, which are characterized by a moderate amount of noise and a relatively small number of qubits and gates. We are currently in this stage, known as the *noisy intermediate-scale quantum* (NISQ) era. According to experts, we will likely remain in this phase for several years (even decades) before achieving fault tolerance.

The algorithms expected to be implemented in the near term are known as *hybrid quantum–classical algorithms*. These combine quantum and classical parts: The quantum subroutines address problems that are hard for classical computers, while the classical computer handles tasks where its efficiency is well established. For example, *variational quantum algorithms* (VQAs) are NISQ algorithms designed to demonstrate *quantum advantage* (practical *quantum supremacy*) in the near future. Because many problems—not only in physics and chemistry but also in finance—share a common underlying structure, the techniques used in VQAs can be applied to a wide variety of situations. VQAs are considered *heuristic*, which means that

although there is currently no rigorous proof that they outperform known classical algorithms, there are theoretical reasons for optimism. The hope is that future results will demonstrate their advantage.

Machine Learning for Finance

As mentioned in several chapters of this book, *artificial intelligence* (AI) is transforming many industries, including finance. Banks and public institutions are using AI to detect fraud, assess credit risk, and identify investment opportunities. This section focuses on *machine learning* (ML)—a subfield of AI concerned with algorithms that seek to uncover patterns in data. In finance, where data such as stock movements and customer behavior are abundant, ML models are used to analyze this information and generate actionable insights. In this section, I will review several ML algorithms currently used in finance—especially those seen as promising candidates for enhancement through quantum computing. But first, it is essential to clarify what is meant by "data."

The term *data* refers to information associated with physical objects or abstract concepts. In ML, such information comes in various forms and is often classified into two major types. *Structured data* are organized and easily represented in tabular formats, such as matrices. In contrast, *unstructured data* lack this inherent organization; examples include raw text, images, and audio recordings. Despite being more prevalent in the real world, unstructured data must first undergo cleaning and formatting before they can be used in ML models. If the data are incomplete, inconsistent, or poorly selected, models built on such foundations may yield inaccurate or misleading predictions. Therefore, the preprocessing stage is not a peripheral step but a foundational component of the machine learning pipeline.

Supervised learning is among the most widely used ML paradigms. In this approach, models are trained on *labeled data*, where the inputs and their corresponding outputs are known in advance. The objective is to learn a mapping from inputs to outputs that generalizes well to new, unseen data. During the *training phase*, the model iteratively adjusts itself to minimize the discrepancy between its predictions and the actual labels. This adjustment process is typically achieved by minimizing a *loss function*, and the model's ability to generalize is then evaluated using a separate *test set*. Supervised learning tasks are commonly divided into regression and classification problems. In *regression*, the model predicts continuous outcomes. *Classification* tasks, in contrast, involve assigning discrete labels to data points.

Unsupervised learning diverges from supervised learning in a fundamental way: It operates on datasets that lack labeled outputs. The aim of unsupervised learning algorithms is to uncover hidden structures or patterns in the data without the aid of explicit guidance. In these cases, the input consists solely of features, and the algorithm is tasked with identifying intrinsic relationships among them. One common challenge in such scenarios is managing high-dimensional data, where the number of features is large. Reducing the dimensionality of the feature space becomes an essential step—not only to simplify the data but also to enhance the efficiency of subsequent analysis. One systematic method to perform such feature reduction is *principal component analysis* (PCA), a key technique under the broader category of *dimensionality reduction*.

Neural networks represent a class of ML models that can be supervised or unsupervised. Inspired by the structure of the human brain, these models consist of layers of interconnected

nodes, or *neurons*. Each neuron performs a computation based on its inputs and transmits the result to the next *layer*. The network is trained by adjusting internal parameters, such as weights and biases, to optimize an objective function—often using such algorithms as *gradient descent*. For tasks involving sequential data, *recurrent neural networks* are often used. These networks differ from traditional feed-forward models in that they maintain a hidden state, enabling them to incorporate information from previous inputs. However, recurrent networks are prone to such issues as the *vanishing gradient problem*, which hinders their ability to capture long-term dependencies. *Long short-term memory networks* address this limitation by introducing gating mechanisms that preserve important information over extended sequences.

ML applications in finance are vast and span various domains, such as credit scoring, fraud detection, portfolio management, and market analysis. Some explicit examples are covered next. The improvement of these ML methods with quantum computers is expected to enhance financial performance and services.

Supervised models, both regression and classification, are particularly useful in credit scoring, where they predict the probability of a borrower repaying a loan or defaulting. The dataset typically includes personal information about borrowers—such as their age, gender, and financial data, including loan amount, credit history, and repayment records. Once the model is trained on this enriched dataset, it can predict the likelihood of loan repayment for new applicants. Financial institutions can then use this prediction to decide whether to approve a loan or set credit limits based on the applicant's perceived creditworthiness. Risk assessment in finance is a more comprehensive task than credit risk evaluation alone. In this context, risk assessment involves predicting both the probability and potential cost of adverse events that could impact a company's financial health. The specific dataset and features used depend on the type of risk being assessed, which could include market risk, credit risk, operational risk, or country risk. If the goal is to assess the risk an event poses to a company's market valuation, relevant data are gathered and the supervised model is trained to uncover patterns. By analyzing the relationships between the input features and target variables (such as changes in market valuation or earnings volatility), financial institutions and corporations can make more-informed decisions to mitigate risks.

The *k-nearest neighbors* (kNN) algorithm is a classification method that is particularly well suited for such tasks as credit risk assessment and fraud detection. In credit risk assessment, for example, kNN can predict whether a potential borrower will repay a loan or default. The algorithm works by training on data from previous borrowers, where each data point consists of personal and financial information along with a repayment history. Once trained, the model compares a new applicant's data to the *k*-nearest neighbors in the training dataset and, through majority voting, predicts whether the applicant is likely to repay or default. Similarly, in transaction fraud detection, kNN compares new transactions with past ones, determining whether new transactions appear fraudulent based on their similarity to previous fraudulent or legitimate transactions. The kNN algorithm can also be used to identify money laundering patterns, although money laundering detection often requires more complex feature engineering and domain-specific knowledge.

PCA is another powerful tool used in finance, particularly for such tasks as credit risk assessment and portfolio management. In credit risk assessment, many features in the dataset are highly correlated, such as income, debt-to-income ratio, and credit utilization. PCA helps by reducing the dimensionality of the dataset while preserving the most important variance

in the data. This process allows risk analysts to focus on the most significant factors influencing credit risk, such as identifying borrower clusters with similar risk profiles or spotting outliers that may represent unique risks. PCA can also be applied to portfolio management, where it helps in reducing data complexity, focusing on the most relevant risk factors for asset allocation. By simplifying the covariance matrix of assets, PCA identifies and removes highly correlated assets, allowing portfolio managers to maintain diversification without unnecessarily complicating the portfolio. The efficiency improvements brought by PCA make it a valuable tool in market analysis as well, where it can help focus on key market drivers while discarding less relevant factors.

The *k-means clustering* algorithm is an unsupervised learning technique useful in detecting previously unseen patterns or suspicious behaviors in data. Unlike supervised learning algorithms, *k*-means does not rely on labeled data, making it more flexible in uncovering unexpected trends. This ability is particularly valuable in evolving scenarios, such as fraud detection and anti-money-laundering, where fraudsters continuously adapt their tactics. By grouping similar data points together and identifying outliers that do not fit well into any cluster, *k*-means can reveal unusual or fraudulent behaviors that might otherwise go unnoticed. Although not explicitly designed as an anomaly detection algorithm, *k*-means' ability to highlight atypical data points makes it an important tool for detecting fraud and preventing money laundering.

Quantum Algorithms for Finance

In the dynamic environment of the modern financial industry—characterized by intense competition and evolving regulations—quantum computing holds great promise because it is expected to surpass classical systems in both efficiency and security. Some experts predict that finance may be one of the first industries to undergo a transformation driven by quantum computing. The timeline for the availability of fully functional quantum computers remains uncertain, however.

Remember that fully reliable, fault-tolerant quantum computers are still many years from being realized. We are in the so-called NISQ era, characterized by quantum devices that are relatively noisy and support only a limited number of quantum gates. As a result, researchers have focused on hybrid quantum–classical algorithms, which combine the strengths of both classical and quantum computing. In these hybrid models, quantum computers tackle the most computationally demanding parts of a problem, while classical computers handle the remaining tasks. This approach offers practical advantages: The quantum subroutines require only a limited number of coherent qubits and shallow circuits, making them feasible with today's quantum technology.

Quantum Portfolio Optimization

One of the key areas where quantum computing can improve on classical methods is portfolio optimization. Traditional portfolio optimization techniques can struggle with large datasets, particularly when the portfolios are vast and require the processing of complex data. Variational quantum algorithms, such as the *variational quantum eigensolver* (VQE) and the *quantum approximate optimization algorithm* (QAOA), are believed to offer improvements by processing large datasets more efficiently than classical algorithms.

The VQE is an algorithm that leverages the variational principle of quantum mechanics to approximate solutions to the time-independent Schrödinger equation. This equation describes the behavior of complex quantum systems, such as molecules or the electronic configurations of materials. The VQE was originally developed for quantum chemistry applications, but researchers have explored adapting it to represent the quantized version of certain classical problems, including portfolio optimization, by reformulating them as Hamiltonian minimization tasks. The QAOA is another variational algorithm, specifically designed to solve classical combinatorial optimization problems, such as portfolio optimization. What makes VQAs particularly compelling is that they are designed for hybrid classical–quantum computers. The quantum subroutine prepares quantum states and computes the Hamiltonian expectation values, while the classical computer performs the optimization process.

This hybrid approach makes VQAs suitable for solving complex problems that are challenging for purely classical systems, especially in such domains as portfolio optimization. Although it is still uncertain whether VQAs will outperform classical algorithms, the potential for significant improvements in processing massive datasets and performing optimization tasks at a faster rate could open up new avenues in financial modeling and analysis.

Quantum Machine Learning

In this section, I briefly address the main challenges quantum computing faces in enhancing and potentially revolutionizing classical ML techniques.

The primary challenge in *quantum machine learning* today lies in effectively encoding classical data into qubits so that the quantum computer can process the data efficiently. Several methods have been developed to facilitate this encoding process, enabling quantum computers to perform computations on classical data.

I will illustrate this challenge with the simplest example. Suppose you have two classical data points—for example, two positive numbers—and you want to insert this information into a quantum computer to process them using the quantum algorithm you have designed. These classical data points must be transformed into quantum information that the quantum computer can understand. Perhaps the simplest way to encode these classical data points into a quantum state is by using the angles of a single qubit, a process known as *angle encoding*. As previously mentioned, a 1-qubit generally requires two complex numbers, which corresponds to four real numbers, to fully specify it. By the principles of quantum mechanics, however, these four real numbers can be reduced to only two. The single qubit can thus be represented as a vector on the surface of a unit sphere. Because the position of any point on the sphere is completely determined by two angles, the azimuthal and polar angles, the 1-qubit is determined by these two angles. By properly rescaling if necessary, the original two classical data points can be encoded in the 1-qubit by rotating it accordingly. That is, the two data points can be encoded in the rotation angles of the 1-qubit. For more complex situations with many more classical data points, general n-qubits are necessary, but the principle remains the same. The real challenge lies in implementing these ideas in real-world scenarios.

In quantum ML, several algorithms have been developed to speed up learning tasks by processing vast amounts of data more efficiently than classical systems. These algorithms can be applied to a variety of ML problems, ranging from classification and regression to clustering

and optimization. In the context of finance, quantum ML holds the potential to improve financial applications, as discussed earlier.

Note that ML is a technology that was only recently incorporated into the financial sector, and quantum ML is still in the early stages of research. As quantum technology progresses, quantum algorithms are expected to become increasingly capable of handling more complex data and providing substantial advantages over classical approaches.

Quantum Cryptography

As mentioned previously, quantum computing is much more than the acceleration of computationally expensive problems. It also encompasses *secure communication*.

The issue is that sufficiently powerful quantum computers could break the current encryption methods used by most public institutions and private organizations, potentially gaining access to sensitive data, such as military and financial information. The threat posed by quantum computers is a reality that governments and financial institutions are taking very seriously. A discussion of these concepts follows.

An *encryption standard* is a method used to transform information into a form that is not easily related to the original. The original form is referred to as *plaintext*, and the transformed version is known as *ciphertext*. Most contemporary digital encryption standards are based on mathematical problems that are difficult for classical computers to solve. Quantum algorithms, however, have the potential to solve some of these problems efficiently. For example, *RSA*, widely used to secure digital data over the internet, relies on the difficulty of integer factorization, and *ECC* (*elliptic curve cryptography*) is based on the discrete logarithm problem. The concern is that *Shor's algorithm*, a quantum algorithm, can efficiently solve both of these problems. Once sufficiently large quantum computers become available—potentially in the next 5–10 years—these encryption systems could be broken in a relatively short period of time. These two examples represent some of the most vulnerable standards in the quantum era. In response, governments worldwide are enacting laws to secure sensitive data.

Post-quantum cryptography (*PQC*) offers one potential solution. It involves the development and, increasingly, the implementation of a set of encryption methods based on mathematical problems designed to remain secure against both classical computers and, more importantly, quantum algorithms running on quantum computers. These algorithms are based on mathematical problems that are not known to be efficiently solvable by quantum algorithms, such as Shor's and Grover's. Examples of such problems include lattice problems, multivariate polynomials, code-based schemes, and hash-based signatures. In the United States, the National Institute of Standards and Technology (NIST) is currently in the process of standardizing several PQC algorithms to either supplement or completely replace existing cryptographic systems. NIST has determined that by 2030, US federal agencies should treat current standard encryption methods as vulnerable, and by 2035, these methods are expected to be phased out. US financial institutions, however, have not yet established a timeline for transitioning to post-quantum cryptography.

Quantum key distribution (QKD) is regarded as one of the most secure encryption methods because it relies not on complex mathematical problems but on the fundamental laws of physics. Information exchange is protected by principles of quantum mechanics—most notably,

the *no-cloning theorem* and the fact that measurement inherently disturbs the system being observed. Although QKD provides strong theoretical security, its practical implementation is currently constrained by high costs, the need for specialized hardware, and distance limitations. As a result, QKD is best suited for niche applications—such as government or high-security financial sectors—that can support dedicated infrastructure.

Transitioning to a fully *quantum-safe* infrastructure—particularly for such institutions as banks—is costly and may take many years to complete. To initiate this transition, experts recommend adopting hybrid cryptosystems that combine traditional (classical) encryption algorithms, which are less expensive and faster to implement, with quantum-resistant algorithms that may require new types of technology and infrastructure. This combination offers protection against both classical and potential future quantum attacks and is especially valuable during the transition phase.

Conclusion

Quantum computing is a technology still in the research phase, awaiting widespread adoption. Although its practical advantages remain limited at this time, it holds the promise of improving the most computationally demanding calculations in such industries as pharmaceuticals, logistics, and finance. In the case of finance, it could help in the process of portfolio optimization, as well as in many services currently tackled by ML techniques. Another active area of research in the finance community is *quantum-enhanced Monte Carlo simulation*. Unlike with variational quantum algorithms and quantum ML, there is mathematical evidence that the quantum version of classical Monte Carlo can enhance parts of the process. Challenges persist, however: Noise still affects quantum systems, and errors remain significant.

Quantum computation and quantum communication are active areas of research, not only in hardware and software but also in real-world applications. The coming years will bring more powerful quantum hardware and a growing interest in applications such as those in finance. The future is certainly exciting for quantum computing and its role in the financial sector.

ETHICAL AI IN FINANCE

Anna Martirosyan
Strategy and Transactions Manager, EY Parthenon

Introduction

Recent technological progress and greater computational power have significantly boosted the adoption of artificial intelligence (AI) by financial institutions and regulators. AI delivers numerous benefits, including streamlined operations, improved regulatory adherence, tailored financial solutions, and advanced data analysis. The introduction of generative AI (GenAI) and large language models has further broadened its range of applications.

AI has significantly shaped financial markets, improving efficiency, returns, and analytical capabilities. Generative AI marks the latest advancement, driving productivity through automation and enhancing such activities as trading and data analysis. Companies that invest heavily in AI experience significant growth, with research showing that a one standard deviation increase in AI investment is associated with a 19.5% rise in sales, an 18.1% increase in employment, and a 22.3% boost in market value over eight years. These trends are consistently observed across such industries as manufacturing, finance, and retail (Babina, Fedyk, He, and Hodson 2024).

Investment professionals heavily use machine learning tools for managing investment processes, benefiting from increased efficiency, automation, timely insights, and enhanced risk management. Although only 29% of systematic investors currently use AI to develop and test investment strategies, more than three-quarters anticipate doing so in the future. AI continues to revolutionize the financial industry by extending capabilities in data analysis, predictive modeling, and automation, enabling the rapid processing of vast datasets (CFA Institute 2024).

Machine learning and large language models enable better price discovery and lower barriers for quantitative investors, potentially improving liquidity but raising financial stability concerns. Despite progress, fully autonomous AI-driven financial agents remain limited as concerns persist about "black box" strategies and regulatory, ethical, and liability issues (Adrian 2024).

Although AI drives advancements in fraud detection, credit risk assessment, and algorithmic trading, it also presents critical ethical challenges regarding bias, transparency, and accountability that must be addressed to ensure responsible and equitable implementation.

This chapter explores the intersection of machine learning and finance through the lens of ethical AI considerations. I will examine ethical challenges, regulatory frameworks, and mitigation strategies while emphasizing adherence to ethical norms to achieve sustainable outcomes. The chapter concludes with actionable recommendations to guide the responsible implementation of AI in finance.

Understanding Ethical AI

Ethical AI refers to the responsible development and deployment of AI systems based on foundational principles, such as transparency, fairness, accountability, privacy, nonmaleficence,

and justice (Jobin, Ienca, and Vayena 2019). These principles aim to guide AI development in a way that aligns with societal values and minimizes harm. However, their interpretation, scope, and implementation vary significantly across cultures, domains, and stakeholders. This diversity necessitates a cross-cultural approach to AI ethics that reflects the plurality of global perspectives (Goffi 2023).

Key Ethical Frameworks

AI ethics frameworks are structured sets of principles designed to ensure fairness, transparency, and accountability in AI systems. With the rapid growth of AI adoption, expected to increase at a compound annual growth rate of over 37% through 2030 (Grand View Research 2024), these frameworks are essential to mitigating risks and fostering trust.

Three prominent frameworks have been introduced by the Institute of Electrical and Electronics Engineers (IEEE), the EU, and the OECD:

- *IEEE's Ethical AI Framework:* Focuses on embedding human values into AI design and fostering accountability and transparency (Peters, Vold, Robinson, and Calvo 2020)
- *The EU AI Act:* A risk-based regulatory framework that categorizes AI systems by risk level, imposing stricter requirements on high-risk applications to ensure safety, transparency, and alignment with human rights[1]
- *The OECD AI Principles:* Center on promoting trustworthy AI by addressing inclusivity, fairness, transparency, and safety while ensuring societal benefit[2]

These frameworks share common goals while addressing different priorities, such as risk mitigation, regulatory compliance, and societal alignment. UNESCO further supports these efforts through its Global AI Ethics and Governance Observatory, which fosters international cooperation, provides resources, and promotes ethical AI practices globally (UNESCO 2024).

Ethical Dilemmas in AI

Despite these efforts, ethical challenges persist in AI's application. Key concerns include the following:

- *Bias:* Technical and human biases arise from such issues as data deficiencies, demographic homogeneity, spurious correlations, and cognitive biases. These biases can lead to discriminatory outcomes, particularly in sensitive domains, such as auditing and hiring (Murikah, Nthenge, and Musyoka 2024).
- *Privacy:* AI systems often rely on large datasets, raising concerns about how personal data are collected, stored, and used.
- *Transparency:* Many AI systems, especially those powered by deep learning, function as "black boxes," making their decision-making processes difficult to interpret or explain.

[1]The latest details on the EU Artificial Intelligence Act can be found online at https://artificialintelligenceact.eu/.

[2]Further details are on the OECD website at www.oecd.org/en/topics/sub-issues/ai-principles.html.

By aligning global efforts and fostering international collaboration, the future of AI can be both innovative and ethically responsible, ensuring its benefits are equitably distributed across societies.

Applications of AI in Finance

Artificial intelligence is revolutionizing the financial services sector, offering transformative applications in fraud detection, risk management, algorithmic trading, and credit scoring. By using advanced techniques such as machine learning (ML), deep learning (DL), and natural language processing, financial institutions are enhancing their capabilities to detect anomalies, mitigate risks, and optimize decision-making processes. Despite its potential, however, AI's adoption in finance faces challenges, including concerns around data quality, model transparency, and regulatory compliance (Pattyam 2019).

Fraud detection has emerged as one of the most impactful applications of AI in finance. AI has been shown to improve audit efficiency, minimize financial losses, and bolster stakeholder trust (Kamuangu 2024). Algorithms such as random forest are particularly effective at identifying fraudulent transactions with high precision, as noted by Lin (2024). These advancements not only enhance security but also enable financial institutions to proactively safeguard their operations against financial crimes. To address concerns about the opaque nature of AI systems, explainable AI (XAI) techniques, such as feature importance analysis and LIME (local interpretable model-agnostic explanations), have been used (Gayam 2021). These approaches demystify AI's decision-making processes, fostering greater trust in its applications.

Risk management is another area where AI has shown remarkable promise. With the ability to analyze vast datasets in real time, AI systems can identify emerging threats and adapt to new challenges. Javaid (2024) emphasized the transformative potential of AI in recognizing patterns, detecting anomalies, and responding to risks dynamically. This adaptability not only reduces financial risks but also strengthens consumer trust, ultimately enhancing the resilience of financial ecosystems.

In the realm of algorithmic trading, AI is reshaping market dynamics by processing extensive data and executing trades at unparalleled speed and accuracy. Cohen (2022) highlighted how ML and DL techniques can uncover hidden correlations and predict market trends, allowing AI systems to outperform human traders in efficiency and effectiveness. Similarly, Beverungen (2019) noted that AI is redefining the so-called cognitive ecology of financial markets, enabling smarter and faster trading strategies that challenge traditional human capabilities.

AI also plays a pivotal role in credit scoring by improving the accuracy and fairness of risk assessments. By analyzing nontraditional data sources, such as social media behavior and transaction histories, AI systems reduce reliance on conventional metrics, thereby increasing financial inclusivity for underserved populations. This shift enables financial institutions to make more informed and equitable lending decisions.

ML underpins these advancements by driving predictive analytics, data-driven insights, and automation across financial services. For example, ML models enhance fraud detection by analyzing historical and real-time data for suspicious activities, and self-learning algorithms in risk management adapt to emerging threats to ensure dynamic mitigation. In trading, DL models identify complex patterns in financial data, optimizing strategies and maximizing returns.

Additionally, ML-powered credit scoring systems use diverse data sources to deliver more accurate and less biased assessments of creditworthiness.

The integration of AI into financial accounting and management further demonstrates its potential to drive efficiency and innovation. AI, combined with robotic process automation, extends traditional accounting capabilities, enabling comprehensive data analysis and adaptability to evolving business processes (Zhan, Ling, Xu, Guo, and Zhuang 2024). These technologies contribute to an integrated system that enhances enterprise management, ensuring that financial institutions can respond effectively to complex decision-making challenges.

Investment management firms are also turning to AI to relieve the burden of regulatory compliance, allowing human resources to be deployed in more productive, client-focused endeavors. Firms can streamline compliance processes by automating regulatory compliance with AI applications that, for instance, complete mandatory reporting forms by extracting data from internal systems in a timely and accurate manner. Tools track and interpret applicable regulatory changes by monitoring the rule-making activity of regulatory bodies to scan and interpret new guidance, enforcement actions, and rule updates. These tools then apply those changes to firm processes and practices and suggest updates to compliance policies and procedures. Enhanced transaction monitoring can detect unethical and illegal activity, such as insider trading, market manipulation, front running, or money laundering. ML can distinguish normal customer activity from suspicious activity more accurately than rule-based systems, learn from historical trading violations, and adapt to evolving marketing behavior. In sum, using AI for compliance helps enhance monitoring of financial activities and changes in the regulatory landscape to reduce operational risk.

Despite its vast potential, the adoption of AI in finance is not without obstacles. Such issues as model interpretability, data quality, and compliance with regulatory frameworks must be addressed to fully harness AI's capabilities. Nevertheless, the advancements in fraud detection, risk management, algorithmic trading, and credit scoring illustrate how AI is fundamentally transforming the financial sector, offering unprecedented opportunities for innovation and growth.

Ethical Challenges in Financial AI

The integration of big data analytics, AI, and blockchain technologies into financial services has marked the beginning of a new era of enhanced inclusivity and accessibility. As highlighted by Giudici (2018), however, these innovations also bring significant ethical and operational risks. Such issues as underestimation of creditworthiness, market risk noncompliance, fraud, and cyberattacks are becoming increasingly prevalent as financial institutions rely on automated systems. These concerns necessitate the adoption of robust ethical and regulatory frameworks to safeguard the financial ecosystem (Financial Stability Board 2024).

Data privacy and security emerge as critical considerations in financial AI systems. Protecting sensitive customer data and adhering to regulatory requirements are essential during automated data processing (Zhan et al. 2024). Without strict data governance practices, financial AI systems risk compromising customer trust and violating privacy laws. As AI continues to evolve and integrate with other technologies, such as blockchain, maintaining secure and transparent data practices will remain paramount to building trust and ensuring compliance.

One of the most pressing ethical challenges in financial AI is algorithmic bias and fairness. Biased AI decisions, particularly in such areas as loan approvals and credit scoring, can reinforce societal inequalities and lead to discriminatory outcomes. For instance, improperly trained AI systems may deny loans to specific demographic groups because of biased historical data. Addressing this issue requires responsible AI practices, including diversifying training datasets, implementing fairness metrics, and conducting regular audits to detect and mitigate bias.

Transparency and explainability are equally critical, especially as financial institutions increasingly rely on complex AI models, such as convolutional neural networks and recurrent neural networks. Although these models excel in pattern recognition and time-series predictions, their "black box" nature often makes it difficult for stakeholders to understand how decisions are made. The lack of interpretability not only poses ethical risks but also undermines trust in AI systems. Such strategies as data pretreatment, regularization, and the adoption of explainable AI techniques are essential for improving model interpretability (Karanam, Natakam, Boinapalli, Sridharlakshmi, Allam, Gade, Venkata, Kommineni, and Manikyala 2018). These measures are essential to ensure that financial decisions are transparent, equitable, and accountable.

The ethical challenges of AI extend beyond decision making to include risks associated with algorithmic trading. Although AI systems have revolutionized trading strategies by forecasting trends and optimizing trades, they also raise concerns about market manipulation, systemic bias, and a lack of human oversight. The ethical risks associated with algorithmic trading include the potential for unethical practices and unintended market distortions (Cohen 2022). To address these challenges, a balanced approach combining automation with regulatory oversight and human intervention is crucial.

Finally, the reliance of AI systems on vast amounts of data and computational resources introduces disparities in access to these technologies. Ensuring equitable distribution of computational resources and prioritizing data quality are vital for fostering an ethical and inclusive financial AI ecosystem. Although neural networks have significant potential to enhance trading strategies, their sensitivity to market volatility and dependence on extensive datasets limit their flexibility. As such, ongoing research and policy development are required to mitigate these limitations and maximize the effectiveness of AI in finance.

Regulatory and Governance Considerations

Regulatory and governance frameworks for AI in finance are essential for mitigating the ethical risks associated with its rapid adoption across industries. Considering increasing cyberattacks and data protection vulnerabilities, financial institutions are compelled to implement proactive measures. These include strengthening cybersecurity defenses, ensuring data traceability, and protecting sensitive information.

Such strategic actions are not merely risk averse; they also promote sustainable growth and facilitate the responsible adoption of AI technologies in financial systems. Data management stands as a cornerstone of effective AI governance. Research conducted by Ernst & Young (2020) highlights the need for enhanced data governance, strengthened cybersecurity measures, and privacy-centric strategies. These priorities align with the European Union's broader efforts to build trust and ensure compliance through such initiatives as the European Approach to Excellence and Trust (European Commission 2020).

Globally, the regulatory landscape for AI in finance remains multifaceted and dynamic. Research by A&O Shearman (2024) highlights the EU's leadership with the EU AI Act, a comprehensive framework categorizing AI systems by risk and imposing corresponding obligations. This framework balances innovation with ethical compliance. In contrast, the United States has adopted a state-level regulatory approach focusing on risk management and impact assessments, while the United Kingdom has taken a sector-specific route, assigning regulatory bodies to address gaps in oversight.

Across regions, data governance remains a universal priority. The EU and the United Kingdom emphasize protections similar to those in the General Data Protection Regulation (GDPR) for data, meaning they provide individuals with similar rights and require businesses to adhere to similar principles of data privacy. The United States focuses on AI-specific risk management frameworks. Extraterritoriality also plays a significant role because regulations frequently apply to AI systems used within a jurisdiction, regardless of their origin.

The handling of third-party providers presents additional regional variation. In the United States, risk management frameworks are promoted, while the EU enforces information and communications technology security through its Digital Operational Resilience Act, and the United Kingdom has developed a dedicated regime for critical third-party providers. These differing approaches underline the challenge of achieving regulatory harmonization across global markets.

In financial services, such regulators as Financial Industry Regulatory Authority (FINRA) in the United States are clarifying the application of existing laws to AI technologies. FINRA's Regulatory Notice 24-09 emphasizes the need for firms to comply with securities laws when leveraging AI, including generative AI, for such activities as data analysis and investor education. Such risks as accuracy, privacy violations, and bias are highlighted, alongside the importance of human oversight and robust supervisory systems. FINRA emphasizes that traditional regulations, such as those governing public communications, apply equally to AI-generated content. As AI adoption accelerates, FINRA advises firms to carefully assess AI applications and ensure compliance with existing regulatory standards (Complex Discovery 2024).

Robust AI policy and governance frameworks are essential for managing the ethical and operational risks posed by AI.[3] Such frameworks are essential for ensuring transparency, accountability, and fairness while balancing innovation with the protection of individual rights, public safety, and societal values. Key governance practices include setting clear objectives for AI systems, ensuring robust data governance, encouraging cross-functional collaboration, conducting regular audits, and investing in education. As AI technologies evolve, governance frameworks must also adapt, incorporating ethical considerations, fostering international collaboration, and remaining responsive to emerging risks and opportunities.

In the future, dynamic and collaborative governance approaches will become ever more essential to manage the challenges posed by rapidly advancing AI technologies. By refining ethical principles, fostering global cooperation, and adapting to technological shifts, regulators and industry stakeholders can harness AI's transformative potential while mitigating its ethical and operational pitfalls. Through such efforts, the financial sector can achieve a balance between

[3]See Scytale's "AI Policy and Governance: Shaping the Future of Artificial Intelligence" webpage: https://scytale.ai/center/grc/ai-policy-and-governance-shaping-the-future-of-artificial-intelligence/.

innovation and responsibility, ensuring that AI adoption aligns with societal and regulatory expectations.

Mitigating Ethical Risks

The increasing reliance on AI in finance brings immense opportunities but also profound ethical challenges. Addressing these risks requires a balanced approach that integrates technical measures, organizational culture, and robust governance, while also embracing a cross-cultural perspective.

Bias Mitigation Techniques

Bias in AI systems can lead to inequitable outcomes, especially in finance, where decisions significantly impact individuals and economies. Effective bias mitigation starts with careful curation of data. Training datasets must be diverse and representative, reflecting a broad range of demographics, socioeconomic factors, and global perspectives. This approach helps reduce the risk of perpetuating systemic biases and fosters greater inclusion of marginalized groups.

In addition, features in AI models must be scrutinized to avoid correlations with protected characteristics, such as race, gender, or age. Regular audits—both internal and external—are vital for continuously evaluating AI systems for fairness and discrimination. These efforts must be complemented by debiasing techniques, including reweighting training data, modifying learning algorithms, and postprocessing adjustments. Bias detection software provides additional assurance, offering specific insights into algorithmic fairness.

A cross-cultural approach further enhances bias mitigation by incorporating diverse ethical perspectives. This ensures AI systems align with global values and reflect the plurality of the world's viewpoints, reducing ethical risks. As Goffi (2023) asserted, diversity enriches our world and must be translated into AI ethics to represent and honor a variety of perspectives.

Enhancing Transparency with Explainable AI

Transparency is foundational to ethical AI. XAI tools make AI decisions understandable, enabling stakeholders to detect and address biases and fostering trust in the technology. XAI not only enhances accountability but also supports compliance with ethical and regulatory standards.

Transparency also involves clear communication about methodologies, data sources, and system limitations. Engaging regularly with stakeholders—including customers, regulators, and advocacy groups—ensures alignment with societal and ethical expectations. Such openness strengthens trust and fosters collaboration in addressing ethical concerns.

Building an Ethical AI Culture

Ethical AI requires more than technical solutions; it demands an organizational culture grounded in fairness, accountability, and transparency. An ethical AI charter can serve as a guiding framework, embedding these principles into every decision-making process.

Continuous learning through workshops, conferences, and ethical simulations equips employees with the skills to navigate complex ethical challenges related to AI use. Implementing

whistleblower protections allows employees to report unethical AI practices or biased outputs safely, fostering accountability. Recognizing and rewarding ethical behavior further strengthens the organizational commitment to ethical AI, encouraging responsible use of AI in generating outputs that reflect fairness, transparency, and inclusivity.

Conclusion and Recommendations

Summary of Key Points

AI has emerged as a transformative force in finance, enhancing efficiencies, improving decision making, and offering innovative solutions to long-standing challenges. Its rapid adoption introduces ethical complexities, however, including bias, lack of transparency, and accountability. This chapter underscores the importance of embedding ethical AI principles—transparency, fairness, nonmaleficence, accountability, and privacy—into AI systems to create a sustainable, trustworthy, and equitable financial ecosystem. Effective governance, transparency, and proactive regulation are crucial to addressing such risks as algorithmic bias, privacy violations, and the balance between automation and human oversight. Moreover, the cross-disciplinary training of regulators is essential to bridge the gap between AI technology and domain-specific ethical standards.

Practical Recommendations

To implement ethical AI in financial institutions, the following actions are recommended:

- *Transparency and accountability:* Ensure that AI systems are explainable and transparent, enabling stakeholders to understand decision-making processes. Regular audits and the use of explainable AI tools are key to detecting biases and enhancing trust.
- *Bias mitigation:* Pursue strategies to reduce bias in AI, such as using diverse datasets, conducting regular audits, and applying debiasing techniques to ensure fairness and equity.
- *Data privacy and security:* Prioritize cybersecurity and data governance practices to safeguard sensitive financial data, ensuring compliance with privacy regulations, such as GDPR, and mitigating data breach risks.
- *Human oversight:* Ensure that human oversight remains central to AI decision making, particularly in high-stakes financial situations, to complement and guide AI's capabilities.
- *Regulatory engagement:* Collaborate with regulators to ensure that AI adoption aligns with evolving ethical standards and legal requirements. Cross-disciplinary training for regulators, provided by industry groups, such as the Montreal AI Ethics Institute or the AI Now Institute, will help bridge the knowledge gap (Montreal AI Ethics Institute 2020).

The role of AI ethics in finance will continue to evolve as AI technologies advance and become further embedded in financial services. The increasing complexity of AI decision making will require adaptive governance frameworks to address new ethical challenges. As AI systems become more integrated into financial ecosystems, collaboration among policymakers, developers, and financial institutions will be crucial to ensure AI's transformative power is harnessed ethically (Huriye 2023).

Although AI holds tremendous potential to revolutionize financial services, its adoption must be guided by a commitment to ethical principles. Prioritizing transparency, fairness, and accountability while maintaining a balance between automation and human oversight will enable the financial sector to use AI responsibly and sustainably. Embedding ethical AI principles into AI systems will build a trustworthy and equitable financial ecosystem, ensuring AI's benefits are realized without compromising ethical standards. Future developments in AI will bring new ethical considerations, and ongoing dialogue and innovation will be essential to creating a fair, sustainable, and accountable financial ecosystem.

References

Adrian, Tobias. 2024. "Artificial Intelligence and Its Impact on Financial Markets and Financial Stability." Speech delivered at Bund Summit 2024 "Navigating a Changing World," Shanghai, China. International Monetary Fund. www.imf.org/en/News/Articles/2024/09/06/sp090624-artificial-intelligence-and-its-impact-on-financial-markets-and-financial-stability.

A&O Shearman. 2024. "Zooming In on AI #6: AI Under Financial Regulations in the U.S., EU and U.K.—A Comparative Assessment of the Current State of Play: Part 2" (7 October). www.aoshearman.com/en/insights/ao-shearman-on-tech/zooming-in-on-ai-ai-under-financial-regulations-in-the-us-part-2.

Babina, Tania, Anastassia Fedyk, Alex He, and James Hodson. 2024. "Artificial Intelligence, Firm Growth, and Product Innovation." *Journal of Financial Economics* 151 (January). doi:10.1016/j.jfineco.2023.103745.

Beverungen, Armin. 2019. "Algorithmic Trading, Artificial Intelligence and the Politics of Cognition." In *The Democratization of Artificial Intelligence: Net Politics in the Era of Learning Algorithms*, edited by A. Sudmann, 77–93. doi:10.25969/mediarep/13550.

CFA Institute. 2024 "How Machine Learning Is Transforming the Investment Process" (4 March). www.cfainstitute.org/insights/articles/how-machine-learning-is-transforming-the-investment-process.

Cohen, Gil. 2022. "Algorithmic Trading and Financial Forecasting Using Advanced Artificial Intelligence Methodologies." *Mathematics* 10 (18). doi:10.3390/math10183302.

Complex Discovery. 2024. "FINRA Reaffirms Regulatory Standards for AI Adoption in Financial Services" (3 July). https://complexdiscovery.com/finra-reaffirms-regulatory-standards-for-ai-adoption-in-financial-services/.

Ernst & Young. 2020. "Three Priorities for Financial Institutions to Drive a Next-Generation Data Governance Framework."

European Commission. 2020. "On Artificial Intelligence—A European Approach to Excellence and Trust." White paper (19 February). https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52020DC0065.

Financial Stability Board. 2024. "The Financial Stability Implications of Artificial Intelligence" (14 November). www.fsb.org/2024/11/the-financial-stability-implications-of-artificial-intelligence/.

Gayam, Swaroop Reddy. 2021. "Artificial Intelligence for Financial Fraud Detection: Advanced Techniques for Anomaly Detection, Pattern Recognition, and Risk Mitigation." *African Journal of Artificial Intelligence and Sustainable Development* 1 (2): 377–412. https://africansciencegroup.com/index.php/AJAISD/article/view/142.

Giudici, Paolo. 2018. "Fintech Risk Management: A Research Challenge for Artificial Intelligence in Finance." *Frontiers in Artificial Intelligence* 1 (26 November). doi:10.3389/frai.2018.00001.

Goffi, Emmanuel R. 2023. "Teaching Ethics Applied to AI from a Cultural Standpoint: What African 'AI Ethics' for Africa?" In *AI Ethics in Higher Education: Insights from Africa and Beyond*, edited by Caitlin C. Corrigan, Simon Atuah Asakipaam, Jerry John Kponyo, and Christoph Luetge, 13–26. Cham, Switzerland: Springer International Publishing.

Grand View Research. 2024. "Artificial Intelligence Market Size, Share & Trends Analysis Report by Solution, by Technology (Deep Learning, Machine Learning, NLP, Machine Vision, Generative AI), by Function, by End-Use, by Region, and Segment Forecasts, 2024–2030." https://www.grandviewresearch.com/industry-analysis/artificial-intelligence-ai-market.

Huriye, Aisha Zahid. 2023. "The Ethics of Artificial Intelligence: Examining the Ethical Considerations Surrounding the Development and Use of AI." *American Journal of Theoretical and Applied Ethics* 2 (1). https://gprjournals.org/journals/index.php/AJT/article/view/142.

Javaid, Haider Ali. 2024. "How Artificial Intelligence Is Revolutionizing Fraud Detection in Financial Services." *Innovative Engineering Sciences Journal* 10 (1).

Jobin, Anna, Marcello Ienca, and Effy Vayena. 2019. "The Global Landscape of AI Ethics Guidelines." *Nature Machine Intelligence* 1 (2 September): 389–99. www.nature.com/articles/s42256-019-0088-2.

Kamuangu, Paulin. 2024. "A Review on Financial Fraud Detection Using AI and Machine Learning." *Journal of Economics, Finance and Accounting Studies* 6 (1): 67–77. doi:10.32996/jefas.2024.6.1.7.

Karanam, Raghunath Kashyap, Vineel Mouli Natakam, Narasimha Rao Boinapalli, Narayana Reddy Bommu Sridharlakshmi, Abhishekar Reddy Allam, Pavan Kumar Gade, Satya Surya Mklg Gudimetla Naga Venkata, Hari Priya Kommineni, and Aditya Manikyala. 2018. "Neural Networks in Algorithmic Trading for Financial Markets." *Asian Accounting and Auditing Advancement* 9 (1): 115–26. www.researchgate.net/publication/383035503.

Lin, Ahn Kun. 2024. "The AI Revolution in Financial Services: Emerging Methods for Fraud Detection and Prevention." *Jurnal Galaksi: Global Knowledge, Artificial Intelligence and Information System* 1 (1): 43–51. doi:10.70103/galaksi.v1i1.5.

Montreal AI Ethics Institute. 2020. "Report Prepared by the Montreal AI Ethics Institute for the European Commission's Whitepaper on AI 2020 on Artificial Intelligence—A European Approach to Excellence and Trust." arXiv. https://arxiv.org/ftp/arxiv/papers/2006/2006.09428.pdf.

Murikah, Wilberforce, Jeff Kimanga Nthenge, and Faith Mueni Musyoka. 2024. "Bias and Ethics of AI Systems Applied in Auditing—A Systematic Review." *Scientific African* 25 (September). doi:10.1016/j.sciaf.2024.e02281.

Pattyam, Sandeep Pushyamitra. 2019. "AI in Data Science for Financial Services: Techniques for Fraud Detection, Risk Management, and Investment Strategies." *Distributed Learning and Broad Applications in Scientific Research* 5: 385–416. https://dlabi.org/index.php/journal/article/view/123.

Peters, Dorian, Karina Vold, Diana Robinson, and Rafael A. Calvo. 2020. "Responsible AI—Two Frameworks for Ethical Design Practice." *IEEE Transactions on Technology and Society* 1 (1): 34–47. doi:10.1109/TTS.2020.2974991.

UNESCO. 2024. "Recommendation on the Ethics of Artificial Intelligence." www.unesco.org/en/artificial-intelligence/recommendation-ethics.

Zhan, Xiaoan , Zhipeng Ling, Zeqiu Xu, Lingfeng Guo, and Shikai Zhuang. 2024. "Driving Efficiency and Risk Management in Finance Through AI and RPA." *Preprints* (30 October). doi:10.20944/preprints202407.0083.v2.

Named Endowments

CFA Institute Research Foundation acknowledges with sincere gratitude the generous contributions of the Named Endowment participants listed below.

Gifts of at least US$100,000 qualify donors for membership in the Named Endowment category, which recognizes in perpetuity the commitment toward unbiased, practitioner-oriented, relevant research that these firms and individuals have expressed through their generous support of CFA Institute Research Foundation.

Ameritech
Anonymous
Robert D. Arnott
Theodore R. Aronson, CFA
Asahi Mutual Life Insurance Company
Batterymarch Financial Management
Boston Company
Boston Partners Asset Management, L.P.
Gary P. Brinson, CFA
Brinson Partners, Inc.
Capital Group International, Inc.
Concord Capital Management
Dai-Ichi Life Insurance Company
Daiwa Securities
Mr. and Mrs. Jeffrey Diermeier
Gifford Fong Associates
John A. Gunn, CFA
Investment Counsel Association of America, Inc.
Jacobs Levy Equity Management
Jon L. Hagler Foundation
Long-Term Credit Bank of Japan, Ltd.
Lynch, Jones & Ryan, LLC
Meiji Mutual Life Insurance Company
Miller Anderson & Sherrerd, LLP
John B. Neff, CFA
Nikko Securities Co., Ltd.
Nippon Life Insurance Company of Japan
Nomura Securities Co., Ltd.
Payden & Rygel
Provident National Bank
Frank K. Reilly, CFA
Salomon Brothers
Sassoon Holdings Pte. Ltd.
Scudder Stevens & Clark
Security Analysts Association of Japan
Shaw Data Securities, Inc.
Sit Investment Associates, Inc.
Standish, Ayer & Wood, Inc.
State Farm Insurance Company
Sumitomo Life America, Inc.
T. Rowe Price Associates, Inc.
Templeton Investment Counsel Inc.
Frank Trainer, CFA
Travelers Insurance Co.
USF&G Companies
Yamaichi Securities Co., Ltd.

Senior Research Fellows

Financial Services Analyst Association

For more on upcoming CFA Institute Research Foundation publications and webcasts, please visit www.cfainstitute.org/research/foundation.

CFA Institute Research Foundation Board of Trustees 2025–2026

Chair
Jeffery V. Bailey, CFA

Vice Chair
Susan Spinner, CFA

President and CEO, CFA Institute
Margaret Franklin, CFA

*Emeritus

Giuseppe Ballocchi, PhD, CFA
Aaron Brown, CFA
Cheng Sung, PhD
Bill Fung, PhD*
Will Goodhart

Philip Graham, CFA
Joanne M. Hill, PhD
Aaron Low, CFA*
Lotta Moberg, PhD, CFA*
Kurt D. Winkelmann, PhD

Officers and Directors

Co-Director of Research
Luis Garcia-Feijóo, CFA, CIPM

Co-Director of Research
Lionel Martellini

Gary P. Brinson Director of Research Emeritus
Laurence B. Siegel

Treasurer
Kim Maynard

Secretary
Lydia Ooghe

Strategic Operations Manager
Stacie Shure

For more information on members of the CFA Institute Research Foundation Board of Trustees, please visit https://rpc.cfainstitute.org/research-foundation.

Research Foundation Review Board

William J. Bernstein, PhD
Elroy Dimson, PhD
William N. Goetzmann, PhD
Elizabeth R. Hilpman
Paul D. Kaplan, PhD, CFA
Robert E. Kiernan III
Andrew W. Lo, PhD
Stephen Sexauer

ISBN 978-1-952927-64-5

Available online at
rpc.cfainstitute.org