Building Cross-Sectional Systematic Strategies By Learning to Rank

Daniel Poh, Bryan Lim, Stefan Zohren, and Stephen Roberts

***Abstract*—The success of a cross-sectional systematic strategy depends critically on accurately ranking assets prior to portfolio construction. Contemporary techniques perform this ranking step either with simple heuristics or by sorting outputs from standard regression or classification models, which have been demonstrated to be sub-optimal for ranking in other domains (e.g. information retrieval). To address this deficiency, we propose a framework to enhance cross-sectional portfolios by incorporating learning-to-rank algorithms, which lead to improvements of ranking accuracy by learning pairwise and listwise structures across instruments. Using cross-sectional momentum as a demonstrative case study, we show that the use of modern machine learning ranking algorithms can substantially improve the trading performance of cross-sectional strategies – providing approximately threefold boosting of Sharpe Ratios compared to traditional approaches.**

I. Introduction

Cross-sectional strategies are a popular style of systematic trading – with numerous flavours documented in the academic literature across different trading insights and asset classes [1]. In contrast to time-series approaches [2] which consider each asset independently, cross-sectional strategies capture risk premia by trading assets against each other – buying assets with the highest expected returns and selling those with the lowest. The classical cross-sectional momentum strategy of [3], for instance, selects stocks by ranking their respective returns over the past year and betting that the order of returns will persist into the future – buying assets in the top decile while selling the bottom decile. By trading winners against losers, cross-sectional strategies have been shown to be more insulated against common market moves, and perform in cases even when assets have non-negligible correlations (e.g. equity markets) [1, 4, 5].

With the rise of machine learning in recent years, many cross-sectional systematic strategies which incorporate advanced prediction models have been proposed [6–8], often demonstrating significant improvements over traditional baselines. In general, these machine learning models are trained in a supervised fashion, and aim to minimise the mean squared error (MSE) of forecast returns over the holding time period. While the regression models under this framework accurately provide a mean estimate of future asset returns, they do not explicitly consider the *expected ordering* of returns – which is at the core of the design of cross-sectional strategies. This could hence have a negative impact on strategy performance, and consequently lead to sub-optimal investment decisions.

D. Poh, B. Lim, S. Zohren and S. Roberts are with the Department of Engineering Science and the Oxford-Man Institute of Quantitative Finance, University of Oxford, Oxford, United Kingdom (email:{dp, blim, zohren, sjrob}@robots.ox.ac.uk).

The limits of standard regression methods for ranking have been extensively studied in information retrieval applications, with a number of metrics and methodologies proposed [9]. Collectively referred to as Learning to Rank (LTR) [10], today's variants make use of modern learning techniques such as deep neural networks and tree-based methods [11–13] – leading to dramatic improvements in accuracy over simpler baselines. While LTR algorithms have also been used to a small degree in finance, the majority of papers focus on its use in simple stock recommendations [14], and lack a framework for the development and evaluation of general cross-sectional strategies.

In this paper, we show how LTR models can be used to enhance traditional cross-sectional systematic strategies – adopting cross-sectional momentum as a demonstrative use-case. We first start by casting stock selection as a general ranking problem, which allows for the flexibility of switching between different LTR algorithms and incorporating state-of-the-art models developed in other domains. Next, we concretely evaluate LTR models against a mixture of standard supervised learning methods and heuristic benchmarks. Through tests on US equities, we demonstrate that the pairwise and listwise models better substantively improve the ranking accuracy of stocks – leading to overall improvements in strategy performance with the adoption of LTR methods. Finally, although our ranking algorithms make use of momentum predictors, their modular nature allows these inputs to be adapted to incorporate other feature sets – providing a novel and generalisable framework for general cross-sectional strategies.

II. Related Works

A. *Cross-Sectional Momentum Strategies*

Momentum strategies can either be categorised as (univariate) time series or (multivariate) cross-sectional. In time series momentum, an asset's trading rule depends only on its own historical returns. It was first proposed by [2], who documented the profitability of the strategy in trading nearly 60 different liquid instruments individually over 25 years. This has prompted numerous subsequent works such as [1, 15, 16] which explore various trading rules alongside different trend estimation and position sizing techniques that are aimed at refining the overall strategy.

Cross-sectional momentum employs a similar idea but focuses on comparing the relative performance of assets. It is characterised by first sorting the instruments by performance (typically taken to be returns), and then buying some fraction of top performers (winners) while simultaneously selling a similar-sized fraction of underperformers (losers). Since the earlier work in [3] which considers this strategy for US equity markets, the literature has been replete with a spectrum of works – ranging from those that report the existence of the momentum phenomenon in other markets and asset classes [17–22], to others that propose new ways to improve various aspects of the strategy. For instance, [6] performs ranking with the forward-looking Sharpe ratio instead of historical returns. [23] constructs rankings based on returns standardised by their respective volatilities, arguing that this allows for a fairer comparison of instrument performance; [1] employs a similar but more sophisticated approach by using volatility-normalised moving-average convergence divergence (MACD) indicators as inputs. A common thread across recent works is their use of a regress-then-rank approach [24] – minimising return predictions against targets before ranking the outputs and finally constructing a zero-investment portfolio by trading the tails of the sorted results. Notably, the loss commonly employed is the MSE (mean-squared error) which is a pointwise function, and some examples of works training with the MSE are [6] and [8].

Surveying works related to cross-sectional momentum, we believe that this is the first paper to consider the use of ranking algorithms to enhance cross-sectional momentum strategies. Instead of ranking based on heuristics or on outputs produced by models trained on pointwise losses, we propose using Learning to Rank algorithms – demonstrating that learning the pairwise and listwise structure across securities produces better ranking and consequently better out-of-sample strategy performance.

B. *Learning to Rank in Finance*

Learning to Rank is a key area of research in Information Retrieval and focuses on using machine learning techniques to train models to perform ranking tasks [10, 25]. The information explosion along with modern computing advances on both the hardware (cloud-based GPUs and TPUs [26]) and software (open source Deep Learning frameworks such as `Tensorflow` [27] and `PyTorch` [28]) fronts have induced a shift in how machine learning algorithms are designed – going from models that required handcrafting and explicit design choices towards those that employ neural networks to learn in a data-driven manner. This has prompted a parallel trend in the space of ranking models, with researchers migrating from using probabilistic varieties like the BM25 and LMIR (Language Model for Information Retrieval) that required no training [10], to sophisticated architectures that are built on the aforementioned developments [9].

Today, LTR algorithms play a key role in a myriad of commercial applications such as search engines [25], e-commerce [29] and entertainment [30]. These algorithms have been explored to a more limited degree in finance, with specific applications in equity sentiment analysis and ranking considered in [24, 31]. [31] apply RankNet and ListNet to ten years of market and news sentiment data, demonstrating higher risk-adjusted profitability over the S&P 500 index return, the HFRI Equity Market Neutral Index (HFRI EMN) as well as pure sentiment-based techniques. On the other hand, [24] documents the superiority of LambdaMART over standard neural networks in predicting intraday returns on the Shenzhen equity exchange, adopting order book based features.

Despite the initial promising results, we note that these applications do not consider comparisons to traditional styles of systematic trading – with comparisons only performed against market returns in [31] and neural network models in [24] – making it difficult to evaluate the true value added by LTR methods. The ranking methodologies are also customised to specific sentiment trading applications, making generalisation to other types of cross-sectional trading strategies challenging. We address these limitation explicitly in our paper – proposing a general framework for incorporating LTR models in cross-sectional strategies, and evaluating their performance.

III. Problem Definition

Given a securities portfolio that is rebalanced monthly, the returns for a cross-sectional momentum (CSM) strategy at τ_m can be expressed as follows:

$$r^{CSM}_{\tau_m,\tau_{m+1}} = \frac{1}{n_{\tau_m}} \sum_{i=1}^{n_{\tau_m}} X^{(i)}_{\tau_m} \frac{\sigma_{tgt}}{\sigma^{(i)}_{\tau_m}} r^{(i)}_{\tau_m,\tau_{m+1}} \tag{1}$$

where $\tau_m, \tau_{m+1} \in \boldsymbol{\mathcal{T}} \subset \{1, ..., t-1, t, t+1, ..., T\}$ and $\boldsymbol{\mathcal{T}}$ denotes the set of indices coinciding with the last trading day of every month; $r^{CSM}_{\tau_m,\tau_{m+1}}$ are the realised portfolio returns from month τ_m to τ_{m+1}, n_{τ_m} refers to the number of stocks in the portfolio and $X^{(i)}_{\tau_m} \in \{-1, 0, 1\}$ characterises the cross-sectional momentum signal or trading rule for security i. We rebalance monthly to avoid excessive transaction costs that comes with trading at higher frequencies, daily for example. We also fix the annualised target volatility σ_{tgt} at 15% and scale asset returns with $\sigma^{(i)}_{\tau_m}$ which is an estimator for ex-ante monthly volatility. In this paper, we use a rolling exponentially weighted standard deviation with a 63-day span on daily returns for $\sigma^{(i)}_{\tau_m}$, but note that more sophisticated methods (e.g. GARCH [32]) can be used.

A. Strategy Framework

The general framework for the CSM strategy comprises of the following 4 components.

1) Score Calculation:

$$Y^{(i)}_{\tau_m} = f(\boldsymbol{u}^{(i)}_{\tau_m}), \tag{2}$$

Presented with an input vector $\boldsymbol{u}^{(i)}_{\tau_m}$ for asset i at τ_m, the strategy's prediction model f computes its corresponding score $Y^{(i)}_{\tau_m}$. For a cross-sectional universe of size N_{τ_m} at τ_m, the list of scores of assets considered for trading is represented by the vector $Y_{\tau_m} = \{Y^{(1)}_{\tau_m}, ..., Y^{(N_{\tau_m})}_{\tau_m}\}$.

2) Score Ranking:

$$Z^{(i)}_{\tau_m} = \mathcal{R}(Y_{\tau_m})^{(i)} \tag{3}$$

with $Z^{(i)}_{\tau_m} \in \{1, ..., N_{\tau_m}\}$ being the position index for asset i after applying the operator $\mathcal{R}(\cdot)$ to sort scores in ascending order.

3) Security Selection:

$$X^{(i)}_{\tau_m} = \begin{cases} -1 & Z^{(i)}_{\tau_m} \leq \lfloor 0.1 \times N_{\tau_m} \rfloor \\ 1 & Z^{(i)}_{\tau_m} > \lfloor 0.9 \times N_{\tau_m} \rfloor \\ 0 & \text{Otherwise} \end{cases} \tag{4}$$

Selection is usually a thresholding step where some fraction of assets are retained to form the respective long/short portfolios. Equation (4) assumes that we are using the typical decile-sized portfolios for the strategy (i.e. top and bottom 10%).

4) Portfolio Construction: Finally, simple portfolios can then be constructed by volatility scaling the selected instruments based on Equation (1).

In the following section, we provide an overview of score calculation techniques for both current strategy approaches and LTR models.

IV. Score Calculation Methodologies

Most CSM strategies adhere to this framework and are generally similar over the last three steps, i.e., how they go about ranking scores, selecting assets and constructing the portfolio. However, they are particularly diverse in their choice of the prediction model f used to calculate scores, ranging from simple heuristics [3] to sophisticated architectures on an expansive list of macroeconomic inputs [7]. While numerous techniques to compute scores exist, they can be grouped into 3 categories: Classical Momentum, Regress-then-Rank – both of which are current approaches – and finally Learning to Rank, which is our proposed method.

A. Classical Cross-Sectional Momentum

Classical variants of the CSM tend to lean towards the use of comparatively simple procedures for the score calculation.

Jegadeesh & Titman, 1993 [3]:

The authors who first documented the CSM strategy, propose scoring an asset with its raw cumulative returns, computed over the past 3 to 12 months:

$$\textbf{Score Calculation:} \qquad Y^{(i)}_{\tau_m} = r^{(i)}_{\tau_m-252,\tau_m} \tag{5}$$

where $r^{(i)}_{\tau_m-252,\tau_m}$ is the raw returns over the previous 252 days (12 months) from τ_m for asset i.

Baz et al., 2015 [1]:

A sophisticated alternative uses volatility-normalised MACD indicators as an intermediate signal, forming the final signal by combining indicators computed over different time scales. The indicator is given as:

$$\tilde{Y}^{(i)}_{\tau_m} = \xi^{(i)}_{\tau_m} / \text{std}(z^{(i)}_{\tau_m-252:\tau_m}) \tag{6}$$

$$\xi^{(i)}_{\tau_m} = \text{MACD}(i, \tau_m, S, L) / \text{std}(p^{(i)}_{\tau_m-63:\tau_m}) \tag{7}$$

$$\text{MACD}(i, \tau_m, S, L) = m(i, S) - m(i, L) \tag{8}$$

where $\text{std}(p^{(i)}_{\tau_m-63:\tau_m})$ represents the 63-day rolling standard deviation of security i, while $m(i,S)$ is an exponentially weighted moving average of prices for asset i and S translates to a half-life decay factor $HL = \log(0.5)/\log(1-1/S)$. The final composite signal combines different volatility-scaled MACDs over different time scales involving a response function $\phi(\cdot)$ and a set of short and long time scales $S_k \in \{8, 16, 32\}$ and $L_k \in \{24, 48, 96\}$ as set out in [1]:

$$\textbf{Score Calculation:} \quad Y^{(i)}_{\tau_m} = \sum_{k=1}^{3} \phi\big(\tilde{Y}^{(i)}_{\tau_m}(S_k, L_k)\big). \tag{9}$$

B. Regress-then-Rank Method

Newer works employing a regress-then-rank approach typically compute the score via a standard regression (refer to Section II-A):

$$\textbf{Score Calculation:} \qquad Y^{(i)}_{\tau_m} = f(\boldsymbol{u}^{(i)}_{\tau_m}; \boldsymbol{\theta}) \tag{10}$$

where f characterises a machine learning prediction model parameterised by $\boldsymbol{\theta}$ presented with some input vector $\boldsymbol{u}^{(i)}_{\tau_m}$. Using the volatility normalised returns as the target, the model is trained by minimising the loss, which is typically the MSE:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) =& \frac{1}{M} \sum_{\Omega} \left(Y^{(i)}_{\tau_m} - \frac{r^{(i)}_{\tau_m,\tau_{m+1}}}{\sigma^{(i)}_{\tau_m}} \right)^2 \\ \Omega =& \Big\{ (Y^{(1)}_{\tau_1}, r^{(1)}_{\tau_1,\tau_2}/\sigma^{(1)}_{\tau_1}), \ldots, \\ & (Y^{(N_{\tau_{m-1}})}_{\tau_{m-1}}, r^{(N_{\tau_{m-1}})}_{\tau_{m-1},\tau_m}/\sigma^{(N_{\tau_{m-1}})}_{\tau_{m-1}}) \Big\} \end{aligned} \tag{11}$$

where Ω represents the set of all M possible forecasted and target tuples over the set of instruments and relevant time steps.

C. Learning to Rank Algorithms

LTR methods can be categorised as being pointwise, pairwise or listwise. The pointwise (pairwise) approach casts the ranking problem as a classification, regression or ordinal classification of individual (pairs of) samples, while the listwise approach on the other hand learns the appropriate ranking model by using ranking lists as inputs. In terms of ranking performance, the pointwise method has been observed to be inferior relative to the last two techniques [10]. Additionally, the loss function is not just the key difference across these models [10] – by incorporating the pairwise and listwise information across assets, it makes LTR models collectively distinct from the those outlined in Sections IV-A and IV-B.

We provide a high level overview of 4 LTR algorithms which we use in conjunction with the momentum strategy, highlighting the loss function but omitting technical details to keep our exposition brief. For details adapting the LTR framework for the momentum strategy, please refer to Section VII-A in the Appendix.

Burges et al., 2005 (RankNet):

While techniques that apply neural networks to the ranking problem already exist, RankNet was the first to train network based on *pairs* of samples. Similar to contemporary methods, RankNet uses a neural network. Instead of minimising the MSE however, RankNet focuses on minimising the cross entropy error from classifying sample pairs – optimising the network based on the probability that one element has a higher rank than the other. As training is conducted on individual pairs using stochastic gradient descent, RankNet has a complexity that scales quadratically with the number of securities at rebalance time.

Burges et al., 2010 (LambdaMART):

LambdaMART is a state-of-the-art [33, 34] pairwise method that combines LambdaRank [35] with Multiple Additive Regression Trees (MART). Interestingly, training in LambdaMART via LambdaRank does not involve directly optimising a loss function but rather making use of heuristic approximations of the gradients (referred to as λ-gradients) – exploiting the fact that only the gradients and not actual loss values are required to train neural networks. This allows the models to circumvent dealing with the often flat, discontinuous and non-differentiable losses such as the NDCG (Normalised Discounted Cumulative Gain) [36], which is simultaneously a popular position-sensitive information retrieval metric and that LambdaRank has been shown to locally optimise [37, 38]. Given the formulation of λ-gradients, the loss involves the product of a pairwise cross entropy loss and the gain on some information retrieval metric (typically taken to be NDCG) [13]. MART on the other hand is a tree boosting method known for its flexibility. It also offer a simple way to trade off speed and accuracy via truncation which are important for time-critical applications such as search engines [13]. LambdaMART which is the result of marrying these methods thus combines LambdaRank's observed empirical optimality (with respect to NDCG) [37] with the flexibility and robustness of MART.

Cao et al., 2007 (ListNet):

ListNet was developed to address the practical issues inherent with pairwise techniques such as their prohibitive computational costs, as well as their mismatched objective of minimising errors related to pairs classification instead of the overall ranking itself. ListNet resolves these problems by making use of a listwise loss, adopting a probablistic approach on permutations. By first computing "top one" probability distributions over a list of scores and ground truth labels and then normalising each with a softmax operator, the loss is then defined to be the cross entropy between both these distributions. By using the entire cross-section of securities as inputs, ListNet has a complexity of $O(N_{\tau_m})$ at τ_m – making it more efficient than RankNet which has a quadratic complexity of $O(N_{\tau_m}^2)$ since training is conducted on pairs.

Xia et al., 2008 (ListMLE):

Seeking to analyse and provide more theoretical support linking the choice of a ranking model's listwise loss function to its corresponding performance, [39] proposed the use of the likelihood loss due to its nice properties of consistency, soundness and linear complexity. Additionally, the likelihood loss is continuous, differentiable and convex [40]. This culminated in the development of ListMLE – a probabilistic ranking approach that casts the ranking problem as minimising the likelihood loss, or equivalently as maximising the likelihood function of a probability model. ListMLE has been shown to outperform other listwise methods on benchmark data sets [39], and shares the same linear complexity as ListNet. Given our results which we further discuss later in the paper, we note that the benefit of the linear complexity possessed by both ListMLE and ListNet might be relevant for larger data sets.

D. Training Details

RankNet, ListNet and ListMLE were trained with the `Adam` optimiser based on each model's respective ranking loss function. Backpropagation was conducted for a maximum of 100 epochs where for a given training set, we partition 90% of the data for training and leave the remaining 10% for validation. As a matter of practicality, we set our target to be the returns 21 days ahead instead of the next month for training and validation. For Learning to Rank models using neural networks (i.e. RankNet, ListNet and ListMLE), we use 2 hidden layers but treated the width as a tunable hyperparameter. Early stopping was used to prevent model overfitting and is triggered when the model's loss on the validation set does not improve for 25 consecutive epochs. We also used dropout regularisation [41] in the networks-based models as an additional safeguard against overfitting and similarly treated the dropout rates as a hyperparameter to be calibrated over model learning. Across all models, hyperparameters were tuned by running 50 iterations of search using `HyperOpt` [42]. Further details on calibrating the hyperparameters can be found in Section VII-B of the Appendix.

V. Performance Evaluation

A. Dataset Overview

We construct our monthly portfolios using data from CRSP (Center for Research in Security Prices) [43]. Our universe comprises of actively traded firms on NYSE from 1980 to 2019 with a CRSP share code of 10 and 11. At each rebalancing interval, we only use stocks that are trading above \$1. Additionally, we only consider stocks with valid prices and that have been actively trading over the previous year. All prices are closing prices.

B. Backtest and Predictor Description

With the exception of both the classical strategies employing heuristic rankings, all other models were re-tuned at 5-year intervals. The weights and hyperparameters of the calibrated models were then fixed and used for out-of-sample portfolio rebalancing for the following 5-year window. The rebalancing takes place on the last trading day of each month. Focusing on ranking, we trade 100 stocks for each long and short portfolios at all times – amounting to approximately 10% of all tradeable stocks at each rebalancing interval. For predictors, we use a simple combination of the predictors employed by the classical approaches in Section IV-A:

1) *Raw cumulative returns* – Returns as per [3] over the past 3, 6 and 12-month periods.
2) *Normalised returns* – Returns over the past 3, 6 and 12-month periods standardised by daily volatility and then scaled to the appropriate time scale.
3) *MACD-based indicators* – Retaining the final signal as defined in Equation (9) from [1], we also augment our set of predictors by including the set of raw intermediate signals $\tilde{Y}_{\tau_m}^{(i)}(S_k, L_k)$ in Equation (6) for $k = 1, 2, 3$ computed at t as well as for the past 1, 3, 6 and 12-month periods – giving us a total of 16 features for this group.

C. Models and Comparison Metrics

The LTR and reference benchmarks models (with their corresponding shorthand in parentheses) studied in this paper are:

1) *Random (Rand)* – This model select stocks at random, and is included to provide an absolute baseline sense of what the ranking measures might look like assuming portfolios are composed in such a manner.
2) *Raw returns (JT)* – Heuristics-based ranking technique based on [3], which is one of the earliest works documenting the CSM strategy.
3) *Volatility Normalised MACD (Baz)* – Heuristics-based ranking technique with a relatively sophisticated trend estimator proposed by [1].
4) *Multi-Layer Perceptron (MLP)* – This model characterises the typical Regress-then-rank techniques used by contemporary methods.
5) *RankNet (RNet)* – Pairwise LTR model by [44].
6) *LambdaMART (LM)* – Pairwise LTR model by [45].
7) *ListNet (LNet)* – Listwise LTR model by [46].
8) *ListMLE (LMLE)* – Listwise LTR model by [39].

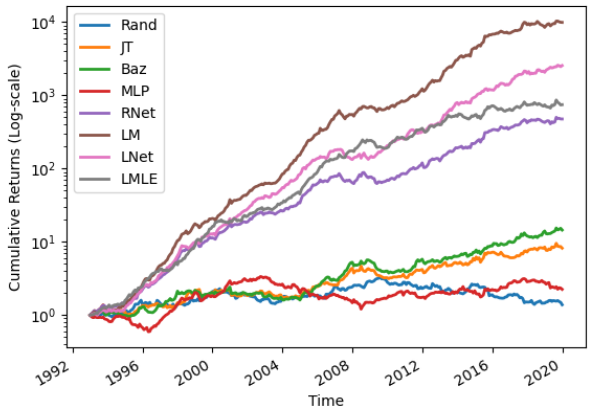
Performance of the various algorithms are finally evaluated with two sets of metrics – the first involving those commonly found in finance (1 to 3), and the latter from the information retrieval and ranking literature (4):

1) Profitability: Expected returns ($\mathbb{E}$[Returns]) as well as the percentage of positive returns at the portfolio-level obtained over the out-of-sample period.
2) Risks: Monthly volatility, Maximum Drawdown (MDD) and Downside Deviation.
3) Financial Performance: Sharpe $\left(\frac{\mathbb{E}[\text{Returns}]}{\text{Volatility}}\right)$, Sortino $\left(\frac{\mathbb{E}[\text{Returns}]}{\text{MDD}}\right)$ and Calmar $\left(\frac{\mathbb{E}[\text{Returns}]}{\text{Downside Deviation}}\right)$ ratios are used as a gauge to measure risk-adjusted performance. We also include the average profit divided by the average loss $\left(\frac{\text{Avg. Profits}}{\text{Avg. Loss}}\right)$.
4) Ranking Performance: Kendall's Tau, Normalised Discounted Cumulative Gain at k (NDCG@k) [36], which is suited for non-binary relevance (scoring) measures while also emphasising top returned results [13]. We note that k is a pre-defined threshold, which we set fix at $k = 100$ in our paper to cover the size of each of our long/short portfolios.

D. Results and Discussion

To study the out-of-sample performance across various strategies, we chart their cumulative returns in Exhibit 1 and tabulate key measures of financial performance in Exhibit 2. To allow for better comparability between strategy performance, we also apply an additional layer of volatility scaling at the portfolio level, bringing overall returns for each strategy in line with our 15% target. All returns in this section are computed without transaction costs to focus on the raw predictive ability of the models. From both the plot and statistics, it is evident that our proposed class of LTR algorithms out-performs the set of benchmarks on all measures of performance – with LambdaMART placed at the top for most metrics.

Exhibit 1: Cumulative Returns - Rescaled to Target Volatility.



In terms of profitability, the rankers significantly improve expected returns and the percentage win rate. The 'worst' LTR model significantly outperform the best reference benchmark for each metric considered. While all models have been rescaled to trade around similar levels of volatility, LTR based strategies come across as being less subjected to huge drawdowns and downside risks. On a performance basis, there is again an identical pattern of the lowest ranker dominating the best benchmark, and the best LTR model demonstrating substantial gains across various performance-based measures. This clear disparity in performance underscores the importance of learning the cross-sectional rankings as it leads to better performance for the momentum strategy. Further analysing the relative performance of models within each group, we first note that there is no clear superiority of the listwise LTR algorithms over their pairwise counterparts. One might have assumed that the listwise methods emerge more performant since they learn the broader listwise structure, which the results show is not necessarily the case. This might be explained by the inherently poor signal-to-noise ratio typical of financial datasets, which is further exacerbated by the limited size of the data used – specifically, the listwise approaches use approximately $12 \times N_{avg}$ samples per year while the pairwise methods

Exhibit 2: Performance Metrics – Rescaled to Target Volatility.

	Benchmarks				Learning to Rank Models			
	Rand	JT	Baz	MLP	RNet	LM	LNet	LMLE
E[returns]	0.024	0.092	0.112	0.044	0.243	***0.359**	0.306	0.260
Volatility	0.156	0.167	0.161	0.165	0.162	0.166	***0.155**	0.162
Sharpe	0.155	0.551	0.696	0.265	1.502	***2.156**	1.970	1.611
Downside Dev.	0.106	0.106	0.097	0.112	0.081	***0.067**	0.068	0.071
MDD	0.584	0.328	0.337	0.641	0.294	***0.231**	0.274	0.236
Sortino	0.228	0.872	1.157	0.389	3.012	***5.321**	4.470	3.647
Calmar	0.042	0.281	0.333	0.068	0.828	***1.555**	1.115	1.102
% +ve Returns	0.545	0.582	0.591	0.551	0.693	***0.762**	0.715	0.681
Avg. P / Avg. L	0.947	1.114	1.184	1.001	1.407	1.594	***1.679**	1.534

Exhibit 3: Ranking Metrics – Average over all Rebalancing Months.

	Benchmarks				Learning to Rank Models			
	Rand	JT	Baz	MLP	RNet	LM	LNet	LMLE
Kendall's Tau	0.000	0.016	0.013	0.008	0.032	0.032	***0.033**	0.020
NDCG@100 (Longs)	0.549	0.555	0.562	0.550	0.576	0.576	***0.578**	0.565
NDCG@100 (Shorts)	0.552	0.562	0.555	0.564	0.575	***0.585**	0.579	0.567

have access to $12 \times N_{avg}^2$, where N_{avg} is the average size of the cross-sectional universe at each month. Across benchmarks, the Random model performed the worst as expected while the results of MLP are only marginally better. We suspect that this might also be the consequence of working with limited and noisy data which leads to overfitting, as well as the sub-optimality of the regress-then-rank approach utilised by MLP. Furthermore, the computed scores of MLP are essentially forecasts of (monthly) returns, which is regarded as a challenging problem [47] that is made even more so when all models used in this work are restricted to only using price-based data (See Section V-A).

By measuring an item's quality using *graded* relevance and applying a discount over weights, the NDCG is well suited for assessing the quality of top-ranked items [48] and is thus a widely used metric in the search literature [10]. The NDCG is particularly appropriate for the CSM strategy since it determines the extent to which models are able to accurately rank stocks by profitability – a model that is able to rank in a more precise manner makes it likelier that top-ranked assets get selected for inclusion in the respective long/short portfolio. Given this, we assess all models based on the NDCG and set the cutoff $k = 100$ to match the size of each of our long/short[1] portfolios. From the set of compiled ranking metrics that is averaged across all months in Exhibit 3, all LTR models surpass the benchmarks when measured using NDCG@100 – highlighting their ability to produce rankings that are more accurate, leading to better out-of-sample portfolio performance. With Kendall's Tau (rank correlation coefficient), we also see the same pattern of out-performance, noting that this time ranking quality is assessed across the *entire* list of assets.

To further examine how rankings quality is linked to out-of-sample results, we construct long-only decile portfolios: At each month, these are formed by partitioning the asset universe into equally weighted deciles based on the signals/scores produced by their respective models. For instance, assets in the top (long) decile for MLP would contain the highest 10%[2] of the model's predictions. Returns are computed similarly to Equation (1) but with a decile membership indicator $D_{\tau_m}^{(i)}$ used in place of $X_{\tau_m}^{(i)}$:

$$r_{\tau_m,\tau_{m+1}}^{DEC} = \frac{1}{n_{\tau_m}} \sum_{i=1}^{n_{\tau_m}} D_{\tau_m}^{(i)} \frac{\sigma_{tgt}}{\sigma_{\tau_m}^{(i)}} r_{\tau_m,\tau_{m+1}}^{(i)} \tag{12}$$

where $D_{\tau_m}^{(i)} = \{0, 1\}$ and takes the value of 1 for the decile of interest and 0 otherwise. We also perform an additional level of scaling at the portfolio level. Referring to the summary results (Exhibit 4), there is a general trend of returns and Sharpe increasing from decile 1 to 10 across all strategies except Random which emphasises the

[1]To compute NDCG for shorts, we reversed our relevance scores which allows the most negative returns to attain the highest scores.

[2]This differs from our earlier approach of using 100 instruments for each long and short portfolios at all times.

Exhibit 4: Performance Metrics – Decile Portfolios Rescaled to Target Volatility

	Decile										
	1	2	3	4	5	6	7	8	9	10	$L-S$
Rand											
E[returns]	0.110	0.120	0.118	0.124	0.114	0.123	0.126	0.115	0.126	0.115	0.028
Volatility	0.162	0.163	0.164	0.162	0.164	0.162	0.163	0.161	0.163	0.164	0.156
Sharpe	0.675	0.737	0.721	0.769	0.697	0.763	0.772	0.710	0.772	0.702	0.177
JT											
E[returns]	0.059	0.071	0.096	0.108	0.122	0.127	0.137	0.151	0.146	0.146	0.094
Volatility	0.165	0.164	0.165	0.164	0.163	0.163	0.163	0.162	0.160	0.156	0.167
Sharpe	0.360	0.435	0.581	0.661	0.746	0.780	0.840	0.928	0.910	0.938	0.565
Baz											
E[returns]	0.095	0.094	0.083	0.094	0.092	0.109	0.124	0.137	0.160	0.185	0.107
Volatility	0.163	0.163	0.162	0.162	0.163	0.162	0.162	0.162	0.163	0.163	0.161
Sharpe	0.582	0.573	0.510	0.579	0.566	0.671	0.765	0.849	0.984	1.130	0.664
MLP											
E[returns]	0.072	0.112	0.122	0.114	0.127	0.126	0.130	0.135	0.124	0.132	0.097
Volatility	0.163	0.161	0.163	0.162	0.163	0.162	0.163	0.163	0.163	0.164	0.168
Sharpe	0.443	0.697	0.751	0.703	0.780	0.779	0.800	0.825	0.758	0.806	0.578
RNet											
E[returns]	0.043	0.067	0.079	0.095	0.115	0.121	0.140	0.147	0.163	0.202	0.246
Volatility	0.164	0.164	0.164	0.164	0.164	0.163	0.161	0.162	0.161	0.163	0.161
Sharpe	0.263	0.405	0.480	0.580	0.698	0.742	0.870	0.906	1.014	1.238	1.527
LM											
E[returns]	0.012	0.074	0.097	0.098	0.117	0.132	0.131	0.141	0.171	0.201	0.349
Volatility	0.164	0.164	0.162	0.162	0.164	0.164	0.164	0.162	0.162	0.163	0.165
Sharpe	0.075	0.449	0.599	0.606	0.716	0.806	0.800	0.868	1.053	1.232	2.107
LNet											
E[returns]	0.037	0.069	0.089	0.102	0.116	0.117	0.137	0.152	0.151	0.186	0.296
Volatility	0.161	0.165	0.162	0.163	0.164	0.162	0.164	0.162	0.163	0.162	0.155
Sharpe	0.232	0.416	0.549	0.628	0.711	0.720	0.837	0.938	0.929	1.148	1.911
LMLE											
E[returns]	0.059	0.080	0.088	0.110	0.109	0.123	0.126	0.151	0.163	0.193	0.244
Volatility	0.165	0.164	0.164	0.165	0.161	0.164	0.162	0.161	0.160	0.163	0.160
Sharpe	0.360	0.489	0.537	0.671	0.677	0.750	0.782	0.937	1.016	1.186	1.530

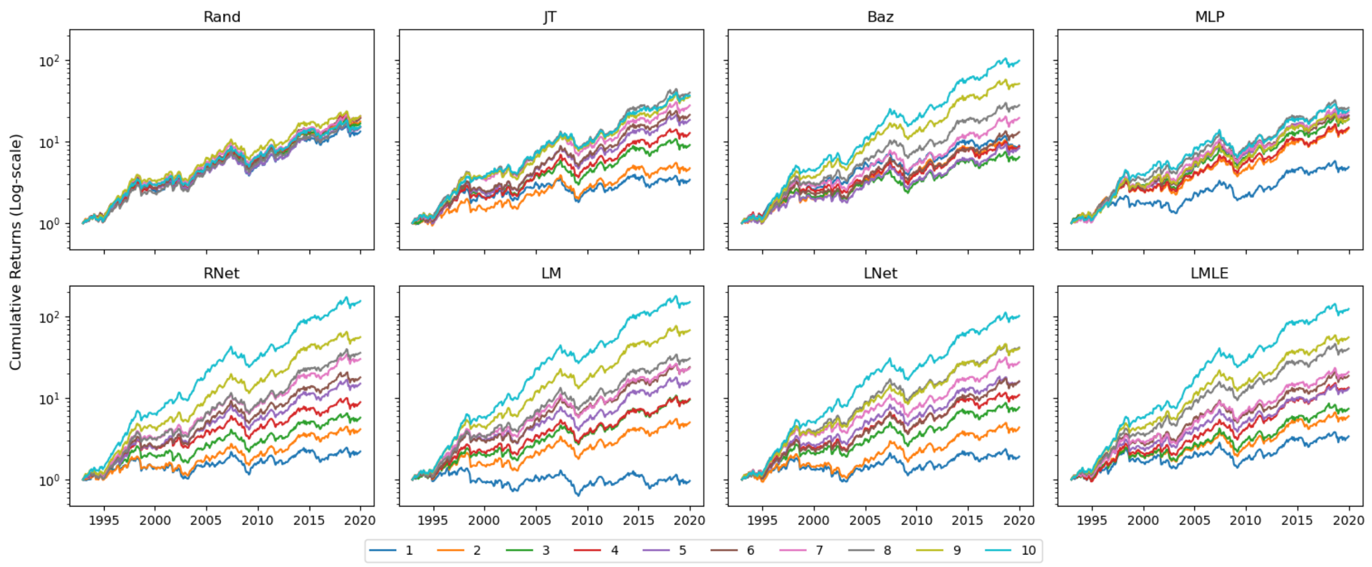
consistency of the momentum factor. More important is the steeper rise of these figures for the LTR models stemming from their ability to place assets in their appropriate deciles with a greater degree of precision – leading to a greater difference in returns between the decile 1 and decile 10 portfolios. The plot of decile portfolio returns across strategies (Exhibit 5) cement this observation, illustrating the connection between the model's ranking ability and the dispersion across return streams. This relationship is most pronounced for the group of LTR models which echoes the preceding statistics, thus validating our hypothesis that better asset rankings improves strategy performance.

VI. Conclusions

Focusing on cross-sectional momentum as a demonstrative use-case, we introduce Learning to Rank algorithms as a novel way of ranking assets which is an important step required by cross-sectional systematic strategies. Additionally, the modular framework underpinning these algorithms allows additional feature inputs to be flexibly incorporated – providing a generalisable platform for a broader set of cross-sectional strategies. In learning the relational (pairwise or listwise) structure across assets that both heuristics and regress-then-rank approaches only superficially capture, we obtain a more accurate ranking across instruments. This translates to Sharpe Ratios being boosted approximately threefold over traditional approaches, as well providing clear and significant improvements to both performance and ranking-quality related measures.

Some directions for future work include innovating in terms of architecture or model ensembling to further improve strategy performance, as well as studying the

Exhibit 5: Cumulative Returns - Decile Portfolios Rescaled to Target Volatility



effectiveness of these ranking techniques on higher frequency data (e.g. order-book) and asset classes.

References

[1] J. Baz, N. M. Granger, C. R. Harvey, N. Le Roux, and S. Rattray, "Dissecting Investment Strategies in the Cross Section and Time Series," *SSRN Electronic Journal*, 2015.

[2] T. J. Moskowitz, Y. H. Ooi, and L. H. Pedersen, "Time series momentum," *Journal of Financial Economics*, vol. 104, no. 2, pp. 228–250, May 2012.

[3] N. Jegadeesh and S. Titman, "Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency," *The Journal of Finance*, vol. 48, no. 1, pp. 65–91, Mar. 1993.

[4] T. Roncalli, "Keep up the momentum," *SSRN Electronic Journal*, 2017.

[5] P. Jusselin, E. Lezmi, H. Malongo, C. Masselin, T. Roncalli, and T.-L. Dao, "Understanding the momentum risk premium: An in-depth journey through trend-following strategies," *SSRN Electronic Journal*, 2017.

[6] S. Kim, "Enhancing the momentum strategy through deep regression," *Quantitative Finance*, vol. 19, no. 7, pp. 1121–1133, Jul. 2019.

[7] S. Gu, B. Kelly, and D. Xiu, "Empirical Asset Pricing via Machine Learning," National Bureau of Economic Research, Cambridge, MA, Tech. Rep. w25398, Dec. 2018.

[8] S. Gu, B. T. Kelly, and D. Xiu, "Autoencoder Asset Pricing Models," *SSRN Electronic Journal*, 2019.

[9] R. K. Pasumarthi, S. Bruch, X. Wang, C. Li, M. Bendersky, M. Najork, J. Pfeifer, N. Golbandi, R. Anil, and S. Wolf, "TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage AK USA: ACM, Jul. 2019, pp. 2970–2978.

[10] H. Li, "Learning to Rank for Information Retrieval and Natural Language Processing," *Synthesis Lectures on Human Language Technologies*, vol. 4, no. 1, pp. 1–113, Apr. 2011.

[11] B. Wang and D. Klabjan, "An Attention-Based Deep Net for Learning to Rank," *arXiv:1702.06106 [cs]*, Dec. 2017.

[12] P. Li, Z. Qin, X. Wang, and D. Metzler, "Combining Decision Trees and Neural Networks for Learning-to-Rank in Personal Search," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage AK USA: ACM, Jul. 2019, pp. 2032–2040.

[13] Q. Wu, C. J. C. Burges, K. M. Svore, and J. Gao, "Adapting boosting for information retrieval measures," *Information Retrieval*, vol. 13, no. 3, pp. 254–270, Jun. 2010.

[14] P. Wang, C. Liu, Y. Yang, and S. Huang, "A robo-advisor design using multiobjective ranknets with gated neural network structure," in *2019 IEEE International Conference on Agents (ICA)*, 2019, pp. 77–78.

[15] J. Rohrbach and S. Suremann, "Momentum in Traditional and Cryptocurrencies Made Simple," *SSRN Electronic Journal*, 2017.

[16] B. Lim, S. Zohren, and S. Roberts, "Enhancing Time Series Momentum Strategies Using Deep Neural Networks," *SSRN Electronic Journal*, 2019.

[17] K. G. Rouwenhorst, "International Momentum Strategies," *The Journal of Finance*, vol. 53, no. 1, pp. 267–284, 1998.

[18] J. M. Griffin, X. Ji, and J. S. Martin, "Momentum Investing and Business Cycle Risk: Evidence from Pole to Pole," *The Journal of Finance*, vol. 58, no. 6, pp. 2515–2547, 2003.

[19] A. C. W. Chui, S. Titman, and K. C. J. Wei, "Individualism and Momentum around the World," *The Journal of Finance*, vol. 65, no. 1, pp. 361–392, 2010.

[20] W. de Groot, J. Pang, and L. Swinkels, "The cross-section of stock returns in frontier emerging markets," *Journal of Empirical Finance*, vol. 19, no. 5, pp. 796–818, 2012.

[21] C. B. Erb and C. R. Harvey, "The Strategic and Tactical Value of Commodity Futures," *Financial Analysts Journal*, vol. 62, no. 2, pp. 69–97, Mar. 2006.

[22] B. LeBaron, "Technical Trading Rule Profitability and Foreign Exchange Intervention," National Bureau of Economic Research, Cambridge, MA, Tech. Rep. w5505, Mar. 1996.

[23] C. Pirrong, "Momentum in Futures Markets," *SSRN Electronic Journal*, 2005.

[24] L. Wang and K. Rasheed, "Stock ranking with market microstructure, technical indicator and news," in *Proceedings on the International Conference on Artificial Intelligence (ICAI)*. The Steering Committee of The World Congress in Computer Science, Computer ..., 2018, pp. 322–328.

[25] T.-Y. Liu, *Learning to Rank for Information Retrieval*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[26] G. Cloud, "Cloud TPU documentation — Google Cloud," https://cloud.google.com/tpu/docs.

[27] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[29] S. K. K. Santu, P. Sondhi, and C. Zhai, "On Application of Learning to Rank for E-Commerce Search," *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 475–484, Aug. 2017.

[30] B. L. Pereira, A. Ueda, G. Penha, R. L. T. Santos, and N. Ziviani, "Online learning to rank for sequential music recommendation," in *Proceedings of the 13th ACM Conference on Recommender Systems*. Copenhagen Denmark: ACM, Sep. 2019, pp. 237–245.

[31] Q. Song, A. Liu, and S. Y. Yang, "Stock portfolio selection using learning-to-rank algorithms with news sentiment," *Neurocomputing*, vol. 264, pp. 20–28, Nov. 2017.

[32] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of Econometrics*, vol. 31, no. 3, pp. 307–327, April 1986. [Online]. Available: https://ideas.repec.org/a/eee/econom/v31y1986i3p307-327.html

[33] P. Nguyen, J. Wang, and A. Kalousis, "Factorizing LambdaMART for cold start recommendations," *Machine Learning*, vol. 104, no. 2, pp. 223–242, Sep. 2016.

[34] S. Bruch, "An Alternative Cross Entropy Loss for Learning-to-Rank," *arXiv:1911.09798 [cs, stat]*, May 2020.

[35] C. J. C. Burges, R. Ragno, and Q. V. Le, "Learning to rank with nonsmooth cost functions," in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, ser. NIPS'06. Cambridge, MA, USA: MIT Press, 2006, pp. 193–200.

[36] K. Järvelin and J. Kekäläinen, "IR evaluation methods for retrieving highly relevant documents," in *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '00*. Athens, Greece: ACM Press, 2000, pp. 41–48.

[37] P. Donmez, K. M. Svore, and C. J. Burges, "On the local optimality of LambdaRank," in *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '09*. Boston, MA, USA: ACM Press, 2009, p. 460.

[38] Y. Yue and C. J. Burges, "On using simultaneous perturbation stochastic approximation for learning to rank, and the empirical optimality of LambdaRank," Microsoft Research, Tech. Rep. MSR-TR-2007-115, Aug. 2007.

[39] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li, "Listwise approach to learning to rank: Theory and algorithm," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: Association for Computing Machinery, 2008, pp. 1192–1199.

[40] S. Boyd and L. Vandenberghe, *Convex Optimization*. USA: Cambridge University Press, 2004.

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[42] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, "Hyperopt: A Python library for model selection and hyperparameter optimization," *Computational Science & Discovery*, vol. 8, no. 1, p. 014008, Jul. 2015.

[43] T. U. o. C. B. S. o. B. Center for Research in Security Prices (CRSP), "Nyse equity data from 1980 to 2019," *Calculated (or Derived) based on data from CRSP Daily Stock ©2019*, 2019. [Online]. Available: https://wrds-web.wharton.upenn.edu/wrds/

[44] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd International Conference on Machine Learning*, ser. ICML '05. New York, NY, USA: Association for Computing Machinery, 2005, pp. 89–96.

[45] C. J. C. Burges, "From RankNet to LambdaRank to LambdaMART: An overview," Microsoft Research, Tech. Rep. MSR-TR-2010-82, 2010.

[46] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: From pairwise approach to listwise approach," in *Proceedings of the 24th International Conference on Machine Learning - ICML '07*. Corvalis, Oregon: ACM Press, 2007, pp. 129–136.

[47] A. Naccarato, A. Pierini, and G. Ferraro, "Markowitz portfolio optimization through pairs trading cointegrated strategy in long-term investment," *Annals of Operations Research*, Apr. 2019.

[48] Y. Wang, L. Wang, Y. Li, D. He, and T.-Y. Liu, "A theoretical analysis of NDCG type ranking measures," in *Proceedings of the 26th Annual Conference on Learning Theory*, ser. Proceedings of Machine Learning Research, S. Shalev-Shwartz and I. Steinwart, Eds., vol. 30. Princeton, NJ, USA: PMLR, Jun. 2013, pp. 25–54.

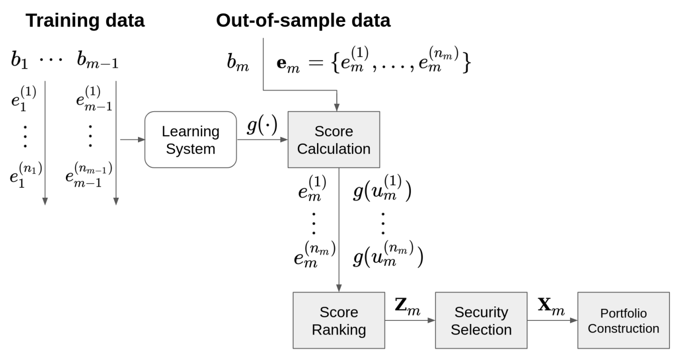
[49] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785

VII. Appendix

A. Learning to Rank For Cross-sectional Momentum

Learning to rank (LTR) is a supervised learning task involving training and testing phases. Document retrieval is the standard problem setting to make concrete the LTR framework, and we follow this convention.

Exhibit 6: Learning to Rank for Cross-Sectional Momentum



For training, we are provided with a set of queries $\boldsymbol{\mathcal{Q}} = \{q_1, ..., q_m\}$. Each query q_i has an associated list of documents $\boldsymbol{d}_i = \{d_i^{(1)}, ..., d_i^{(n_i)}\}$ where n_i is the total number of documents for q_i, and an accompanying set of document labels $\boldsymbol{y}_i = \{y_i^{(1)}, ..., y_i^{(n_i)}\}$ where the labels represent grades. Letting $\boldsymbol{\mathcal{Y}} = \{\mathcal{Y}_1, ..., \mathcal{Y}_\ell\}$ be the label set, we have $y_i^{(j)} \in \boldsymbol{\mathcal{Y}}$ for $\forall\ i, j$. We also have $\mathcal{Y}_\ell \succ \mathcal{Y}_{\ell-1} \succ ... \succ \mathcal{Y}_1$ where $\succ$ stands for the order relation – a higher grade on a given document implies a stronger relevance of the document with respect to its query. For each query-document pair, a feature vector $x_i^{(j)} = \phi\big(q_i, d_i^{(j)}\big)$ can be formed, noting that $\phi(\cdot)$ is a feature function, $i \in \{1, ..., m\}$ and $j \in \{1, ..., n_i\}$. Letting $\boldsymbol{x}_i = \{x^{(1)}, ..., x^{(n_i)}\}$, we can assemble the training set $\big\{\boldsymbol{x}_i, \boldsymbol{y}_i\big\}_{i=1}^{m}$. The goal of LTR is to learn a function f that predicts a score $f(x_{m+1}^{(i)}) = z_{m+1}^{(i)}$ when presented with an out-of-sample input $x_{m+1}^{(i)}$. For more details, we point the reader to [10].

Transposing the preceding framework to the momentum strategy, we can treat each query as being analogous to a portfolio rebalancing event, while an associated document and its accompanying label can be respectively thought of as an asset and its assigned decile at the *next* rebalance based on some performance measure (conventionally taken to be returns). Exhibit 6 provides a schematic of this adaptation which we further make concrete. For training, let $\boldsymbol{\mathcal{B}} = \{b_1, ..., b_{m-1}\}$ be a series of monthly rebalances where at each b_i we have a set of equity instruments $\boldsymbol{e}_i = \{e_i^{(1)}, ..., e_i^{(n_i)}\}$ and the set of assigned deciles $\boldsymbol{\delta}_{i+1} = \{\delta_{i+1}^{(1)}, ..., \delta_{i+1}^{(n_i)}\}$ where $\delta_{i+1}^{(j)} \in \boldsymbol{\mathcal{D}} = \{\mathcal{D}_1, ..., \mathcal{D}_{10}\}$ for $\forall\ i, j$. Similar to above, $k > l \Rightarrow \mathcal{D}_k \succ \mathcal{D}_l$ for $k, l \in \{1, ..., 10\}$. With each rebalance-asset pair, we can form the feature vector $u_i^{(j)} = \phi(b_i, e_i^{(j)})$ as well as the broader training set $\big\{\boldsymbol{u}_i, \boldsymbol{\delta}_{i+1}\big\}_{i=1}^{m-1}$ where $\boldsymbol{u}_i = \{u_i^{(1)}, ..., u_i^{(n_i)}\}$. Note that other features can be incorporated for $u_i^{(j)}$ – allowing different types of cross-sectional strategies to be developed. Presented with set of feature vectors for testing at interval m with a trained function g produced by the Learning System, we compute the set of scores $g(\boldsymbol{u}_m)$ at the Score Calculation phase and then go on to form the long/short portfolios by following the instructions in Section III-A.

B. Additional Training Details

Python Libraries: LambdaMART uses `XGBoost` [49], while the others – RankNet, ListNet, ListMLE are developed using `Tensorflow` [27].

Hyperparameter Optimisation: Hyperparameters assume discrete values and are tuned using `HyperOpt` [42]. For LambdaMART, we refer to the hyperparameters as they are named in the `XGBoost` library.

Multi-layer Perceptron (MLP)

- Dropout Rate – [0.0, 0.2, 0.4, 0.6, 0.8]
- Hidden Width – [64, 128, 256, 512, 1024, 2048]
- Max Gradient Norm – $[10^{-3}, 10^{-2}, 10^{-1}, 1, 10]$
- Learning Rate – $[10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1]$
- Minibatch Size – [64, 128, 256, 512 1024]

RankNet

- Dropout Rate – [0.0, 0.2, 0.4, 0.6, 0.8]
- Hidden Width – [64, 128, 256, 512, 1024, 2048]
- Max Gradient Norm – $[10^{-3}, 10^{-2}, 10^{-1}, 1, 10]$
- Learning Rate – $[10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1]$
- Securities used to form Pairs for Minibatch – [64, 128, 256, 512 1024]

LambdaMART

- 'objective' – 'rank:pairwise'
- 'eval_metric' – 'ndcg'
- 'eta' – $[10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1]$
- 'num_boost_round' – $[5, 10, 20, 40, 80, 160, 320]$
- 'max_depth' – $[2, 4, 6, 8, 10]$
- 'tree_method' – 'gpu_hist'

ListMLE, ListNet

- Dropout Rate – [0.0, 0.2, 0.4, 0.6, 0.8]
- Hidden Width – [64, 128, 256, 512, 1024, 2048]
- Max Gradient Norm – $[10^{-3}, 10^{-2}, 10^{-1}, 1, 10]$

- Learning Rate – $[10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}]$
- Minibatch Size – [1, 2, 4, 8, 16]