Statistical Arbitrage in Options Markets by Graph Learning and Synthetic Long Positions

Yoonsik Hong

Department of Industrial Engineering and Management Sciences, Northwestern University, YoonsikHong2028@u.northwestern.edu

Diego Klabjan

Department of Industrial Engineering and Management Sciences, Northwestern University, d-klabjan@northwestern.edu

Abstract. Statistical arbitrages (StatArbs) driven by machine learning has garnered considerable attention in both academia and industry. Nevertheless, deep-learning (DL) approaches to directly exploit StatArbs in options markets remain largely unexplored. Moreover, prior graph learning (GL)—a methodological basis of this paper—studies overlooked that features are tabular in many cases and that tree-based methods outperform DL on numerous tabular datasets. To bridge these gaps, we propose a two-stage GL approach for direct identification and exploitation of StatArbs in options markets. In the first stage, we define a novel prediction target isolating pure arbitrages via synthetic bonds. To predict the target, we develop RNConv, a GL architecture incorporating a tree structure. In the second stage, we propose SLSA—a class of positions comprising pure arbitrage opportunities. It is provably of minimal risk and neutral to all Black-Scholes risk factors under the arbitrage-free assumption. We also present the SLSA projection converting predictions into SLSA positions. Our experiments on KOSPI 200 index options show that RNConv statistically significantly outperforms GL baselines, and that SLSA consistently yields positive returns, achieving an average P&L-contract information ratio of 0.1627. Our approach offers a novel perspective on the prediction target and strategy for exploiting StatArbs in options markets through the lens of DL, in conjunction with a pioneering tree-based GL.

Key words: Statistical Arbitrage, Graph Learning, Options Market, Deep Learning

1. Introduction

We address the problem of directly identifying and exploiting statistical arbitrage (StatArb) opportunities in options markets by graph learning (GL). The concepts of StatArb strategies, though defined differently across studies, can be summarized as a class of trading strategies that generate a positive expected return with an acceptable risk of loss by leveraging price discrepancies (Lazzarino et al. 2018). StatArbs have attracted considerable attention in both academia and the financial industry, where they have been widely researched and applied (Zhan et al. 2022). To capture and exploit StatArbs in options markets, in this paper, we develop a GL method and trading strategy.

Although various machine learning approaches have been proposed in the options markets, they did not focus on direct identification of StatArbs. Specifically, Zapart (2003) treated StatArbs as an ancillary subject, and although their strategy was labeled as such, it remained substantially exposed to the canonical risk factors of Black and Scholes (1973). Horikawa and Nakagawa (2024), François et al. (2025) employed StatArbs solely as a lens through which to compare deep hedging and delta

hedging strategies, rather than as a phenomenon to be directly uncovered. Other machine learning studies in the options markets likewise fall short of addressing the direct identification of StatArbs. Consequently, the direct detection of StatArbs via machine learning remains largely uncharted in the literature. Our approach directly identifies and capitalizes on StatArbs, while maintaining a prudently low level of risk.

Prior GL methods, a core methodological basis of this study, overlooked that features are tabular in many cases. GL excels at learning structured representations through node and edge embeddings (Bhatti et al. 2023). Options market data are inherently tabular but exhibit relational structure. For instance, the maturity and strike prices are tabular features, and options sharing the same underlying instrument are related. Treating such features as node or edge features enables the straightforward use of GL; however, tabular data are not embeddings. Although deep learning can embed such features, it underperforms compared to tree-based models on numerous tabular datasets (Shwartz-Ziv and Armon 2022, Borisov et al. 2024, McElfresh et al. 2023, Rana et al. 2023, Yıldız and Kalayci 2025). Hence, rather than relying on deep-learning-based embeddings, tree-based embeddings within GL architectures can be a more promising avenue, which is realized in our method based on tree-based embeddings.

We present a two-step approach bridging the two key gaps related to the subject and architectures: the lack of StatArb trading strategies in options markets and the insufficient consideration of the tabular features in GL. The first step focuses on developing a graph nueral network (GNN) architecture that incorporates a tree-based architecture to detect StatArb opportunities, while the second step aims to exploit the identified opportunities through an appropriate, new trading strategy.

In the first step, we develop a GNN architecture, **R**evised **N**eural oblivious decision ensemble graph **Conv**olution (RNConv), along with a prediction target that exclusively contains arbitrages. Distinct from prior studies, our prediction target is a provably pure arbitrage opportunity, arising from deviations in the prices of synthetic zero-coupon bonds constructed from synthetic-long and underlying instrument positions. These prices should be identical under the law of one price—or more strictly, under the **a**rbitrage-**f**ree (AF) assumption—but diverge in practice (see Figure 2) due to market inefficiencies. Next, we construct graphs where a node is a put-call pair sharing the same maturity and strike price, with node features comprising past arbitrage opportunities, implied volatility, moneyness, and time to maturity. Then, RNConv extracts market-level and idiosyncratic information from the graph and each node, respectively, to predict arbitrage opportunities. Our RNConv architecture incorporates a tree structure built upon **N**eural **O**blivious **D**ecision **E**nsembles

(NODE) (Popov et al. 2020), yielding superior performance compared to other GNN architectures on the KOSPI 200 index option datasets. RNConv is a single model combining a standard GNN with differentiable trees and is trained as a single unit.

To exploit StatArbs in the second step, we present **S**ynthetic-**L**ong-**S**hort-**A**rbitrage (SLSA) and an SLSA projection method. We define SLSA as a class of synthetic-long positions satisfying certain constraints. Every SLSA position exclusively contains arbitrage opportunities, which is confirmed by the property that the price of SLSA positions vanishes under the AF assumption. Moreover, SLSA values exhibit zero variance and are neutral to all Black and Scholes (1973) risk factors under the AF assumption. Every SLSA position carries minimal risk under the assumption of an AF market or the existence of a present value function, both of which are common. Although real-world markets may contain arbitrages, they are expected to become AF (Varian 1987, Langenohl 2018), thereby enabling SLSA to have low risk. Lastly, we present the SLSA projection method which converts predictions into an SLSA position that yields StatArbs.

From our two-step approach, we set forth a novel formulation of the prediction target and strategy for exploiting StatArbs in options markets through the lens of deep learning in conjunction with a pioneering tree-based GNN. The main contributions of this study are as follows.

- We present a novel machine-learning prediction target comprising pure arbitrages in options markets.

- To the best of our knowledge, this work is the first to introduce RNConv, a graph convolutional architecture that integrates neural trees in order to effectively exploit tabular node features while capturing relational structures inherent in option markets.

- To the best of our knowledge, we are the first to propose SLSA, a class of synthetic-long positions that solely contain arbitrage opportunities and are theoretically low-risk and neutral to all Black and Scholes (1973) risk factors under the arbitrage-free assumption, and present the SLSA projection method that transforms arbitrage predictions into such SLSA positions capable of generating StatArbs.

- We demonstrate the effectiveness of our approach on the KOSPI 200 index options, where RNConv statistically significantly outperform benchmarks and SLSA achieves steadily increasing P&Ls with an average P&L-contract information ratio of 0.1627.

This paper is structured as follows. Section 2 reviews literature. Section 3 introduces the preliminaries and background. Section 4 explains the overarching setup and notation. Section 5 presents our proposed methods, including the prediction target, RNConv, and SLSA. Section 6 reports experimental results in the KOSPI 200 index options market. Finally, Section 7 concludes the paper.

2. Related Work

Our study fills gaps in the existing literature concerning learning-based methodologies for StatArbs in option markets, as well as the embedding of tabular features within GNN architectures. In this section, we review relevant prior work with respect to these two aspects, highlight their limitations, and articulate the motivation behind our approach.

2.1. Learning-Based Approaches in Option Markets

Since strong assumptions are required in mathematical models (e.g., geometric Brownian motion in (Black and Scholes 1973), jump diffusion process in (Merton 1976), displaced diffusion process in (Rubinstein 1983)), neural network approaches to problems in option markets have drawn attention (Malliaris and Salchenberger 1993, Anders et al. 1998, Wang et al. 2024, Farahani et al. 2024). Neural networks have been studied in a wide range of problems in option markets such as option pricing (Malliaris and Salchenberger 1993, Dugas et al. 2009, Wang et al. 2024), implied volatility estimation (Malliaris and Salchenberger 1996, Osterrieder et al. 2020, Zheng et al. 2021), hedging strategies (Shin and Ryu 2012, Chen and Sutcliffe 2012, Buehler et al. 2019), the optimal stopping problem (Kohler et al. 2010, Becker et al. 2019), and risk-neutral density estimation (Schittenkopf and Dorffner 2001). Moreover, various international options markets have been covered employing neural networks, including the KOSPI 200 in South Korea (Choi et al. 2004, Park et al. 2014), the S&P 500 in the United States (Yang et al. 2017, Zheng et al. 2021), the FTSE 100 in the United Kingdom (Zhang and Huang 2021), and the DAX 30 in Germany (Liu et al. 2019).

Although deep learning has been applied to a wide range of problems in options markets, the predominant focus in the literature has been on option pricing (Ruf and Wang 2020), while StatArb identification and exploitation in options markets has been addressed only as a secondary topic. For instance, Zapart (2003) used neural networks to predict volatility and subsequently priced options, but their model was not designed to identify arbitrage opportunities. Although Zapart (2003) sought to exploit pricing discrepancies through delta-hedged positions, their exposures to volatility, time, interest rates, and higher-order sensitivities to the underlying instrument indicate that they cannot be regarded as StatArbs. Horikawa and Nakagawa (2024), François et al. (2025) analyzed the difference between deep hedging and delta hedging in the context of StatArbs, but their main research focus was not on predicting and exploiting StatArbs, which is our topic.

Learning–based StatArb research focused on stock markets (e.g., Guijarro-Ordonez et al. (2022), Zhao et al. (2022)) with a few discussing options, while no studies in options markets aim to mainly address StatArbs. To bridge this gap, we explore StatArbs in an options market employing graph learning.

2.2.   Graph Neural Networks (GNNs)

GNNs are deep-learning methods for graph-structured data, where nodes represent entities and edges denote their relationships (Liu and Zhou 2022). Unlike convolutional neural networks designed for grid-structured data (e.g., images), GNNs capture intricate relational dependencies in non-Euclidean domains (Bhatti et al. 2023), such as social networks (Chen et al. 2022). Recently, Wang et al. (2024) employed a GNN to capture momentum spillover effects, where an asset's past return predicts those of related assets (Ali and Hirshleifer 2020), between options, and their GNN outperformed canonical neural networks. Motivated by their work, we tackle our StatArb problem by developing a GNN to extract market signals in the options market.

Various graph convolutions have been presented thus far (Wu et al. 2019). Our paper focuses on four representative GNN architectures—GCN (Kipf and Welling 2017), SAGE (Hamilton et al. 2017), GAT (Veličković et al. 2018), and GPS (Rampášek et al. 2022)—chosen for their historical significance, architectural diversity, and practical relevance. These methods are listed in PyTorch-Geometric (Fey and Lenssen 2019), which attests to their wide recognition and validation. In PyTorch-Geometric (Fey and Lenssen 2019), GCN, SAGE, and GAT are the most cited methods while GPS is one of the most recent methods. Accordingly, we adopt them as benchmarks to assess the effectiveness of our RNConv.

GCN (Kipf and Welling 2017) is a foundational GNN that utilizes a linear approximation of spectral graph convolutions on graph-structured data. SAGE (Hamilton et al. 2017) is an inductive framework that generates node embeddings by aggregating information from a node's local neighborhood. GAT (Veličković et al. 2018) employs an attention mechanism between each node and its neighbors to assign different importances to each neighbor during aggregation. GPS (Rampášek et al. 2022) combines global transformer-based attention and local message passing with positional and structural encodings. This hybrid architecture can extract both local and global information, showing state-of-the-art performance in their experiments. Together, these models mark the evolution toward more expressive and scalable graph representation learning.

However, these four methods——and prior GNN techniques, to our knowledge——overlook a key property in real-world settings: features are tabular in many cases. Tree-based models have been shown to outperform neural networks on numerous tabular datasets (Shwartz-Ziv and Armon 2022, Borisov et al. 2024, McElfresh et al. 2023, Rana et al. 2023, Yıldız and Kalayci 2025). In option pricing, Ivașcu (2021) found that tree-based methods (e.g., LightGBM (Ke et al. 2017), XGBoost (Chen and Guestrin 2016)) surpass both standard neural networks and classical parametric

models (Black and Scholes 1973, Corrado and Su 1996). Moreover, NODE (Popov et al. 2020), an ensemble of differentiable neural trees, outperformed methods based on non-differentiable trees such as CatBoost (Prokhorenkova et al. 2018) and XGBoost (Chen and Guestrin 2016) on some tabular datasets. Accordingly, we propose RNConv, a GNN that incorporates NODE (Popov et al. 2020) to more effectively leverage tabular node features.

3. Preliminaries

This section discusses NODE (Popov et al. 2020) and the Low-Rank Cross Network (Wang et al. 2021), which underpin the basis of our proposed method, RNConv.

3.1. Neural Oblivious Decision Ensembles (NODE)

NODE (Popov et al. 2020) integrates a decision tree ensemble into deep learning. It reformulates the oblivious decision trees (ODTs) (Kohavi 1994, Lou and Obukhov 2017) into differentiable ODTs, assembles them into a NODE layer, and stacks NODE layers.

An ODT is a binary decision tree where all nodes at the same depth share the same splitting criterion (Kohavi 1994, Lou and Obukhov 2017). Given an input $\mathbf{x} \in \mathbb{R}^p$, the output of a depth-$d$ ODT is defined as Kohavi (1994), Lou and Obukhov (2017):

$$h^{ODT}(\mathbf{x}) = R_{2-\mathbb{I}(\mathbf{s}_1^T \mathbf{x} - b_1), \ \dots, \ 2-\mathbb{I}(\mathbf{s}_d^T \mathbf{x} - b_d)} \in \mathbb{R} \tag{1}$$

where $R \in \mathbb{R}^{\overbrace{2 \times \cdots \times 2}^{d}}$ is a response tensor; $R_{i_1, i_2, \dots, i_d} \in \mathbb{R}$ indicates the entry of R at $(i_1, i_2, \dots, i_d)$; and $\mathbb{I}(\cdot)$ is a Heaviside step function. For all depth-i nodes, $\mathbf{s}_i \in \{0, 1\}^p$ is a one-hot vector (i.e., $\mathbf{1}^T \mathbf{s}_i = 1$, where $\mathbf{1}$ denotes the column vector of all ones used consistently throughout this paper) that selects the splitting feature, and $b_i \in \mathbb{R}$ is the threshold.

Popov et al. (2020) define a **d**ifferentiable oblivious **d**ecision **t**ree (DDT) $h^{DDT}(\cdot)$ by replacing $\mathbf{s}_i^T \mathbf{x}$ and $\mathbb{I}(x')$ in (1) with $\mathbf{x}^T entmax_\alpha(\hat{\mathbf{s}}_i)$ and a two-class entmax $\sigma_\alpha(x')$, respectively:

$$h^{DDT}(\mathbf{x}) = \sum_{i_1, \dots, i_d \in \{1,2\}^d} R_{i_1, \dots, i_d} \cdot C_{i_1, \dots, i_d}(\mathbf{x}) \tag{2}$$

where

$$C(\mathbf{x}) = \begin{bmatrix} c_1(\mathbf{x}) \\ 1 - c_1(\mathbf{x}) \end{bmatrix} \otimes \ldots \otimes \begin{bmatrix} c_d(\mathbf{x}) \\ 1 - c_d(\mathbf{x}) \end{bmatrix}, \tag{3}$$

$$c_i(\mathbf{x}) = \sigma_\alpha \left(\frac{\mathbf{x}^T entmax_\alpha(\hat{\mathbf{s}}_i) - b_i}{\kappa_i} \right), \tag{4}$$

$$\sigma_\alpha(x') = entmax_\alpha([x'\ 0]^T). \tag{5}$$

The α-entmax transformation, $entmax_\alpha$ (Peters et al. 2019), is a sparsity-inducing activation function that generalizes softmax and sparsemax. Popov et al. (2020) set α to 1.5. The parameters $R \in \mathbb{R}^{\overbrace{2 \times \cdots \times 2}^{d}}, \hat{s}_i \in \mathbb{R}^p, b_i \in \mathbb{R}, \kappa_i \in \mathbb{R}$ are learnable where κ_i is introduced for scaling. The operator $\otimes$ denotes the outer product and constructs the response selection tensor $C(\mathbf{x}) \in \mathbb{R}^{\overbrace{2 \times \cdots \times 2}^{d}}$. For example, if $c_1(\mathbf{x}) = 1$ (i.e., $\mathbf{s}_1^T \mathbf{x} - b_1 > 0$), then $C_{2,i_2,\ldots,i_d}(\mathbf{x}) = 0$, while $C_{1,i_2,\ldots,i_d}(\mathbf{x})$ can be nonzero, allowing only entries of the form $R_{1,i_2,\ldots,i_d}$ to be considered. Likewise, if $c_1(\mathbf{x}) = 1$ and $c_2(\mathbf{x}) = 0$, the output is further narrowed to $R_{1,2,\ldots,i_d}$. Repeating this process ultimately yields a single selected element from R.

Popov et al. (2020) define a NODE layer $g^{NL}(\mathbf{x}) \in \mathbb{R}^{1 \times n_{ddt}}$ as a stack of DDTs h_j^{DDT} for $j \in \{1, \ldots, n_{ddt}\}$. Formally,

$$g^{NL}(\mathbf{x}) = \begin{bmatrix} h_1^{DDT}(\mathbf{x}) & h_2^{DDT}(\mathbf{x}) & \ldots & h_{n_{ddt}}^{DDT}(\mathbf{x}) \end{bmatrix}. \tag{6}$$

Finally, NODE is defined with NODE layers g_l^{NL}, where $l \in \{1, \ldots, l_{NODE}\}$, following DenseNet (Huang et al. 2017). Its output is given by:

$$f^{NODE}(\mathbf{x}) = \sum_{l=1}^{l_{NODE}} g_l^{NL}(\mathbf{z}_l)\mathbf{1} \quad \in \mathbb{R} \tag{7}$$

where

$$\mathbf{z}_l = \begin{cases} \mathbf{x} & \text{if } l = 1 \\ \left[\mathbf{x}^T\ g_1^{NL}(\mathbf{z}_1)\ \ldots\ g_{l-1}^{NL}(\mathbf{z}_{l-1}) \right]^T & \text{if } l > 1, \end{cases} \tag{8}$$

and $\mathbf{z}_l \in \mathbb{R}^{p+(l-1)n_{odt}}$. Each NODE layer receives the original input and all previous layer outputs, and the final output is the sum of all layer outputs.

3.2. Low-Rank Cross Network

The Low-Rank Cross Network Wang et al. (2021), addresses the limitation observed by Beutel et al. (2018), Wang et al. (2017) that deep neural networks are inefficient at learning feature interactions. Wang et al. (2021) define the $(l+1)$-th layer of a cross network given an input $\mathbf{x} = \mathbf{x}_0 \in \mathbb{R}^p$ as:

$$\mathbf{x}_{l+1} = \mathbf{x} \odot (W_l \mathbf{x}_l + \mathbf{b}_l) + \mathbf{x}_l \tag{9}$$

where $W_l \in \mathbb{R}^{p \times p}$ is learnable, and $\odot$ is element-wise multiplication. Since $\mathbf{x}$ and $\mathbf{x}_l$ are multiplied in (9), the network captures feature interactions. By applying a low-rank technique, Wang et al. (2021) derived the Low-Rank Cross Network, whose $(l+1)$-th layer is given by:

$$\mathbf{x}_{l+1} = \mathbf{x}_0 \odot \left(U_l^{CN} \cdot \phi(C_l^{CN} \cdot \phi(V_l^{CN^T} \mathbf{x}_l)) + \mathbf{b}_l \right) + \mathbf{x}_l \tag{10}$$

where ϕ is an activation function, and $U_l^{CN}, V_l^{CN} \in \mathbb{R}^{p \times p_{CN}}, C_l^{CN} \in \mathbb{R}^{p_{CN} \times p_{CL}}, \mathbf{b}_l \in \mathbb{R}^p$ are learnable.

4. Setup and Notation

We delineate the overarching setup adopted throughout the study, encompassing the investment environment, asset types, notation, prediction dataset structure, and underlying assumptions.

4.1. Investment Environment

Time Horizon. We assume there exists a bijective mapping from all trading dates to $\mathbb{N}$ that preserves temporal order. We map each time point on each trading date to $\mathcal{T} = [1, \infty)$ by assigning time point $HH{:}MM{:}SS$ on the t-th trading date ($t \in \mathbb{N}$) to $t + \frac{HH \times 60^2 + MM \times 60 + SS}{24 \times 60^2} \in \mathcal{T}$. Thus, $t \in \mathbb{N}$ and $\tau \in \mathcal{T}$ denote a trading date and a time point, respectively. We use $\lfloor \tau \rfloor \in \mathbb{N}$ to refer to the trading date corresponding to $\tau \in \mathcal{T}$, where $\lfloor \cdot \rfloor$ denotes the floor function. We define $o(t)$ as a function that maps each $t \in \mathbb{N}$ to a value in $[t, t+1) \subset \mathcal{T}$ corresponding to the market open time point on the trading date t. For example, if the market opens at 08:30:00 on trading date t, then $o(t) = t + \frac{8 \times 60^2 + 30 \times 60}{24 \times 60^2}$.

Assets. An asset a is defined as a structured tuple $(c; p)$, where the semicolon separates the asset type c from the parameter tuple p. For example, $(PT; M, K)$ is a put option PT with maturity M and strike price K, where $c = PT$ and $p = (M, K)$. In this paper, we only consider asset types from the set $C = \{UI, PT, CL, SL, LS, SA, RF\}$, where:

- UI: **u**nderlying **i**nstrument,
- PT: European **p**u**t** option,
- CL: European **c**a**l**l option,
- SL: **s**ynthetic **l**ong (see below for details),

- *LS*: synthetic **l**ong-**s**hort, as defined in Section 5.3,
- *SA*: synthetic long-**s**hort **a**rbitrage, as defined in Section 5.3, and
- *RF*: **r**isk-**f**ree asset.

For $c \in C$, let $\mathcal{A}(c) = \{a \mid a = (c; p)\}$ denote the set of all assets of type c. The set of all assets is given by $\mathcal{A} = \bigcup_{c \in C} \mathcal{A}(c)$. We denote by $\mathcal{A}_t(c)$ the set of all c-type assets listed in the market on $t \in \mathbb{N}$ (if an asset is synthetic, its listing is defined as the listing of all its constituent assets). We define $\mathcal{L}_t(c) \subset \mathcal{A}_t(c)$ as the subset of c-type assets that are traded by other investors on $t \in \mathbb{N}$ and that do not expire before the market closes on $t + 1$. Finally, we define $\mathcal{U}_t \subset \mathcal{A}$ as the trading universe for $t \in \mathbb{N}$, i.e., the set of all assets eligible to be invested by the agent on t. The agent herein is any entity employing the proposed method.

Pricing of Assets. For an asset $a \in \mathcal{A}$, we denote its market price at $\tau \in \mathcal{T}$ by $P_\tau(a) \in \mathbb{R}$. A present value function Π_τ is a linear functional mapping a cash flow to a real number that represents its value at $\tau \in \mathcal{T}$ and exists under the AF assumption (see Skiadas (2024)). In this paper, we consider only cash flows, each comprising a single flow at a single time point, as inputs to Π_τ. Hence, for simplicity, we denote the present value function as a function $\Pi_\tau(K; M)$ from a cash flow value K at $M \in \mathcal{T}$ to a real number. Since Π_τ is a linear functional, we have

$$\Pi_\tau(\lambda K + \lambda' K'; M) = \lambda \Pi_\tau(K; M) + \lambda' \Pi_\tau(K'; M) \tag{11}$$

for all $\lambda, \lambda' \in \mathbb{R}$. If the risk-free interest rate is assumed to be a constant, we have

$$\Pi_\tau(K; M) = K e^{-r_f(M-\tau)}, \tag{12}$$

where $r_f \in \mathbb{R}$ is a constant risk-free interest rate with continuous compounding (Hull et al. 2013).

Underlying Instrument. An option is defined on an underlying instrument, such as a stock. In this paper, we consider a single underlying instrument, whose price at $\tau \in \mathcal{T}$ is denoted by S_τ.

Put and Call Options. A European put (call) option is the right to sell (buy) an underlying instrument UI at a strike price K upon maturity $M \in \mathcal{T}$. A put or call option is fully specified by parameters UI, M, and K. As we focus on a single underlying instrument, we omit UI for simplicity. Thus, we write put and call options as $(PT; M, K)$ and $(CL; M, K)$, respectively. An option yields a payoff only at maturity M: $\max\{K - S_M, 0\}$ for a put and $\max\{S_M - K, 0\}$ for a call; it is zero at all other times.

We now formalize notations for maturities and strike prices. We define $\mathcal{M} \subset \mathcal{T}$ and $\mathcal{K} \subset \mathbb{R}$ as the sets of all maturities and strike prices, respectively. For $c \in \{PT, CL, SL\}$, for $a \in \mathcal{A}(c)$,

we let M_a and K_a denote its maturity and strike price, respectively. Then, for $t \in \mathbb{N}$, we define $\mathcal{L}_t(c; M) = \{a' \in \mathcal{L}_t(c) : M_{a'} = M\}$ as the set of all type-c assets in $\mathcal{L}_t(c)$ with maturity M. We define $\mathcal{M}_t(c) = \{M_{a'} : a' \in \mathcal{L}_t(c)\}$ as the set of maturities of all type-c assets in $\mathcal{L}_t(c)$.

Under the AF assumption, the put-call parity establishes a relationship between put and call option prices with the same maturity and strike price as follows: for any $M \in \mathcal{M}, K \in \mathcal{K}, \tau \in \mathcal{T}$,

$$P_\tau(PT; M, K) + S_t = P_\tau(CL; M, K) + \Pi_\tau(K; M). \tag{13}$$

Synthetic-Longs. A synthetic-long $(SL; M, K) \in \mathcal{A}(SL)$ is a synthetic asset consisting of one long contract of $(CL; M, K)$ and one short contract of $(PT; M, K)$. Its price at $\tau \in \mathcal{T}$ is

$$P_\tau(SL; M, K) = -P_\tau(PT; M, K) + P_\tau(CL; M, K). \tag{14}$$

Thus, we can rewrite the put-call parity (13) as:

$$S_\tau = P_\tau(SL; M, K) + \Pi_\tau(K; M). \tag{15}$$

Furthermore, $(SL; M, K)$ yields a payoff of

$$\text{Payoff}_\tau(SL; M, K) = \begin{cases} S_\tau - K & \text{if } \tau = M \\ 0 & \text{if } \tau \neq M \end{cases}. \tag{16}$$

In this paper, we only invest in synthetic-longs and their combinations (LS, SA).

4.2. Notation and Assumptions

Notations. For finite $X \subset \mathbb{R}$, we define $\min^k X$ as the k-th smallest element in X. For simplicity, we write $\min X := \min^1 X$. The notations $\max^k X$ and $\max X := \max^1 X$ are defined analogously. Given a function $f : X \to \mathbb{R}$, we define $\arg\min_x^k \{f(x) : x \in X\}$ as the set of elements in X for which $f(x)$ equals the k-th smallest value of $f(X)$. We similarly define $\arg\min_x := \arg\min_x^1$, as well as $\arg\max_x^k$ and $\arg\max_x$ analogously. Lastly, for $\mathcal{A}' \subset \mathcal{A}$ and vectors $\mathbf{x}_a \in \mathbb{R}^p$, $a \in \mathcal{A}'$, $[\mathbf{x}_a^T]_{a \in \mathcal{A}'} \in \mathbb{R}^{|\mathcal{A}'| \times p}$ denotes a row-stacked matrix ordered lexicographically.

We utilize two datasets, $\mathcal{D}^{tr} = \{D_t^{tr} : t \in \mathbb{N}\}$ and $\mathcal{D}^{ar} = \{D_t^{ar} : t \in \mathbb{N}\}$ with

$$D_t^{tr} = \left([\mathbf{x}_{tr,a,t-1}^T]_{a \in \mathcal{L}_{t-1}}, [\tilde{I}_{tr,a,t}]_{a \in \mathcal{L}_{t-1}} \right), \tag{17}$$

$$D_t^{ar} = \left([\mathbf{x}_{ar,a,t-1}^T]_{a \in \mathcal{U}_t}, G_{t-1}(p_{dg}), [y_{a,o(t)}]_{a \in \mathcal{U}_t} \right). \tag{18}$$

Each D_t^{tr} and D_t^{ar} is constructed to predict tradability (Section 5.1) and arbitrage (Section 5.2) on t, respectively, using information up to the date of $t-1$. Components $\mathbf{x}_{tr,a,t-1} \in \mathbb{R}^{p_{tr}}$, $\mathbf{x}_{ar,a,t-1} \in \mathbb{R}^{p_{ar}}$, $G_{t-1}(p_{dg})$, $\tilde{I}_{tr,a,t} \in \{0,1\}$, $y_{a,o(t)}$ will be specified in Sections 5 and 6.

For the prediction models, we repeatedly split the datasets $\mathcal{D}^{tr}$ and $\mathcal{D}^{ar}$ into train, validation, and test subsets. Let $\mathcal{T}_{fit} = \{t_1, ..., t_{n_{fit}}, t_{n_{fit}+1}\} \subset \mathbb{N} \cup \{\infty\}$ with $t_i < t_{i+1}$ and $t_{n_{fit}+1} = \infty$ denote the starting dates of test splits, where $n_{fit} \in \mathbb{N}$. Each index $i \in [1 : n_{fit}]$ defines a train-test round. Next, for each $k \in \{tr, ar\}$, we define the split time index sets $\mathcal{T}_{test,i}^k, \mathcal{T}_{train,i}^k, \mathcal{T}_{val,i}^k$ as

$$\mathcal{T}_{test,i}^k = [t_i : t_{i+1}), \tag{19}$$

$$\mathcal{T}_{train,i}^k \cup \mathcal{T}_{val,i}^k = [1 : t_i), \tag{20}$$

$$\mathcal{T}_{train,i}^k \cap \mathcal{T}_{val,i}^k = \emptyset, \tag{21}$$

where $\mathcal{T}_{val,i}^k$ is randomly sampled from $[1 : t_i)$ using a split ratio $p_{val} = |\mathcal{T}_{val,i}^k|/(|\mathcal{T}_{val,i}^k| + |\mathcal{T}_{train,i}^k|)$. Then, we define data splits as $\mathcal{D}_{train,i}^k = \{D_t^k : t \in \mathcal{T}_{train,i}^k\}$, $\mathcal{D}_{val,i}^k = \{D_t^k : t \in \mathcal{T}_{val,i}^k\}$, and $\mathcal{D}_{test,i}^k = \{D_t^k : t \in \mathcal{T}_{test,i}^k\}$. Lastly, we group them as $\mathcal{D}_i^k = (\mathcal{D}_{train,i}^k, \mathcal{D}_{val,i}^k, \mathcal{D}_{test,i}^k)$. Given $\mathcal{D}_i^k$, we train prediction models on $\mathcal{D}_{train,i}^k$, select the best model based on $\mathcal{D}_{val,i}^k$, and evaluate it on $\mathcal{D}_{test,i}^k$[1].
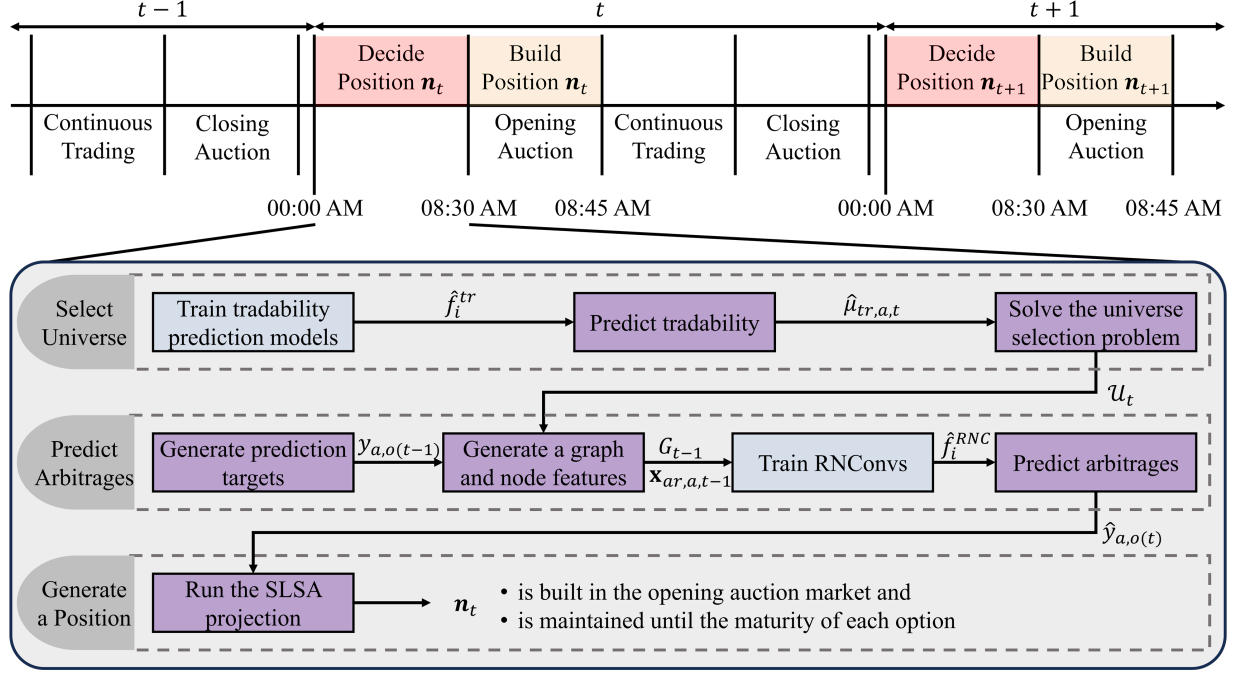
Assumptions. Throughout the paper, we assume the following:

- Positions may consist of any number of option contracts.
- The agent's trades have no impact on markets.
- Option margin requirements can be ignored due to sufficiently large long-term holdings of substitute assets, such as stocks, bonds, and funds.
- Short selling is permitted without limit.
- Unlimited borrowing and lending are allowed at the risk-free rate.
- There are no taxes or transaction costs.
- There are no restrictions on market orders during the opening market.

5. Proposed Method

The final goal of our method is to determine the position $\mathbf{n}_t = \left[n_{a,t}\right]_{a \in \mathcal{U}_t} \in \mathbb{R}^{|\mathcal{U}_t|}$ for each $t \in \mathbb{N}$, where each $n_{a,t}$ represents the number of contracts for asset a. For $t \in \mathbb{N}$, our method (i) determines

---

[1] These splits support proper model evaluation. Each $\mathcal{D}_i^k$ contains disjoint train, validation, and test splits. In addition, (19) ensures test datasets are non-overlapping and jointly cover $[t_1 : \infty)$: $\forall i \neq j$, $\mathcal{T}_{test,i}^k \cap \mathcal{T}_{test,j}^k = \emptyset$, and $\bigcup_{i=1}^{n_{fit}} \mathcal{T}_{test,i}^k = [t_1 : \infty)$. Moreover, since $t < t'$, $\forall t \in \mathcal{T}_{train,i}^k \cup \mathcal{T}_{val,i}^k$, $\forall t' \in \mathcal{T}_{test,i}^k$, the splits avoid look-ahead bias.

Figure 1   **Timeline.** The blue boxes are executed only on the dates that belong to $\mathcal{T}_{fit}$ and are skipped on all other days. On each $t \in [t_i : t_{i+1})$, the most recently trained models, $\hat{f}_i^{tr}$ and $\hat{f}_i^{RNC}$, are used.

the trading universe $\mathcal{U}_t$[2], (ii) predicts arbitrage opportunities $\hat{y}_{a,o(t)}$, and (iii) derives a low-risk position $\mathbf{n}_t$. The three tasks are conducted for every t, as depicted in Figure 1 based on times from the Korean market. The following sections elaborate on our approach for each task.

The determined position $\mathbf{n}_t$ is executed in the opening auction and is held until the expiration of each option included in $\mathbf{n}_t$. These processes are conducted on every trading date.

Note that the three tasks of the decision process for $\mathbf{n}_t$ rely solely on information available up to the date of $t-1$, excluding any data from t or beyond. This is because using information available after the decision process on t to determine $\mathbf{n}_t$ incurs look-ahead bias [3].

5.1. Trading Universe Selection Model

A universe selection method is necessary to ensure back-testing validity, practical feasibility, and proper graph construction. For example, deep ITM/OTM options are rarely traded and are often missing from historical data, leading to unavailable prices (Anand and Chakravarty 2007, Blasco et al. 2010). Thus, synthetic-longs involving such options cannot be evaluated.

To address such necessity, for $t \in \mathbb{N}$, we solve (22)–(28) to construct $\mathcal{U}_t \subset \mathcal{L}_{t-1}(SL)$, and on t, we only trade the synthetic-longs in $\mathcal{U}_t$. Since the trading status on t—used to define $\mathcal{L}_t(SL)$—is

---

[2] While the major contributions of our two-step approach are in (ii) and (iii), we begin with (i) to provide a foundation for their appropriate functioning.

[3] See Alonso (2025) and Ter Horst et al. (2001) for a detailed discussion of look-ahead bias.

unknown when $\mathcal{U}_t$ is determined, $\mathcal{U}_t$ is defined as a subset of $\mathcal{L}_{t-1}(SL)$, not $\mathcal{L}_t(SL)$. Moreover, arbitrages on t are predicted only for synthetic-longs traded on $t-1$ (Section 5.2).

For each $a \in \mathcal{L}_{t-1}(SL)$, $u_{a,t} \in \{0, 1\}$ indicates whether asset a is included in $\mathcal{U}_t$. The optimal solution $u_{a,t}^*$ yields universe $\mathcal{U}_t = \{a \in \mathcal{L}_{t-1}(SL) : u_{a,t}^* = 1\}$. Variable $v_{M,t}$ indicates whether synthetic-longs with maturity M are included in $\mathcal{U}_t$.

$$\max \sum_{a \in \mathcal{L}_{t-1}(SL)} \hat{\mu}_{tr,a,t} u_{a,t} \tag{22}$$

s.t.

$$\sum_{a \in \mathcal{L}_{t-1}(SL)} u_{a,t} = p_{univ}, \tag{23}$$

$$u_{N_{atm,t-1}(a),t} \geq u_{a,t},$$

$$\forall a \in \mathcal{L}_{t-1}(SL) \setminus ATM_{t-1}(SL), \tag{24}$$

$$2v_{M,t} \leq \sum_{a \in \mathcal{L}_{t-1}(SL;M)} u_{a,t} \leq |\mathcal{L}_{t-1}(SL;M)| \cdot v_{M,t},$$

$$\forall M \in \mathcal{M}_{t-1}(SL), \tag{25}$$

$$u_{a,t} + u_{a',t} \leq 1, \ \forall (a, a') \in N_{far,t-1}, \tag{26}$$

$$u_{a,t} \in \{0, 1\}, \ \forall a \in \mathcal{L}_{t-1}(SL), \tag{27}$$

$$v_{M,t} \in \{0, 1\}, \ \forall M \in \mathcal{M}_{t-1}(SL). \tag{28}$$

Objective function (22) aims to select $\mathcal{U}_t$ so as to maximize the number of synthetic longs traded by other investors on t, i.e., $\max_{\mathcal{U}_t \subset \mathcal{L}_{t-1}(SL)} |\mathcal{U}_t \cap \mathcal{L}_t(SL)|$. Since trading status on t is unknown when determining $\mathbf{n}_t$ (Figure 1), we define $I_{tr,a,t} \in \{0, 1\}$ as a Bernoulli random variable with $\mu_{tr,a,t}$ indicating whether asset a is traded on t and maximize $\mathbb{E} \sum_{a \in \mathcal{L}_{t-1}(SL)} I_{tr,a,t} u_{a,t}$ instead of the exact number. We assume there exists a function $f_i^{tr}(\mathbf{x}_{tr,a,t-1})$ such that $t \in [t_i : t_{i+1})$, $\mu_{tr,a,t} = f_i^{tr}(\mathbf{x}_{tr,a,t-1}) + \epsilon_{tr,a,t}$, $\mathbb{E}\epsilon_{tr,a,t} = 0$, and $\epsilon_{tr,a,t}$ and $\epsilon_{tr,a',t}$ are independent for all $a \neq a'$. Here, $\mathbf{x}_{tr,a,t-1}$ is a deterministic feature vector. Then, $\mathbb{E}I_{tr,a,t} = \mathbb{E}\mathbb{E}[I_{tr,a,t}|\mu_{tr,a,t}] = \mathbb{E}\mu_{tr,a,t} = f_i^{tr}(\mathbf{x}_{tr,a,t-1})$. We estimate f_i^{tr} and $\mu_{tr,a,t}$ as a machine-learning model $\hat{f}_i^{tr}$ and $\hat{\mu}_{tr,a,t} = \hat{f}_i^{tr}(\mathbf{x}_{tr,a,t-1})$, respectively, described later. Finally, we have $\mathbb{E} \sum_{a \in \mathcal{L}_{t-1}(SL)} I_{tr,a,t} u_{a,t} = \sum_{a \in \mathcal{L}_{t-1}(SL)} \mu_{tr,a,t} u_{a,t} \approx \sum_{a \in \mathcal{L}_{t-1}(SL)} \hat{\mu}_{tr,a,t} u_{a,t}$.

We train $\hat{f}_i^{tr}$ and predict $\hat{\mu}_{tr,a,t}$ using $\mathcal{D}_i^{tr}$. Multiple models are trained on $\mathcal{D}_{train,i}^{tr}$ using the target $\tilde{I}_{tr,a,t}$—a realization of $I_{tr,a,t}$. The best model $\hat{f}_i^{tr}$ is selected using $\mathcal{D}_{val,i}^{tr}$, and it generates predictions $\hat{\mu}_{tr,a,t} = \hat{f}_i^{tr}(\mathbf{x}_{tr,a,t-1})$ on $\mathcal{D}_{test,i}^{tr}$.

We next discuss the constraints. Constraint (23) ensures the cardinality of $\mathcal{U}_t$ becomes p_{univ}, which is a user-specified parameter. Constraint (24) ensures near-ATM assets are selected. Let $ATM_t(SL) = \{a : a \in \operatorname{argmin}_{(SL;M^*,K) \in \mathcal{L}_t} |S_t - K|, \ \forall M^* \in \mathcal{M}_t(SL)\}$ denote the set of all the nearest ATM assets for each maturity. For $a \in \mathcal{L}_t(SL)$, let $N_{atm,t}(a) = \operatorname{argmin}_{a' \in \mathcal{L}_t(SL)}\{|K_a - K_{a'}| : M_a = M_{a'}$ and $|S_t - K_{a'}| < |S_t - K_a|\}$ denote its same-maturity nearest neighbor closer to ATM. Then, an asset a is excluded from $\mathcal{U}_t$ if the universe excludes its same-maturity nearest neighbor closer to ATM. Since $ATM_{t-1}(SL)$ admit no neighbors closer to ATM, they are excluded in (24). Because we expect near-ATM options at $t-1$ are likely to be also traded on t, (24) is included.

Constraint (25) enforces that if any maturity-M option is selected, then at least one of the other maturity-M options must be selected. Since the positions (cf. Section 5.3) require both long and short positions on same-maturity options, isolated inclusion of a maturity-M asset renders it unusable. Therefore, the right inequality of (25) ensures that if a maturity-M asset is selected (i.e., $u_{a,t} = 1$ for some $a \in \mathcal{L}_{t-1}(SL; M)$), then the maturity M itself is selected (i.e., $v_{M,t} = 1$). Moreover, another maturity-M asset must be selected (i.e., $u_{a',t} = 1$ for some $a' \in \mathcal{L}_{t-1}(SL; M)$ with $a \neq a'$) by the left inequality of (25).

Constraint (26) chooses one of the two same-maturity neighborhood assets that are far away from each other. Let $N_{far,t} = \{(a, a') \in \mathcal{L}_t(SL) \times \mathcal{L}_t(SL) : M_a = M_{a'}, |K_a - K_{a'}| = \max\{\min_{a'' \in \mathcal{L}_t(SL;M_a)} |K_{a''} - K^*| : K^* \in \{K_a, K_{a'}\}\} > \Delta K_{max}\}$ denote the set of the pairs of the same-maturity assets of which strike price differences are large. Parameter ΔK_{max} quantifies the notion of being "far." Since a node in our prediction graphs (Section 5.2) is connected to a fixed number of nearest same-maturity neighbors, overly distant connections may arise without this constraint. To avoid such cases, (26) is imposed.

5.2. Arbitrage Prediction

5.2.1. Arbitrage Prediction Problem Statement
The arbitrage prediction problem addressed in this section is a node-level regression problem. Given a graph and its node features $\mathbf{x}_{ar,a,t-1}$, our goal is to predict the novel arbitrage opportunity $y_{a,o(t)}$ corresponding to each node $a \in \mathcal{U}_t$ using the new proposed architecture, RNConv. We next discuss these components.

Prediction Target. Our arbitrage prediction model aims to predict $y_{a,o(t)}$ for $t \in \mathbb{N}$, $a \in \mathcal{U}_t$. For all $\tau \in \mathcal{T}$, $a = (SL; M_a, K_a) \in \mathcal{U}_{\lfloor \tau \rfloor}$ [4], $y_{a,\tau}$ is defined as

$$y_{a,\tau} = \delta_{a,\tau} - \bar{\delta}_{M_a,\tau} \tag{29}$$

where

$$\delta_{a,\tau} = \frac{S_\tau - P_\tau(SL; M_a, K_a)}{K_a}, \tag{30}$$

$$\bar{\delta}_{M_a,\tau} = \frac{\sum_{a' \in \mathcal{U}_{\lfloor \tau \rfloor} \cap \mathcal{L}_{\lfloor \tau \rfloor - 1}(SL; M_a)} \delta_{a',\tau}}{|\mathcal{U}_{\lfloor \tau \rfloor} \cap \mathcal{L}_{\lfloor \tau \rfloor - 1}(SL; M_a)|}. \tag{31}$$

The price $P_\tau(SL; M_a, K_a)$ is computed from the put and call prices via (14), and $\bar{\delta}_{M_a,\tau}$ is the average of $\delta_{a',\tau}$ over the maturity-M_a assets in the universe. We designate the market open time $o(t)$ as the target time point for prediction (i.e., we predict $y_{a,o(t)}$ amongst many $y_{a,\tau}$).

Variable $\delta_{a,\tau}$ represents the price at τ of a zero-coupon bond maturing at M_a with face value 1. By (30), $\delta_{a,\tau}$ is the cost of building a position with $1/K_a$ units of the underlying instrument and $-1/K_a$ synthetic-long contracts. When the positions of $\delta_{a,\tau}$ are cleared at M_a, the resulting payoff is

$$\text{Payoff}_{M_a}(\delta_{a,\tau}) = \frac{S_{M_a}}{K_a} - \frac{S_{M_a} - K_a}{K_a} = 1, \tag{32}$$

which follows from (16). Hence, $\delta_{a,\tau}$ serves as the **d**iscounted price—or equivalently, the **d**iscount factor—of a zero-coupon bond with face value 1 and maturity M_a.

Arbitrage opportunities can be found in $\delta_{a,\tau}$'s. Under the AF assumption, $\delta_{a,\tau} = \delta_{a',\tau}$ should hold for any $a, a' \in \mathcal{L}_{\lfloor \tau \rfloor - 1}(SL; M)$, as both represent the same zero-coupon bond[5]. However, in reality, we have $\delta_{a,\tau} \neq \delta_{a',\tau}$ in many cases, as shown in Figure 2 since the blue and red curves should be zero if $\delta_{a,\tau} = \delta_{a',\tau}$ were true. Hence, arbitrage opportunities can be found in $\delta_{a,t}$'s.
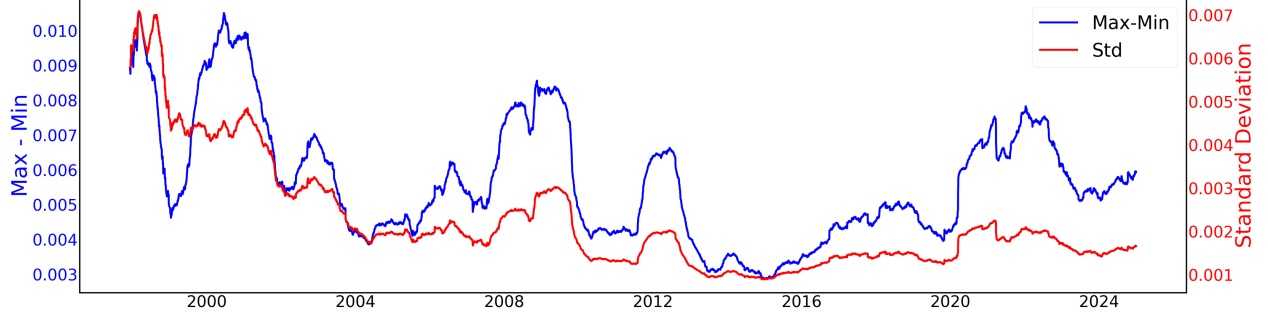
To isolate arbitrages embedded in $\delta_{a,\tau}$, we subtract the average $\bar{\delta}_{M_a,\tau}$ from $\delta_{a,\tau}$ and define $y_{a,\tau}$ as in (29). Predicting $\delta_{a,\tau}$ itself would conflate the bond price with arbitrages, as implied by (32). Thus, we remove the bond by subtracting $\bar{\delta}_{M_a,\tau}$, which serves as an estimate of the bond price.

If $y_{a,\tau} > 0$ were known in advance, arbitrages could be exploited by shorting $\delta_{a,\tau}$ and longing $\bar{\delta}_{M_a,\tau}$ at τ, yielding an arbitrage profit of $|y_{a,\tau}|$ at τ. The reverse would symmetrically yield an arbitrage profit. Hence, predicting (29) directly identifies pure arbitrages.

Proposition 1 shows that $y_{a,\tau}$ comprises pure arbitrages, and that $\bar{\delta}_{M_a,\tau}$ estimates the expected discount factor. This follows from the fact that, under the AF assumption, $y_{a,\tau}$ vanishes, and $\bar{\delta}_{M_a,\tau}$ becomes the present value of a unit zero-coupon bond maturing at M_a in Proposition 1.

---

[4] Here, the subscript $\lfloor \tau \rfloor \in \mathbb{N}$ in $\mathcal{U}_{\lfloor \tau \rfloor}$ implies the trading date corresponding to the time point τ, according to Section 4.1.

[5] Under the AF assumption, the law of one price holds (Skiadas 2024).

Figure 2   **Time-Series of Rolling 252-Trading-Day Avg. of Max-Min and Std. of $\delta_{a,t,o}$ over Nearest-Term, Near-ATM Options.** On each t, the range and standard deviation of $\delta_{a,t,o}$ are computed from the open price data of the KOSPI 200 index and near-ATM, nearest-maturity options ($S_{t,o}/K \in (0.95, 1.05)$). Then, the 252-day rolling averages are plotted.

PROPOSITION 1. *If the market is AF, then the following holds:*

$$y_{a,\tau} = 0, \tag{33}$$

$$\delta_{a,\tau} = \bar{\delta}_{M_a,\tau} = \Pi_\tau(1; M_a). \tag{34}$$

Proof   From (15) and (11), we obtain $S_\tau - P_\tau(SL; M_a, K_a) = K_a \cdot \Pi_\tau(1, M_a)$. Substituting this result into (30) yields $\delta_{a,\tau} = \Pi_\tau(1; M_a)$ for all $a \in \mathcal{U}_{\lfloor \tau \rfloor} \cap \mathcal{L}_{\lfloor \tau \rfloor - 1}(SL; M_a)$, establishing (34). Thus, (33) follows by substituting (34) into (29).   □

Graph Construction. To predict $y_{a,o(t)}$ for $a \in \mathcal{U}_t$ and $t \in \mathbb{N}$, a graph $G_{t-1}(p_{dg}) = (\mathcal{U}_t, L_{t-1}(p_{dg}))$ is considered where $\mathcal{U}_t$ is the node set[6], $L_{t-1}(p_{dg}) \subset \mathcal{U}_t \times \mathcal{U}_t$ is the edge set, and $p_{dg} \in [0, 1]$ is a hyperparameter. If $(a, a') \in L_{t-1}(p_{dg})$, one of the following is satisfied:

1. $M_a = M_{a'}$ and $K_a \in argmin_{a''}^{k'}\{|K_{a'} - K_{a''}| : a'' \in \mathcal{U}_t, M_{a''} = M_{a'}\}$ where k' is the nearest multiple of two to $p_{dg} \cdot |\{a'' \in \mathcal{U}_t : M_{a''} = M_{a'}\}|$ (set to one if the result is zero).

2. $K_a = K_{a'}$ and $M_a \in argmin_{a''}^{k'}\{|M_{a'} - M_{a''}| : a'' \in \mathcal{U}_t, K_{a''} = K_{a'}\}$ where k' is the nearest multiple of two to $p_{dg} \cdot |\{a'' \in \mathcal{U}_t : K_{a''} = K_{a'}\}|$ (set to one if the result is zero).

The first condition requires that a and a' share the same maturity, and that a is among the nearest-k' neighbors of a' by strike price. The second applies the same logic with strike price and maturity reversed. Since strike prices increase by a fixed increment (e.g., 2.5 for KOSPI 200 options), each option typically has two nearest same-maturity neighbors. Thus, k' is set as a multiple of two and applied analogously for maturity.

---

[6] The subscript t of $\mathcal{U}_t$ in $G_{t-1}(p_{dg})$ does not imply the use of future information at t to predict $y_{a,o(t)}$, as clarified in Section 5.1; it merely indicates that the assets in $\mathcal{U}_t$ are traded on t.

Node Features. For $t \in \mathbb{N}$, we use the following as node features $\mathbf{x}_{ar,a,t-1}$ to predict arbitrage $y_{a,o(t)}$ where $c(t), h(t), l(t) \in \mathcal{T}$ map $t \in \mathbb{N}$ into the values in $\mathcal{T}$ corresponding to the time points when the market closes, achieves high, and becomes low, respectively:

- Moneyness $(S_{c(t-1)} - K_a)$ and days remaining until M_a,

- $y_{a,c(t-1)}, y_{a,o(t-1)}$: the closing and opening values of y on $t - 1$,

- $\hat{y}_{a,h(t-1)}, \hat{y}_{a,l(t-1)}$: the estimated high and low values of y on $t - 1$. The high (low) estimate uses the high (low) put and low (high) call prices,

- $y_{a,c(t-1)} - y_{a,c(t-2)}$: the change in the closing target value between $t - 1$ and $t - 2$,

- $\hat{\sigma}^{PT}_{im,a,t-1}, \hat{\sigma}^{CL}_{im,a,t-1}$: the implied volatilities of the put and call options of a on $t - 1$.

5.2.2. Revised Neural Oblivious Decision Ensemble Graph Convolution (RNConv) We first present the Revised NODE (RNODE) layer, which underpins RNConv. Then, we propose the RNConv layer and the overall RNConv architecture.

RNODE Layer. To enhance NODE performance, we present the RNODE layer by replacing b_i, κ_i, and $entmax_\alpha(\hat{\mathbf{s}}_i)$ in (4) with batch normalization (Ioffe and Szegedy 2015), a low-rank cross network (Wang et al. 2021), and a multi-layer perceptron. Let d and n_{rdt} be hyperparameters denoting the depth and number of trees in an RNODE layer, respectively, and γ be the dichotomizing hyperparameter. Given an input $\mathbf{x} \in \mathbb{R}^p$, the output of **RNODE Layer** is defined as

$$g^{RNL}(\mathbf{x}) = \begin{bmatrix} h_{rdt,1}(\mathbf{x}) \ ... \ h_{rdt,n_{rdt}}(\mathbf{x}) \end{bmatrix} \in \mathbb{R}^{1 \times n_{rdt}} \tag{35}$$

where for $j \in \{1, 2, ..., n_{rdt}\}, k \in \{1, 2, .., d\}$,
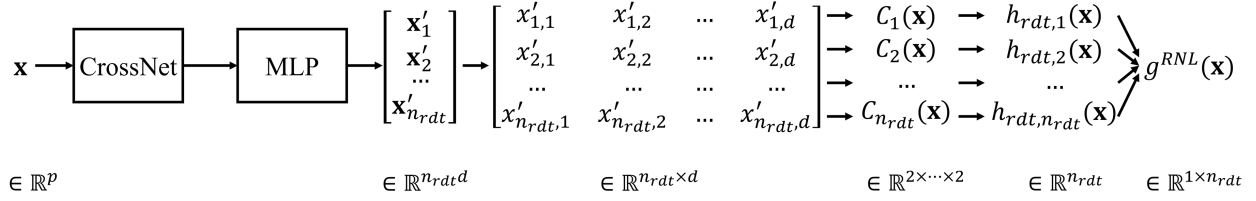
$$h_{rdt,j}(\mathbf{x}) = \sum_{i_1,...,i_d \in \{1,2\}^d} R_{j,i_1,...,i_d} \cdot C_{j,i_1,...,i_d}(\mathbf{x}) \in \mathbb{R}, \tag{36}$$

$$C_j(\mathbf{x}) = \begin{bmatrix} c_{j,1}(\mathbf{x}) \\ 1 - c_{j,1}(\mathbf{x}) \end{bmatrix} \otimes ... \otimes \begin{bmatrix} c_{j,d}(\mathbf{x}) \\ 1 - c_{j,d}(\mathbf{x}) \end{bmatrix}, \tag{37}$$

$$c_{j,k}(\mathbf{x}) = \sigma_\alpha(\gamma \cdot VBN(x'_{j,k})) \in \mathbb{R}, \tag{38}$$

$$\begin{bmatrix} \mathbf{x}'_1 \\ \mathbf{x}'_2 \\ ... \\ \mathbf{x}'_{n_{rdt}} \end{bmatrix} = MLP(CrossNet(\mathbf{x})) \in \mathbb{R}^{n_{rdt}d}, \tag{39}$$

$$\mathbf{x}'_j = [x'_{j,1} \ x'_{j,2} \ ... \ x'_{j,d}]^T \in \mathbb{R}^{d \times 1}. \tag{40}$$

Figure 3   Revised Neural Oblivious Ensemble (RNODE) Layer

We define $h_{rdt,j}(\mathbf{x})$ as a **r**evised **d**ifferentiable oblivious **t**ree. Equations (36) and (37) are derived from (2) and (3), but differ by including a tree index j, which accounts for multiple distinct RDTs. The function $\sigma_\alpha(\cdot)$ is defined in (5). Figure 3 illustrates an RNL layer.

RNODE differs from NODE by replacing (4) with (38)–(40). In (38), $VBN(\cdot)$ is vanilla batch normalization without learnable parameters, while $BN(\cdot)$ includes learnable parameters (Ioffe and Szegedy 2015). Extending the low-rank cross network (10) of Wang et al. (2021), we utilize a batch-normalized variant $CrossNet(\cdot)$ in (39) to enhance optimization stability, defined as

$$CrossNet(\mathbf{x}) = \mathbf{z}_{l_{CN}} \in \mathbb{R}^p \tag{41}$$

where for $l \in \{0, 1, ..., l_{CN} - 1\}$,

$$\mathbf{z}_l = \begin{cases} BN(\mathbf{x}) \in \mathbb{R}^p & \text{if } l = 0 \\ BN(\mathbf{x}_l) \in \mathbb{R}^p & \text{if } l \geq 1 \end{cases}, \tag{42}$$

$$\mathbf{x}_{l+1} = \mathbf{x} \odot \left(U_l^{CN} \phi(C_l^{CN} \phi(V_l^{CN^T} \mathbf{z}_l)) + \mathbf{b}_l^{CN} \right) + \mathbf{z}_l. \tag{43}$$

Function $MLP(\cdot)$ is a multilayer perceptron of which last layer does not have a bias and activation function. Since the output of $MLP(\cdot)$ passes through $VBN(\cdot)$, the last layer's bias becomes meaningless, so it is removed.

To encourage balanced branching, we use $VBN(\cdot)$ in (38). In NODE (Popov et al. 2020), b_i and κ_i determines the branching of a DODT. However, depending on the training process, $|b_i|$ can grow arbitrarily large, causing imbalanced branching. Likewise, if $|\kappa_i|$ becomes large, $c_i(\mathbf{x})$ in (4) becomes nearly identical across all inputs, rendering branching meaningless. To address these issues, RNODE replaces b_i and κ_i with the batch mean and standard deviation within $VBN(\cdot)$. As the batch mean is likely to lie near the center of a batch, branching becomes balanced. Normalizing
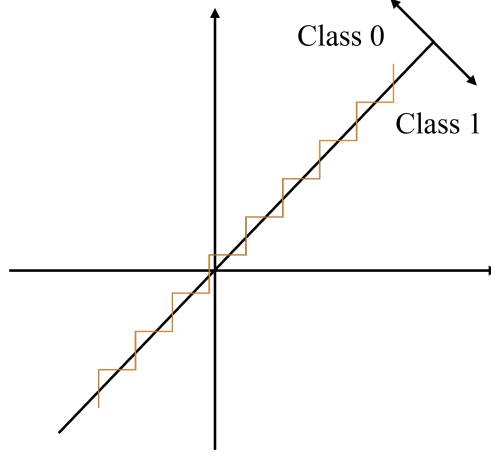
Figure 4   **Example of Inefficient Decision Boundary of Decision Tree Model**

by the batch standard deviation ensures unit variance in $VBN(x'_{l,i})$, which is then scaled by $\gamma \in \mathbb{R}$ to push $c_{j,i}$ toward 0 or 1, addressing the second issue[7].

To overcome the decision tree's limitation of having an axis-aligned decision boundary, in (39), we replace $entmax_\alpha(\hat{\mathbf{s}}_i)$ with $MLP(CrossNet(\cdot))$. In classical decision trees, where each split compares a single variable to a threshold, approximating the true boundary $x_1 - x_2 = 0$ requires many stair-shaped splits (orange in Figure 4). In contrast, allowing the direct use of $x_1 - x_2 > 0$ enables a single, more efficient split. However, NODE uses $\mathbf{x}^T entmax_\alpha(\hat{\mathbf{s}}_i) - b_i$ in (4) to mimic the single-variable criterion of classical decision trees. To address this inefficiency, we replace $\mathbf{x}^T entmax_\alpha(\hat{\mathbf{s}}_i) - b_i$ with $\mathbf{x}^T \hat{\mathbf{s}}_i - b_i$, which can learn decision boundaries like $x_1 - x_2 = 0$. We further extend this linear form to a multilayer perceptron and cross network, enabling the model to learn nonlinear decision boundaries.

RNConv Layer. Given a graph $G = (\mathcal{U}, L)$[8] and node feature vectors $\mathbf{x}_a \in \mathbb{R}^p$ for all $a \in \mathcal{U}$, we define an RNConv layer as

$$g^{RNC}(\mathbf{x}_a, G) = \frac{1}{2}\left(\mathbf{h}_1(\mathbf{x}_a) + \mathbf{h}_2(\mathbf{x}_a)\right) \in \mathbb{R}^{n_{rdt}} \tag{44}$$

---

[7] Note that too large γ may cause gradient saturation during training.

[8] Sets $\mathcal{U}$ and L denote the node and edge sets, respectively, previously written as $(\mathcal{U}_t, L_{t-1}(p_{dg}))$. When discussing RNConv, we omit the subscripts t, $t - 1$, and the argument p_{dg}, and simply write $G = (\mathcal{U}, L)$ instead of $G_{t-1}(p_{dg}) = (\mathcal{U}_t, L_{t-1}(p_{dg}))$ since RNConv can be applied to other problems.
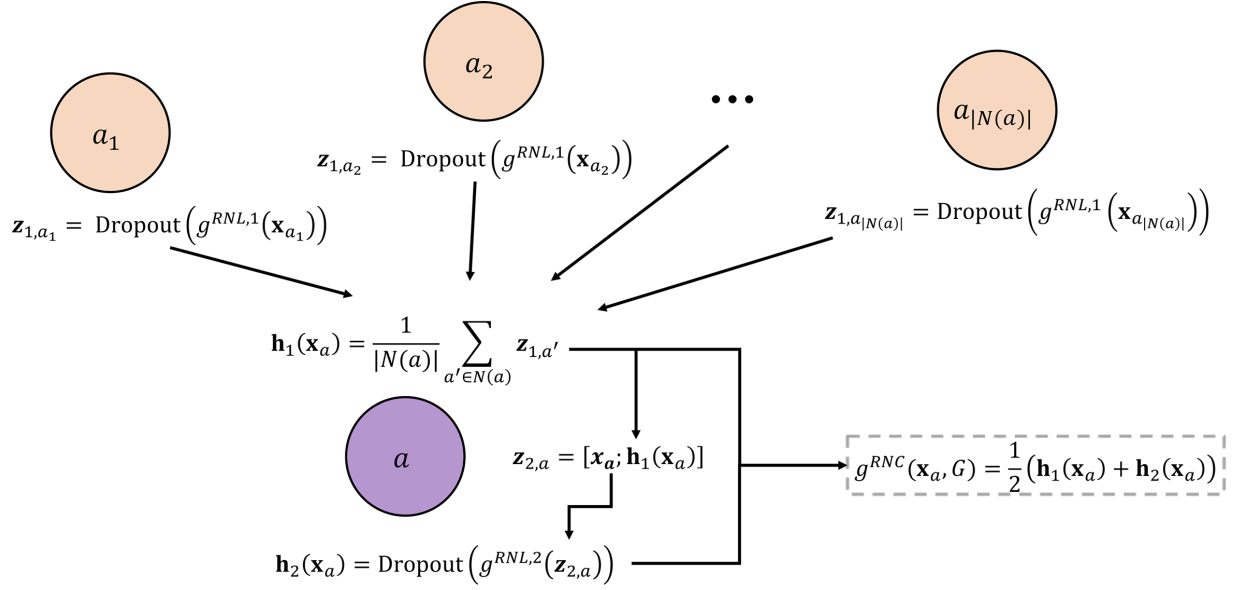
Figure 5   RNConv Layer

where

$$\mathbf{z}_{1,a} = \text{Dropout}\left(g^{RNL,1}(\mathbf{x}_a); q_1\right) \in \mathbb{R}^{n_{rdt}}, \tag{45}$$

$$\mathbf{h}_1(x_a) = \frac{1}{|N(a)|} \sum_{a' \in N(a)} \mathbf{z}_{1,a'} \in \mathbb{R}^{n_{rdt}}, \tag{46}$$

$$\mathbf{z}_{2,a} = [\mathbf{x}_a; \mathbf{h}_1(\mathbf{x}_a)] \in \mathbb{R}^{p+n_{rdt}}, \tag{47}$$

$$\mathbf{h}_2(\mathbf{x}_a) = \text{Dropout}\left(g^{RNL,2}\left(\mathbf{z}_{2,a}\right); q_1\right) \in \mathbb{R}^{n_{rdt}}, \tag{48}$$

$$N(a) = \{a' : (a', a) \in L\}. \tag{49}$$

The architecture is visualized in Figure 5. We first compute $\mathbf{z}_{1,a}$'s for all $a \in \mathcal{U}$ in (45). By using Dropout, different subsets of trees in $g^{RNL,1}$ learn from different samples, enhancing tree diversity. Next, we aggregate the neighborhood's output by averaging $\mathbf{z}_{1,a'}$ in (46). Then, we concatenate the input $\mathbf{x}_a$ and aggregated vector $\mathbf{h}_1(\mathbf{x}_a)$ in (47), and pass the result into the second RNODE layer that extracts node-level information using both local and aggregated context in (48). As in (45), we apply Dropout in (48) to promote tree diversity. Finally, the output in (44) is the average of the neighborhood information ($\mathbf{h}_1$) and the node's own representation ($\mathbf{h}_2$).

RNConv. We define RNConv as

$$f^{RNC}(\mathbf{x}_a, G) = \frac{1}{n_{rdt}l_{RNC}} \sum_{l=1}^{l_{RNC}} g_l^{RNC}(\mathbf{e}_{l,a}, G)\mathbf{1} \in \mathbb{R} \tag{50}$$

where

$$\mathbf{e}_{l,a} = \begin{cases} \text{Dropout}(\mathbf{x}_a; q_2) \in \mathbb{R}^p \text{ if } l = 0 \\ \left[\mathbf{x}_a; \frac{1}{l-1} \sum_{i=1}^{l-1} g_i^{RNC}(\mathbf{e}_{i,a})\right] \in \mathbb{R}^{p+n_{rdt}} \text{ o.w.} \end{cases} \tag{51}$$

Analogous to NODE in (7), we take the average over all RNODE layer outputs in (50). In (51), the average of the interim outputs is fed into the next NODE layer, instead of concatenation as in (8), to avoid excessive parameter growth. Dropout is applied to enhance tree diversity.

5.3. Synthetic-Long-Short Arbitrage

In this section, we propose SLSA, a class of synthetic-long positions designed to tackle the problem of exploiting StatArbs using the predictions from Section 5.2. We begin by presenting the synthetic-long-short position, which serves as the foundation for SLSA. We then propose SLSA and establish its minimal theoretical risk under the AF assumption. Lastly, we introduce the SLSA projection, which maps predictions into tradable SLSA positions capable of realizing StatArbs. Each position is executed at market open and held to maturity, as described in Section 5, yielding profits upon inception without intermediate and final cash flows.

Synthetic-Long-Short. We define a synthetic-long-short $(LS; U, \mathbf{n})$, as a synthetic asset consisting of $m \in \mathbb{R}$ units of the underlying instrument and $n_a \in \mathbb{R}$ contracts of $a \in U$, satisfying

$$m + \sum_{a \in U} n_a = 0 \tag{52}$$

where U is a subset of $\mathcal{U}_t$, and $\mathbf{n} = [n_a]_{a \in U} \in \mathbb{R}^{|U|}$ is a row-stacked vector ordered lexicographically. Negative n_a means a short position of $|n_a|$ contracts on a, which similarly applies to m. Since m becomes automatically determined if $\mathbf{n}$ is specified in (52), m is not required as a parameter. The time-$\tau \in \mathcal{T}(U)$ price of $(LS; U, \mathbf{n})$ is

$$P_\tau(LS; U, \mathbf{n})$$
$$= mS_\tau + \sum_{a \in U} n_a P_\tau(SL; M_a, K_a) \tag{53}$$
$$= mS_\tau + \sum_{a \in U} n_a \left(S_\tau - K_a \delta_{a,\tau}\right) \tag{54}$$
$$= \sum_{a \in U} -n_a K_a \left(y_{a,\tau} + \bar{\delta}_{M_a,\tau}\right) \tag{55}$$

where $\mathcal{T}(U) = \left\{ \tau' : \{a \in U : n_a \neq 0\} \subset \mathcal{A}_{\lfloor \tau' \rfloor}(SL) \right\}$ represents the set of time points when all nonzero-position assets of U are listed and unexpired. Its definition and (30) yield (53) and (54), respectively. We obtain (55) from (29) and (52). Since $P_\tau(LS; U, \mathbf{n})$ includes both $y_{a,\tau}$ and $\bar{\delta}_{M_a,\tau}$, $(LS; U, \mathbf{n})$ can contain both arbitrages and zero-coupon bonds, as discussed in Proposition 1.

Synthetic long-short positions possess desirable properties in that a position in $(LS; U, \mathbf{n})$ eliminates risks related to both the price and volatility of the underlying instrument and results in zero price variance, as demonstrated in Proposition 2. Although the two assumptions in Proposition 2 may not strictly hold in practice, several well-known pricing models—such as the Black and Scholes (1973) model and the Binomial pricing model (Cox et al. 1979)—are derived under these two assumptions. Thus, under the two assumptions, synthetic long-shorts can have equal or lower risk than positions based on such well-known models when they have the same time-related risk level.

PROPOSITION 2. *If there is no arbitrage in the market and the risk-free interest rate is constant, then the following statements hold for all $\tau \in \mathcal{T}(U)$:*

(i) each $(LS; U, \mathbf{n})$ is neutral with respect to both the price and volatility of its underlying instrument,

(ii) $Var(P_\tau(LS; U, \boldsymbol{n})) = 0$.

Proof　Let τ be an arbitrary element of $\mathcal{T}(U)$. Then, the time-τ price of $(LS; U, \mathbf{n})$ becomes $P_\tau(LS; U, \mathbf{n}) = \sum_{a \in U} -n_a K_a \Pi_\tau(1; M_a) = \sum_{a \in U} -n_a K_a e^{-r_f(M_a - \tau)}$, where the first equality follows from (55) and Proposition 1, and the second from (12). Since n_a, K_a, r_f, M_a are constants, $P_\tau(LS; U, \mathbf{n})$ depends solely on τ. Thus, the process $\{P_\tau(LS; U, \mathbf{n})\}_{\tau \in \mathcal{T}(U)}$ is stochastically independent of the volatility and price processes of the underlying instrument over $\mathcal{T}(U)$, implying that $P_\tau(LS; U, \mathbf{n})$ is neutral to both. As $P_\tau(LS; U, \mathbf{n})$ is deterministic, $Var(P_\tau(LS; U, \mathbf{n})) = 0$. □

However, even under the assumptions in Proposition 2, $(LS; U, \mathbf{n})$ can still be exposed to time-related risk. Moreover, if the risk-free interest rate is not constant, as in Vasicek (1977), Proposition 2 no longer holds. Furthermore, synthetic-long-short positions can contain both a unit zero-coupon bond and arbitrage opportunities, but our goal is to exclusively exploit pure arbitrage opportunities. To address these limitations, we propose **S**ynthetic-**L**ong-**S**hort-**A**rbitrage (SLSA).

Synthetic-Long-Short-Arbitrage (SLSA) We define an SLSA $(SA; U, \mathbf{n})$ as a synthetic-long-short satisfying

$$\sum_{a \in U: M_a = M} n_a = 0, \ \forall M \in \mathcal{M}(U), \tag{56}$$

$$\sum_{a \in U: M_a = M} K_a \cdot n_a = 0, \ \forall M \in \mathcal{M}(U) \tag{57}$$

where $\mathcal{M}(U) = \{M_a : a \in U\}$ denotes the set of maturities of all assets in U. Because SLSA is a synthetic-long-short, its $\mathbf{n}$ also satisfies (52). Due to (52) and (56), we have $m = 0$ for every SLSA. For all $\tau \in \mathcal{T}(U)$, the time-τ price of a SLSA is

$$P_\tau(SA; U, \mathbf{n})$$

$$= \sum_{M \in \mathcal{M}(U)} \left(-\bar{\delta}_{M,\tau} \sum_{a \in U: M_a = M} n_a K_a \right) - \sum_{a \in U} n_a K_a y_{a,\tau} \tag{58}$$

$$= \sum_{a \in U} -n_a K_a y_{a,\tau}, \tag{59}$$

where from (55) and the definition of $\mathcal{M}(U)$, (58) follows. Its first term vanishes by (57), yielding (59).

For $t \in \mathbb{N}$, if each component option position of $(SA; U, \mathbf{n})$ is built at $o(t)$ and held until its respective expiration date, as outlined in the introduction of Section 5, then the payoff is as follows:

$$\text{Payoff}_\tau(SA; U, \mathbf{n}) = \begin{cases} \sum_{a \in U} n_a v_{a,t} & \text{if } \tau = o(t) \\ 0 & \text{if } \tau \neq o(t) \end{cases} \tag{60}$$

where $v_{a,t} = K_a y_{a,o(t)}$. At $o(t)$, we pay $P_{o(t)}(SA; U, \mathbf{n})$ for the component options, as derived in (59). For each $M \in \mathcal{M}(U)$, only the options $a \in \{a' \in U : M_{a'} = M\}$ maturing at M yield payoff. Thus, the payoff at M is $\sum_{a \in U: M_a = M} n_a(S_M - K) = (S_M - K) \sum_{a \in U: M_a = M} n_a = 0$, by (16) and (56). No payoff occurs at the other time points. Thus, the only nonzero net cash flow of $(SA, U, \mathbf{n})$ is $-P_{o(t)}(SA; U, \mathbf{n})$ at the initial time point $o(t)$. Accordingly, allocating large n_a to a with large $v_{a,t}$ can increase profit.

Compared to the synthetic-long-shorts that can contain bonds as shown in (55), SLSA solely comprises arbitrage opportunities since $P_\tau(SA; U, \mathbf{n})$ is a linear combination of arbitrages $y_{a,\tau}$ with the fixed coefficients of $-n_a K_a$ in (59). Moreover, if the market is AF, the price of $(SA; U, \mathbf{n})$ vanishes, as shown by the proposition below. Hence, we confirm arbitrage opportunities are exclusively contained in SLSA.

PROPOSITION 3. *Under the AF assumption, the following statements hold for all $\tau \in \mathcal{T}(U)$:*

(i) $P_\tau(SA; U, \mathbf{n}) = 0$,

(ii) $Var(P_\tau(SA; U, \mathbf{n})) = 0$,

(iii) every SLSA is neutral with respect to all Black and Scholes (1973) risk factors (i.e., the risk-free interest rate, time to maturity, and the price and volatility of its underlying instrument).

Proof Proposition 1 and (59) yield $P_\tau(SA; U, \mathbf{n}) = 0$ for all $\tau \in \mathcal{T}(U)$. Thus, process $\{P_\tau(SA; U, \mathbf{n})\}_{\tau \in \mathcal{T}(U)}$ is stochastically independent of all Black and Scholes (1973) risk factors. Hence, every SLSA is neutral to all Black and Scholes (1973) risk factors. As $P_\tau(SA; U, \mathbf{n})$ is a constant, its variance is zero for all $\tau \in \mathcal{T}(U)$. $\square$

SLSA exhibits the lowest risk level among the trading strategies developed in the past under the AF assumption since SLSA has zero variance and is neutral with respect to all the Black and Scholes (1973) risk factors, as shown in Proposition 3. Consequently, SLSA solves the problem of exposure to time in Proposition 2. In addition, Proposition 3 does not assume a constant risk-free interest rate, compared to Proposition 2. SLSA resolves the limitations of the synthetic-long-short.

Although the AF assumption in Proposition 3 may fail to hold in reality, the AF assumption is fundamental to rational decision-making in economics (Nau and McCardle 1992). Moreover, as arbitrage precludes the existence of a present-value function (Skiadas 2024), trading strategies derived in the extensive literature employing the present-value function assume an AF market. While real-world markets may contain arbitrages, they are expected to gravitate toward an AF state (Varian 1987, Langenohl 2018), rendering SLSA relatively low-risk.

In light of the desirable properties established in Proposition 3, we adopt SLSA for our trading positions. We next detail the methodology for determining $\mathbf{n}$ in $(SA; U, \mathbf{n})$.

Synthetic-Long-Short-Arbitrage Projection. For $t \in \mathbb{N}$, given the predicted values $\hat{y}_{a,o(t)}$ for $y_{a,o(t)}$ defined in (29), we construct our position of a SLSA $(SA; \mathcal{U}_t, \mathbf{n}_t)$ as follows:

$$\mathbf{n}_t = \text{Proj}_{\text{Null}(A_t)} \hat{\mathbf{v}}_t = N_t(N_t^T N_t)^{-1} N_t^T \hat{\mathbf{v}}_t \tag{61}$$

where $\mathbf{n}_t = [n_{a,t}]_{a \in \mathcal{U}_t} \in \mathbb{R}^{|\mathcal{U}_t|}$. The matrix A_t represents the constraints in (56) and (57), and N_t consists of orthonormal columns spanning $\text{Null}(A_t)$. Then, the projection matrix is $N_t(N_t^T N_t)^{-1} N_t^T$. A row-stacked vector $\hat{\mathbf{v}}_t = [\hat{v}_{a,t}]_{a \in \mathcal{U}_t} \in \mathbb{R}^{|\mathcal{U}_t|}$ consists of predicted values $\hat{v}_{a,t} \in \mathbb{R}$ for $v_{a,t}$, computed as $\hat{v}_{a,t} = K_a \hat{y}_{a,o(t)}$, where $\hat{y}_{a,o(t)} \in \mathbb{R}$ is the prediction for $y_{a,o(t)}$ defined in (29).

To construct an SLSA position satisfying (56) and (57) while maximizing expected profit with consideration of prediction risk, we formulate our position as in (61). Since $\mathbf{n}_t$ is the projection onto $\text{Null}(A_t)$, (56) and (57) are automatically satisfied. As shown in (60), profit increases with greater allocation to assets a with larger $v_{a,t}$. We thus define the initial (non-SLSA) position as $\hat{\mathbf{v}}_t$, assigning positions in proportion to the predicted values $\hat{v}_{a,t}$. Since these positions are based on predicted values $\hat{v}_{a,t}$ rather than the ground-truth $v_{a,t}$, we avoid concentrating our positions in assets with the most extreme predicted payoffs. That is, we mitigate prediction risk arising from prediction errors by avoiding concentration in such extreme assets. Lastly, since projection matrices are positive semidefinite, (61) ensures a non-negative aggregate predicted payoff, i.e., $\hat{\mathbf{v}}_t^T \mathbf{n}_t \geq 0$.

6. Experiments

In this section, we show that RNConv statistically significantly outperforms benchmarks, and that the P&L of SLSA projection positions exhibits a consistently upward trend. We begin by delineating the experimental setup, followed by the results of universe selection, RNConv, and SLSA.

6.1. Experimental Setup

We utilized historical data on KOSPI 200 and its options spanning from July 7th, 1997 to December 30th, 2024, which are available at Korea Exchange (2025b,a). The data include detailed option information (e.g., option type, strike price) as well as historical time-series data of the close, open, high, and low prices for each option and for the KOSPI 200 index itself. We selected the KOSPI 200 options primarily because the Korea Exchange—a reliable data provider—offers relatively long-term historical data free of charge. Following the data splits in Section 4.2, we set $t_{fit,1}$ as the first trading date of 2015 to ensure a few thousand training graphs, and we defined $\mathcal{T}_{fit} = \{t_{fit,1}, \ldots, t_{fit,n_{fit}}\}$ as the first trading dates of each quarter (i.e., $|\mathcal{T}_{fit}| = 40$), and set $p_{val} = 0.2$.

We implemented our method and conducted the experiments in `Python` and `R`, using `PyTorch`, `PyTorch-Geometric`, `scikit-learn`, and `lawstat` libraries, with the following values for the hyperparameters. We set $p_{dg} = 1/3$ so that, with at least three GNN layers, each node can aggregate information from all others. In (38), $VBN(\cdot)$ is implemented in `PyTorch` with `affine=False`. In (38), we set $\alpha = 1.5$ following (Popov et al. 2020), and $\gamma = 5$ to approximate a 20% probability that

$\sigma_\alpha(Z) \in [0.1, 0.9]$, assuming $Z \sim \mathcal{N}(0, 1)$ and applying Pareto's rule. In (39), MLP($\cdot$) has three layers, where the first two hidden layers have $p/4$ neurons, as in *CrossNet*. Leaky ReLU is used as the activation in both MLP($\cdot$) and CrossNet($\cdot$). We set $l_{CN} = 2$ in (41) and $p_{CN} = p/4$ in (43) and (10), following Wang et al. (2021). Finally, we set $q_1 = 0.5$ in (45) for the interim layers and $q_2 = 0.2$ in (51) for the input layer, following Srivastava et al. (2014).

6.2. Trading Universe Selection Model

We employed a radius neighbor classifier to predict $\hat{\mu}_{tr,a,t}$ in (22)–(28), using `scikit-learn` and scaled[9] features of moneyness ($S_{t-1,c}/K_a$) and time to maturity ($M_a - t$) in $\mathbf{x}_{tr,a,t-1}$. For each $\mathcal{D}_i^{tr}$ from Section 4.2, 50 radius neighbor classifiers were trained with radii linearly spaced between 0.0 and 0.1 (excluding 0.0). The best model $\hat{f}_i^{tr}$ was selected based on $\mathcal{D}_{val,i}^{tr}$ and predicted $\hat{\mu}_{tr,a,t}$ on $\mathcal{D}_{test,i}^{tr}$. If no neighbors exist within a radius, 5-nearest neighbors are used. We also evaluated the best model $\hat{f}_i^{tr}$ on $\mathcal{D}_{test,i}^{tr}$, which are summarized in Table 1.

We solved (22)–(28) for $p_{univ} \in \{16, 24, 32\}$ utilizing `PuLP` and `Gurobi`. When infeasible, the largest feasible p_{univ} was selected. For example, if (22)–(28) are infeasible for $p_{univ} = 24$, we gradually reduce p_{univ} (i.e., 23, 22, ...) until a feasible solution is found. The resulting universe cardinalities over time are shown in Figure 6. The KOSPI 200 options market was inaugurated in 1997. In its early years, option trading increased gradually, resulting in the upward trend in Figure 6. Subsequently, a sufficient number of options were traded for $p_{univ} \in \{16, 24\}$, while trading volume remained insufficient for $p_{univ} = 32$, as indicated by the fluctuations observed from 2016 to 2024.

Table 1   **Descriptive Statistics of Model Evaluation Metrics for Tradability Prediction (2015 Q1–2024 Q4)**

	Accuracy	Precision	Recall	F1	ROC AUC
Mean	76.939	77.597	97.867	86.531	83.803
Std	2.842	2.830	1.538	1.916	2.699
Min	71.601	71.424	93.401	82.890	78.602
Max	85.208	85.359	99.767	91.990	89.370

[9] Each feature was first transformed to follow a uniform distribution via the probability integral transformation, and then mapped to a standard normal distribution using the inverse cumulative distribution. The same scaling method was applied to the arbitrage prediction inputs and outputs.
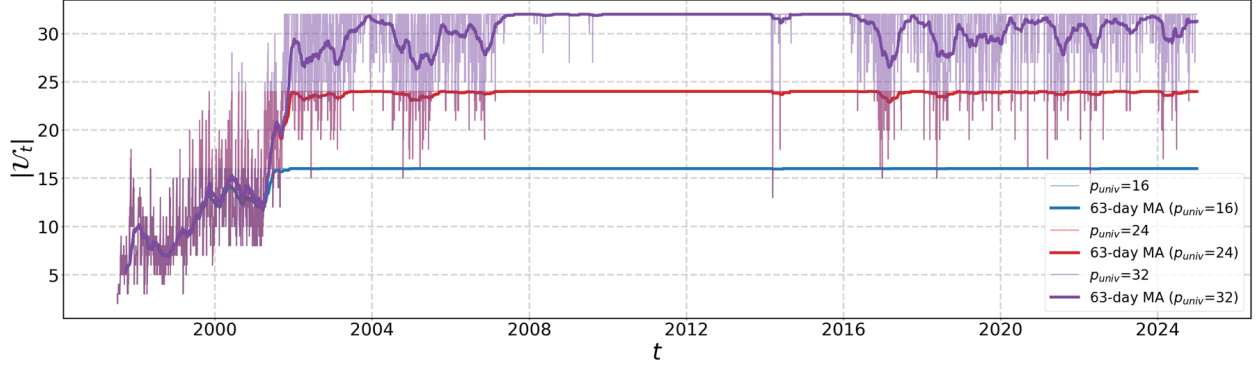
Figure 6   Time Series of $|\mathcal{U}_t|$. MA stands for the moving average.

6.3. Arbitrage Prediction

Benchmarks. Given a node feature $\mathbf{x} \in \mathbb{R}^p$ and graph G, we considered the following architecture for the benchmark convolution methods to evaluate the performance of RNConv:

$$f^{BM}(\mathbf{x}, G; \theta) = \text{Head}(\mathbf{z}_{l_{BM}}) \in \mathbb{R} \tag{62}$$

where for $l = 1, 2, ..., l_{BM}$,

$$\mathbf{z}_l = \text{LeakyReLU}\left(g_l^{BM}(\mathbf{z}_{l-1}, G; \theta_l)\right). \tag{63}$$

Function Head is a linear layer without activation, and g_l^{BM} denotes a benchmark graph convolution with parameters θ_l. We used GCN, GAT, SAGE, and GPS as instances of g_l^{BM}, implemented using `GCNConv`, `GATConv`, `SAGEConv`, and `GPSConv` from the `PyTorch-Geometric` library.[10]

Since deep learning model performance depends on the number of layers and parameters, we performed a grid search over these hyperparameters using $\mathcal{D}^{ar}_{train,i}, \mathcal{D}^{ar}_{val,i}$ in each period indexed by $t_i \in \mathcal{T}_{fit}$ for each p_{univ} and for each graph convolution architecture. In this way, we obtained the best model $\hat{f}^m_i$ for each i, p_{univ}, m. Lastly, we computed $\hat{y}_{a,t}$ and the mean squared error (MSE) of $\hat{f}^m_i$ on $\mathcal{D}^{ar}_{test,i}$ for each i, p_{univ}, m.

We considered the following hyperparameter grids for l_{BM} and the number of parameters: $\{3, 4, 5\} \times \{10^5, 5 \cdot 10^5, 10^6\}$. However, as setting the number of parameters as $10^5, 5 \cdot 10^5$, or 10^6 is infeasible in many cases, we considered the following to set the number of hyperparameters close to these numbers: $p_{ctr} \in \arg\min_{p'_{ctr}} \left|\#\text{params}(f^m(\mathbf{x}, G; \theta(p'_{ctr}))) - n^*\right|$ where n^* is the target number of parameters (e.g., 10^5), and p_{ctr} is a hyperparameter controlling the number of the parameters of $\theta(p_{ctr})$.[11]

---

[10] Additionally, parametric models (e.g., Black and Scholes (1973), Heston (1993)) based on the AF assumption or present-value functions are unsuitable as benchmarks, since they predict $y_{a,t} = 0$ for all a and t, as shown in Proposition 1.

[11] For RNC, p_{ctr} is the number of trees of an RNC layer, which is the same for all the layers. For GCN, SAGE, GAT, and GPS, p_{ctr} is the number of their hidden neurons—specifically, `out_channels` and `channels` in `Pytorch-Geometric`. The other hyperparameters were set as the values suggested in the underlying articles.

Experimental Results. Model performance was evaluated using the MSEs on $\mathcal{D}^{ar}_{\text{test},i}$ for all i. As shown in Table 2, our method consistently outperforms the benchmarks in terms of average MSEs across all p_{univ} values. The MSE magnitudes are low enough to yield profits, which we discuss in Section 6.4. Due to the nature of the prediction target, the MSEs are significantly influenced by unobserved external factors: Figure 7 shows the average with respect to p_{univ}. The zoom-in box shows the values for all five methods. For example, the early 2020 spike corresponds to market disruptions from COVID-19, whose unprecedented dynamics (Baker et al. 2020) were not reflected in the training data or features.

We conducted statistical tests on the MSE differences between our method and each benchmark, similarly to Wu et al. (2021). Since the average MSEs in Table 2 might be distorted by large values as illustrated in Figure 7, Table 3 summarizes several tests applied to the differences $\hat{\sigma}^2_{bm} - \hat{\sigma}^2_{RNC}$ between the MSEs of RNConv and a benchmark for the same i, p_{univ} pairs.

For all values of p_{univ}, our method statistically significantly outperforms all benchmarks, as summarized in Table 4. Bold values indicate p-values for which the null hypothesis in the difference comparison is rejected under a significance level of 0.05, provided that the assumptions of the corresponding statistical tests are satisfied.
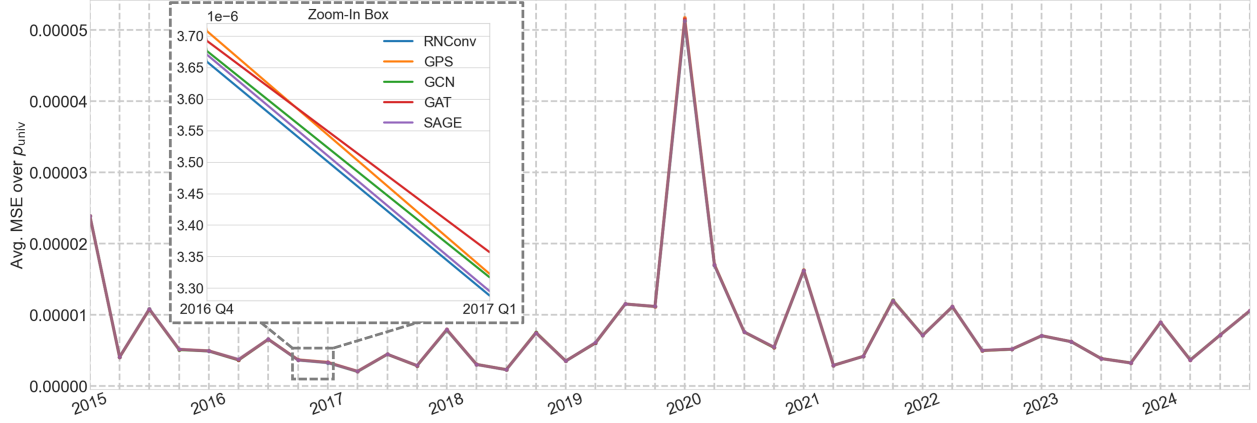
Table 2   **Averages MSEs over time (in 10^{-6})**

p_{univ}	GAT	GCN	GPS	SAGE	RNConv
16	6.3618	6.3361	6.3748	6.3343	**6.3254**
24	8.5003	8.4995	8.5042	8.5027	**8.4767**
32	9.5219	9.5234	9.5407	9.5440	**9.5034**

6.4. Synthetic-Long-Short Arbitrage

Benchmarks. To evaluate the performance of our method, projected SLSA $(SA; \mathcal{U}_t, \mathbf{n}_t)$, we consider the following two benchmark strategies:

- Benchmark 1: positions satisfying (56),
- Benchmark 2: positions satisfying (52) with $m = 0$,

where, in both cases, the positions are the projections of $\hat{\mathbf{v}}_t \in \mathbb{R}^{|\mathcal{U}_t|}$ onto their respective constraint sets, analogous to the SLSA projection. Unlike SLSA, Benchmark 1 may violate (57), and Benchmark 2 may fail to satisfy both (56) and (57), thus functioning as relaxed ablation variants.

Figure 7   Time Series of Avg. MSE over p_{univ} for RNConv

Table 3   Hypotheses of Table 4. $\hat{\sigma}^2_{bm}$ and $\hat{\sigma}^2_{RNC}$ denote the MSEs of a benchmark and RNConv, respectively.

Test	Null Hypothesis (H_0)	Alternative Hypothesis (H_1)
Paired T	$\mathrm{mean}(\hat{\sigma}^2_{\mathrm{bm}} - \hat{\sigma}^2_{\mathrm{RNC}}) \leq 0$	$\mathrm{mean}(\hat{\sigma}^2_{\mathrm{bm}} - \hat{\sigma}^2_{\mathrm{RNC}}) > 0$
Wilcoxon Signed-Rank	$\mathrm{median}(\hat{\sigma}^2_{\mathrm{bm}} - \hat{\sigma}^2_{\mathrm{RNC}}) \leq 0$	$\mathrm{median}(\hat{\sigma}^2_{\mathrm{bm}} - \hat{\sigma}^2_{\mathrm{RNC}}) > 0$
Permutation	$\mathrm{mean}(\hat{\sigma}^2_{\mathrm{bm}} - \hat{\sigma}^2_{\mathrm{RNC}}) = 0$	$\mathrm{mean}(\hat{\sigma}^2_{\mathrm{bm}} - \hat{\sigma}^2_{\mathrm{RNC}}) > 0$
Shapiro-Wilk	$\hat{\sigma}^2_{\mathrm{bm}} - \hat{\sigma}^2_{\mathrm{RNC}} \sim \mathcal{N}(\mu, \sigma^2)$	$\hat{\sigma}^2_{\mathrm{bm}} - \hat{\sigma}^2_{\mathrm{RNC}} \nsim \mathcal{N}(\mu, \sigma^2)$
Kolmogorov-Smirnov	$\hat{\sigma}^2_{\mathrm{bm}} - \hat{\sigma}^2_{\mathrm{RNC}} \sim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$	$\hat{\sigma}^2_{\mathrm{bm}} - \hat{\sigma}^2_{\mathrm{RNC}} \nsim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$
Symmetry (Miao et al. 2006)	$\hat{\sigma}^2_{\mathrm{bm}} - \hat{\sigma}^2_{\mathrm{RNC}}$ is symmetric	$\hat{\sigma}^2_{\mathrm{bm}} - \hat{\sigma}^2_{\mathrm{RNC}}$ is not symmetric

Since our prediction target is newly proposed in this paper and therefore lacks an established benchmark strategy for position-taking, we ad hoc adapted the strategies of Wang et al. (2024) and Wang and Xu (2024) as Benchmark 1 and 2, respectively. In evaluating prediction models for individual put and call option pricing, Wang et al. (2024) considered symmetric-long-short positions within the same-maturity options, while Wang and Xu (2024) considered them for all options. Equation (56) implements the former, whereas (52) with $m = 0$ realizes the latter. Here, symmetric-long-short positions refer to equal numbers of long and short contracts.

Table 4   P-values from Statistical Tests on MSE Differences

p_{univ}	Benchmark	Paired T	Wilcoxon Signed-Rank	Permutation	Shapiro-Wilk	Kolmogorov-Smirnov	Symmetry
16	GAT	0.0659	**0.0001**	0.0001	0.0000	0.0000	0.1462
	GCN	**0.0009**	0.0013	0.0009	0.9718	0.8945	0.9642
	GPS	0.1278	0.0809	**0.0244**	0.0000	0.0000	0.0352
	SAGE	**0.0196**	0.0093	0.0195	0.2657	0.9584	0.6572
24	GAT	0.0001	**0.0000**	0.0001	0.0116	0.2810	0.1244
	GCN	0.0005	**0.0000**	0.0005	0.0113	0.3573	0.5212
	GPS	0.0073	0.0030	**0.0065**	0.0000	0.0344	0.0132
	SAGE	0.0003	0.0000	**0.0001**	0.0010	0.0813	0.0244
32	GAT	**0.0030**	0.0028	0.0027	0.1473	0.5161	0.4722
	GCN	**0.0013**	0.0006	0.0010	0.0821	0.5675	0.7320
	GPS	0.0001	0.0000	**0.0001**	0.0017	0.1606	0.0372
	SAGE	0.0000	0.0000	**0.0000**	0.0000	0.0519	0.0134

Compared to SLSA's payoff (60), the benchmarks can have nonzero payoffs at maturity $M \in \mathcal{M}(\mathcal{U}_t) = \{M_a : a \in \mathcal{U}_t\}$. Their net payoffs at $M \in \mathcal{M}(\mathcal{U}_t)$ are as follows:

$$\text{Payoff}_M(BM_1; \mathcal{U}_t, \mathbf{n}) = \sum_{a \in \mathcal{U}_t : M_a = M} -n_a K_a \tag{64}$$

$$\text{Payoff}_M(BM_2; \mathcal{U}_t, \mathbf{n}) = \sum_{a \in \mathcal{U}_t : M_a = M} n_a(S_M - K_a) \tag{65}$$

while both of their inception payoffs are

$$\sum_{a \in \mathcal{U}_t} n_a K_a \left(y_{a,o(t)} + \bar{\delta}_{M_a, o(t)} \right) \tag{66}$$

from (55), and all the payoffs at the other time points are zero. Both (64) and (65) are derived from (16). Since Benchmark 1 satisfies (56), the terms of S_M cancel out.

Compared to SLSA, both benchmarks can contain synthetic bonds, and Benchmark 1 can additionally hold the underlying instrument. In (64) and (65), nonzero $\sum_{a \in \mathcal{U}_t : M_a = M} n_a K_a$, indicates

exposure to synthetic bonds. Nonzero $\sum_{a \in \mathcal{U}_t : M_a = M} n_a S_M$ in (65) implies a position in the underlying instrument.
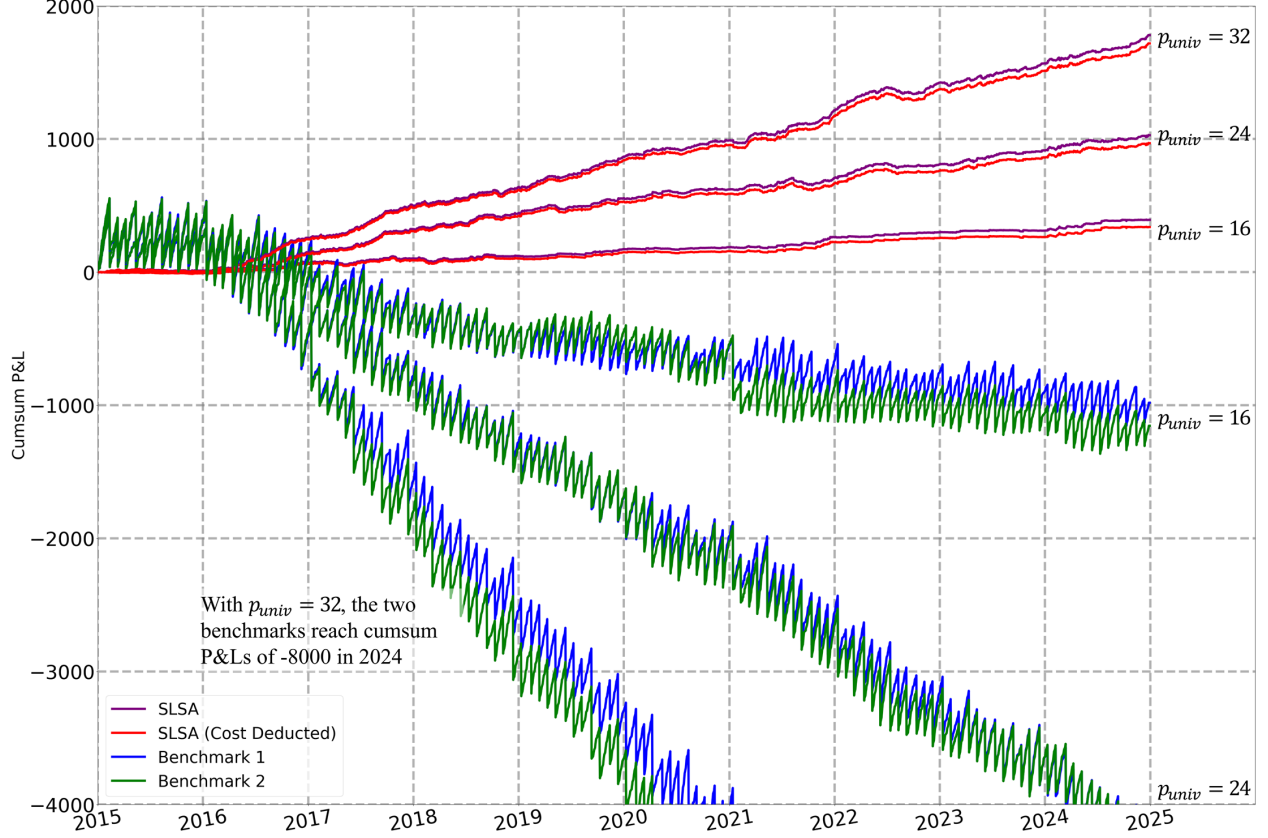
The RNConv predictions $\hat{y}_{a,o(t)}$ are used to compute $\hat{v}_{a,t}$, from which the projected SLSA and benchmark positions are derived. Each morning, positions totaling one long and one short contract are constructed and held until each option's expiration. Performance are evaluated across p_{univ}.

Experimental Results. The cumulative sums of the profits and losses (P&L) of SLSA show steady upward slopes for all p_{univ}, as shown in Figure 8, of which evaluation metrics can be found in Table 5. In Figure 8, transaction costs are set at 0.09% of traded monetary values, reflecting the minimum commission rate offered by Miraeasset Securities in Korea. In Table 5, treating P&L/(Number Of Contracts) as the return, information and Sortino ratios are computed with a zero benchmark return. HHI denotes the Herfindahl-Hirschman Index, a measure of position concentration. The effective N is the reciprocal of the HHI and reflects how many options are effectively held, based on the distribution of positions. The values of $avg|S - K|$ and $avg|M - t|$ represent the averages of $\sum_{a \in \mathcal{U}_t} n_{a,t}|S_t - K_a|$ and $\sum_{a \in \mathcal{U}_t} n_{a,t}|M_a - t|$, respectively. These indicate how much the strategy allocates to deeper ITM or OTM options and to longer-term maturities. Since larger p_{univ} invests in deeper-OTM-ITM options, as shown in $avg|S - K|$ in Table 5, and since such options can provide larger arbitrages, as shown in Figure 9, larger-p_{univ} projected SLSA positions can yield more profits. The projected SLSA generates small P&Ls in 2015, which is, in our opinion, because the projected SLSA position is relatively closer to the signal having relatively high errors in 2015, as shown in Figures 7 and 10.

Compared to the projected SLSAs, Benchmarks 1 and 2 exhibit steady downward slopes with spikes and drops around maturities. Due to the synthetic bond component in (66), the benchmarks may generate larger profits on each trading date. However, they are also exposed to the cash flows required to clear these bonds at each maturity, as described in (64) and (65). Since the positions generated by Benchmarks 1 and 2 are positively correlated with strike prices K, as shown in Table 6, the maturity cash flows ((64) and (65)) become negative in many cases, leading to the observed drops on maturities. From these results, we confirm the necessity of (56) and (57).

7. Conclusion

This paper presents a two-step graph learning framework to exploit StatArbs in options markets, addressing two key gaps in the literature: (i) the lack of direct deep learning methods for StatArbs in options markets, and (ii) overlooking the tabular form of the features commonly seen in practice. In
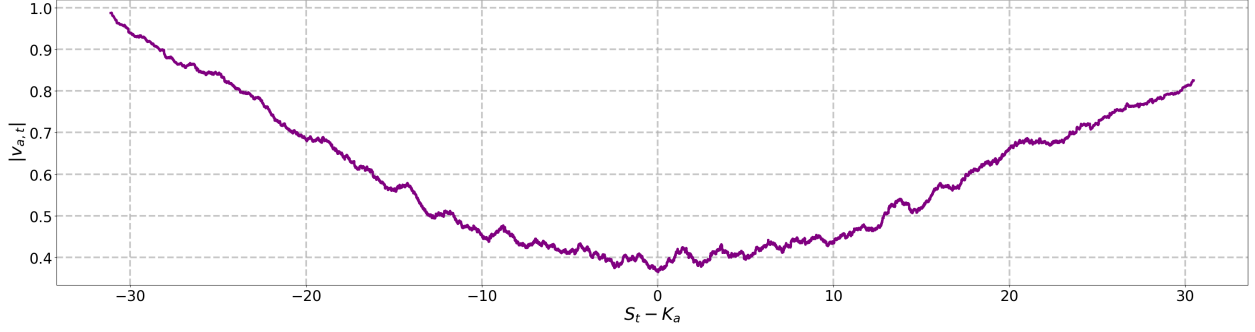
Figure 8   Comparison of Cumulative P&Ls: Projected SLSA and Benchmark Strategies

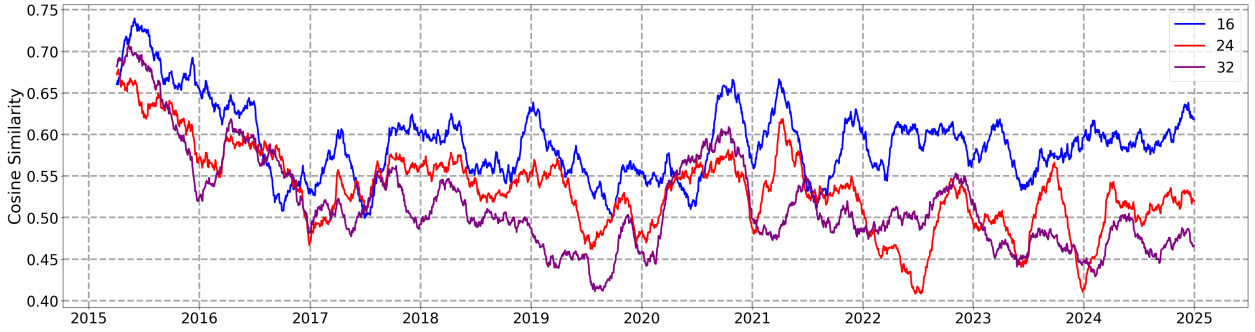**Table 5**   Averages of Annual Metrics of Projected SLSA

p_{univ}	16	24	32
Cumsum P&L	39.2621	102.8484	178.0427
Information Ratio	0.0938	0.1592	0.2351
Max Drawdown	14.4298	24.0106	23.1582
Sortino Ratio	0.1796	0.2585	0.4007
Hit Rate	0.4841	0.5327	0.5762
HHI	0.1033	0.0681	0.0549
Effective N	9.9559	15.0034	18.8209
$avg|S - K|$	8.4020	10.3882	11.8789
$avg|M - t|$	19.0769	23.4991	26.3175

the first step, we present a prediction target that provably contains pure arbitrages. To predict this, we introduce RNConv, a novel architecture that integrates tree-based structures with graph convolution

Figure 9   **Rolling Average of $|v_{a,t}|$ over $S_t - K_a$**

Figure 10   **63-day Rolling Mean of Cosine Similarity between Prediction $\hat{v}_t$ and Position n_t**

Table 6   **Correlation between K and n_t**

p_{univ}	Projected SLSA	Benchmark 1	Benchmark 2
16	0.0000	0.1081	0.1085
24	0.0000	0.1415	0.1413
32	0.0000	0.1636	0.1645

to effectively utilize tabular node features. In the second step, we propose SLSA, a class of synthetic long positions theoretically shown to contain only arbitrage opportunities. We further prove that SLSA has the lowest risk level and possesses neutrality to all Black and Scholes (1973) risk factors under the arbitrage-free assumption. Moreover, we introduce the SLSA projection, which maps model predictions to SLSA positions capable of realizing StatArbs. Experimental results show that RNConv statistically significantly outperforms the benchmarks, and SLSA positions derived from RNConv predictions yield consistent, upward-sloping cumulative P&L curves, achieving an average information ratio of 0.1627 based on P&L-contract returns. Therefore, we offer a perspective on the prediction target and strategy for capitalizing on StatArbs in options markets within the context of deep learning in tandem with pioneering tree-based graph learning.

Nevertheless, there remains room for future work. First, the underlying causes of the arbitrages within our prediction target need to be elucidated. Moreover, such an investigation may lead to more effective predictive features. Incorporating edge features into our method may further enhance predictive power. Next, since fractional contracts cannot be traded in practice, methods for transforming them into integer-valued positions warrant further study. Lastly, depending on the investor's circumstances (e.g., budget), more realistic constraints—such as limiting the investment to a smaller universe—may need to be considered, which is left for future work.

References

Ali U, Hirshleifer D (2020) Shared analyst coverage: Unifying momentum spillover effects. *Journal of Financial Economics* 136(3):649–675.

Alonso NI (2025) Look-ahead bias in large language models (LLMs): Implications and applications in finance. Preprint, submitted January 10, `https://dx.doi.org/10.2139/ssrn.5022165`.

Anand A, Chakravarty S (2007) Stealth trading in options markets. *Journal of Financial and Quantitative Analysis* 42(1):167–187.

Anders U, Korn O, Schmitt C (1998) Improving the pricing of options: A neural network approach. *Journal of Forecasting* 17(5–6):369–388.

Baker SR, Bloom N, Davis SJ, Kost K, Sammon M, Viratyosin T (2020) The unprecedented stock market reaction to COVID-19. *The Review of Asset Pricing Studies* 10(4):742–758.

Becker S, Cheridito P, Jentzen A (2019) Deep optimal stopping. *Journal of Machine Learning Research* 20(74):1–25.

Beutel A, Covington P, Jain S, Xu C, Li J, Gatto V, Chi EH (2018) Latent cross: Making use of context in recurrent recommender systems. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 46–54 (ACM, New York, NY, USA).

Bhatti UA, Tang H, Wu G, Marjan S, Hussain A (2023) Deep learning with graph convolutional networks: An overview and latest applications in computational intelligence. *International Journal of Intelligent Systems* 2023(1):8342104.

Black F, Scholes M (1973) The pricing of options and corporate liabilities. *Journal of Political Economy* 81(3):637–654.

Blasco N, Corredor P, Santamaría R (2010) Does informed trading occur in the options market? Some revealing clues. *Accounting & Finance* 50(3):555–579.

Borisov V, Leemann T, Seßler K, Haug J, Pawelczyk M, Kasneci G (2024) Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems* 35(6):7499–7519.

Buehler H, Gonon L, Teichmann J, Wood B (2019) Deep hedging. *Quantitative Finance* 19(8):1271–1291.

Chen F, Sutcliffe C (2012) Pricing and hedging short sterling options using neural networks. *Intelligent Systems in Accounting, Finance and Management* 19(2):128–149.

Chen J, Wang X, Xu X (2022) GC-LSTM: Graph convolution embedded LSTM for dynamic network link prediction. *Applied Intelligence* 52(7):7513–7528.

Chen T, Guestrin C (2016) XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794 (ACM, New York, NY, USA).

Choi HJ, Lee HS, Han GS, Lee J (2004) Efficient option pricing via a globally regularized neural network. *Advances in Neural Networks - ISNN 2004*, 988–993 (Springer, Berlin, Heidelberg, Germany).

Corrado CJ, Su T (1996) Skewness and Kurtosis in S&P 500 index returns implied by option prices. *Journal of Financial Research* 19(2):175–192.

Cox JC, Ross SA, Rubinstein M (1979) Option pricing: A simplified approach. *Journal of Financial Economics* 7(3):229–263.

Dugas C, Bengio Y, Bélisle F, Nadeau C, Garcia R (2009) Incorporating functional knowledge in neural networks. *Journal of Machine Learning Research* 10(42):1239–1262.

Farahani MS, Babaei S, Esfahani A (2024) "Black-Scholes-artificial neural network": A novel option pricing model. *International Journal of Financial, Accounting, and Management* 5(4):489–523.

Fey M, Lenssen JE (2019) Fast graph representation learning with PyTorch Geometric. *ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds* (ICLR, Online).

François P, Gauthier G, Godin F, Mendoza COP (2025) Is the difference between deep hedging and delta hedging a statistical arbitrage? *Finance Research Letters* 73:106590.

Guijarro-Ordonez J, Pelger M, Zanotti G (2022) Deep learning statistical arbitrage. Preprint, submitted October 7, https://arxiv.org/abs/2106.04028.

Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1025–1035 (Curran Associates, Inc., Red Hook, NY, USA).

Heston SL (1993) A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The Review of Financial Studies* 6(2):327–343.

Horikawa H, Nakagawa K (2024) Relationship between deep hedging and delta hedging: Leveraging a statistical arbitrage strategy. *Finance Research Letters* 62:105101.

Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2261–2269 (IEEE, Piscataway, NJ, USA).

Hull J, Treepongkaruna S, Colwell D, Heaney R, Pitt D (2013) *Fundamentals of Futures and Options Markets* (Pearson Higher Education AU, Camberwell, VIC, Australia).

Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, 448–456 (ACM, New York, NY, USA).

Ivașcu CF (2021) Option pricing using machine learning. *Expert Systems with Applications* 163:113799.

Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu TY (2017) LightGBM: A highly efficient gradient boosting decision tree. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 3149–3157 (Curran Associates Inc., Red Hook, NY, USA).

Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations* (ICLR, Online).

Kohavi R (1994) Bottom-up induction of oblivious read-once decision graphs. *Machine Learning: ECML-94*, 154–169 (Springer, Berlin, Heidelberg, Germany).

Kohler M, Krzyżak A, Todorovic N (2010) Pricing of high-dimensional American options by neural networks. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics* 20(3):383–410.

Korea Exchange (2025a) KRX Global Website. Accessed March 27, https://global.krx.co.kr/main/main.jsp.

Korea Exchange (2025b) KRX Market Data System. Accessed January 12, http://data.krx.co.kr/contents/MDC/MAIN/main/index.cmd.

Langenohl A (2018) Sources of financial synchronism: Arbitrage theory and the promise of risk-free profit. *Finance and Society* 4(1):26–40.

Lazzarino M, Berrill J, Šević A, et al. (2018) What is statistical arbitrage? *Theoretical Economics Letters* 8(05):888.

Liu X, Cao Y, Ma C, Shen L (2019) Wavelet-based option pricing: An empirical study. *European Journal of Operational Research* 272(3):1132–1142.

Liu Z, Zhou J (2022) *Introduction to graph neural networks* (Springer Nature, Cham, Switzerland).

Lou Y, Obukhov M (2017) BDT: Gradient boosted decision tables for high accuracy and scoring efficiency. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1893–1901 (ACM, New York, NY, USA).

Malliaris M, Salchenberger L (1993) A neural network model for estimating option prices. *Applied Intelligence* 3:193–206.

Malliaris M, Salchenberger L (1996) Using neural networks to forecast the S&P 100 implied volatility. *Neurocomputing* 10(2):183–195.

McElfresh D, Khandagale S, Valverde J, Prasad C V, Ramakrishnan G, Goldblum M, White C (2023) When do neural nets outperform boosted trees on tabular data? *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 76336–76369 (Curran Associates Inc., Red Hook, NY, USA).

Merton RC (1976) Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics* 3(1–2):125–144.

Miao W, Gel YR, Gastwirth JL (2006) A new test of symmetry about an unknown median. *Random Walk, Sequential Analysis and Related Topics*, 199–214 (World Scientific, Hackensack, NJ, USA).

Nau RF, McCardle KF (1992) Arbitrage, rationality, and equilibrium. *Decision Making Under Risk and Uncertainty: New Models and Empirical Findings*, 189–199 (Springer, Dordrecht, Netherlands).

Osterrieder J, Kucharczyk D, Rudolf S, Wittwer D (2020) Neural networks and arbitrage in the VIX: A deep learning approach for the VIX. *Digital Finance* 2(1):97–115.

Park H, Kim N, Lee J (2014) Parametric models and non-parametric machine learning models for predicting option prices: Empirical comparison study over KOSPI 200 index options. *Expert Systems with Applications* 41(11):5227–5237.

Peters B, Niculae V, Martins AF (2019) Sparse sequence-to-sequence models. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 1504–1519 (Association for Computational Linguistics, Stroudsburg, PA, USA).

Popov S, Morozov S, Babenko A (2020) Neural oblivious decision ensembles for deep learning on tabular data. *International Conference on Learning Representations* (ICLR, Online).

Prokhorenkova L, Gusev G, Vorobev A, Dorogush AV, Gulin A (2018) CatBoost: Unbiased boosting with categorical features. *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 6639–6649 (Curran Associates Inc., Red Hook, NY, USA).

Rampášek L, Galkin M, Dwivedi VP, Luu AT, Wolf G, Beaini D (2022) Recipe for a general, powerful, scalable graph transformer. *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 14501–14515 (Curran Associates Inc., Red Hook, NY, USA).

Rana PS, Modi SK, Yadav AL, Singla S, et al. (2023) Comparative analysis of tree-based models and deep learning architectures for tabular data: Performance disparities and underlying factors. *International Conference on Advanced Computing & Communication Technologies (ICACCTech)*, 224–231 (IEEE, Piscataway, NJ, USA).

Rubinstein M (1983) Displaced diffusion option pricing. *The Journal of Finance* 38(1):213–217.

Ruf J, Wang W (2020) Neural networks for option pricing and hedging: A literature review. Preprint, submitted May 9, http://arxiv.org/abs/1911.05620.

Schittenkopf C, Dorffner G (2001) Risk-neutral density extraction from option prices: improved pricing with mixture density networks. *IEEE Transactions on Neural Networks* 12(4):716–725.

Shin HJ, Ryu J (2012) A dynamic hedging strategy for option transaction using artificial neural networks. *International Journal of Software Engineering and its Applications* 6(4):111–116.

Shwartz-Ziv R, Armon A (2022) Tabular data: Deep learning is not all you need. *Information Fusion* 81:84–90.

Skiadas C (2024) *Theoretical Foundations of Asset Pricing* (Cambridge University Press, Cambridge, England).

Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Ter Horst JR, Nijman TE, Verbeek M (2001) Eliminating look-ahead bias in evaluating persistence in mutual fund performance. *Journal of Empirical Finance* 8(4):345–373.

Varian HR (1987) The arbitrage principle in financial economics. *Journal of Economic Perspectives* 1(2):55–72.

Vasicek O (1977) An equilibrium characterization of the term structure. *Journal of Financial Economics* 5(2):177–188.

Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. *International Conference on Learning Representations* (ICLR, Online).

Wang R, Fu B, Fu G, Wang M (2017) Deep & cross network for ad click predictions. *Proceedings of the ADKDD'17* (ACM, New York, NY, USA).

Wang R, Shivanna R, Cheng D, Jain S, Lin D, Hong L, Chi E (2021) DCN V2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. *Proceedings of the Web Conference 2021*, 1785–1797 (ACM, New York, NY, USA).

Wang W, Xu J (2024) Deep learning option price movement. *Risks* 12(6):93.

Wang Y, Zhao J, Li Q, Wei X (2024) Considering momentum spillover effects via graph neural network in option pricing. *Journal of Futures Markets* 44(6):1069–1094.

Wu Q, Nasoz F, Jung J, Bhattarai B, Han MV, Greenes RA, Saag KG (2021) Machine learning approaches for the prediction of bone mineral density by using genomic and phenotypic data of 5130 older men. *Scientific Reports* 11(1):4482.

Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS (2019) A comprehensive survey on graph neural networks. Preprint, submitted December 4, `https://arxiv.org/abs/1901.00596`.

Yang Y, Zheng Y, Hospedales T (2017) Gated neural networks for option pricing: Rationality by design. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 52–58 (AAAI Press, Washington, DC, USA).

Yıldız AY, Kalayci A (2025) Gradient boosting decision trees on medical diagnosis over tabular data. Preprint, submitted January 19, `https://arxiv.org/abs/2410.03705`.

Zapart C (2003) Statistical arbitrage trading with wavelets and artificial neural networks. *IEEE International Conference on Computational Intelligence for Financial Engineering*, 429–435 (IEEE, Piscataway, NJ, USA).

Zhan B, Zhang S, Du HS, Yang X (2022) Exploring statistical arbitrage opportunities using machine learning strategy. *Computational Economics* 60(3):861–882.

Zhang J, Huang W (2021) Option hedging using LSTM-RNN: an empirical analysis. *Quantitative Finance* 21(10):1753–1772.

Zhao Y, Xu S, et al. (2022) Deep learning meets statistical arbitrage: An application of long short-term memory networks to algorithmic trading. *Journal of Financial Data Science* 4(4):133–150.

Zheng Y, Yang Y, Chen B (2021) Incorporating prior financial domain knowledge into neural networks for implied volatility surface prediction. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 3968–3975 (ACM, New York, NY, USA).