High-performance stock index trading via neural networks and trees

Chariton Chalvatzis[a,*], Dimitrios Hristu-Varsakelis[1,a]

[a] *University of Macedonia, Department of Applied Informatics, 156 Egnatia St., Thessaloniki, 54006, Greece*

Abstract

Automated asset trading typically involves a price prediction model - of as high an accuracy as possible - together with a trading strategy, sometimes as simple as buying or selling when the price is predicted to rise or fall, respectively. Despite the fact that the model's effectiveness in generating profits may depend on the particular trading strategy it is used with, these two components are often designed separately, in part because of the difficulty involved in jointly optimizing them. Motivated by this interplay between model performance and trading strategy, this work presents a novel automated trading architecture in which the prediction model is tuned to enhance profitability instead of accuracy, while the trading strategy attempts to be more sophisticated in its use of the model's price predictions. In particular, instead of acting simply on whether the price is predicted to rise or fall we show that there is value in taking advantage of the model-specific distribution of predicted returns, and the fact that a prediction's position within that distribution carries useful information about the expected profitability of a trade. Our proposed approach was tested with tree-based models as well as one deep long short-term memory (LSTM) neural networks, all of which were kept structurally simple and generated predictions based on price observations over a modest number of trading days. Tested over the period 2010-2019 on the S&P 500, Dow Jones Industrial Average (DJIA), NASDAQ and Russell 2000 stock indices, and our best overall model achieved cumulative returns of 350%, 403%, 497% and 333%, respectively, outperforming the benchmark buy-and-hold strategy as well as other recent efforts.

Keywords: Finance, LSTM, Deep Learning, Stock Prediction, Automatic trading

1. Introduction

Equity prediction lies at the core of the investment management profession and has also attracted significant attention from academia. One of the long-standing questions has been how to forecast the behavior of stocks and then act accordingly to generate profit in excess of that generated by the

[*]Corresponding author

Email addresses: `charis.chalvatzis@uom.edu.gr` (Chariton Chalvatzis), `dcv@uom.gr` (Dimitrios Hristu-Varsakelis)

market itself [1]. Towards that end, significant effort has been put into predicting the price of major U.S. stock indices [2, 3] or individual stocks, with techniques ranging from early linear models [4], to machine learning and neural network-based approaches [5]. There are two components which are generally part of the overall decision-making process when it comes to "intelligent" or automatic trading: a *predictive model* whose purpose is to anticipate an asset's future price, and a *trading strategy* which uses the model's predictions to generate profit. Developing a process with consistently superior performance is difficult, in part because of the "joint" nature of the goal: profits ultimately depend both on the prediction scheme as well as the trading strategy it is used with, and changing either of the two affects the final outcome. Thus, in principle, these two components should be designed and optimized together, a challenge which currently appears to be out of reach, because of the seemingly endless variety of possible prediction schemes and trading strategies. Instead, most of the relevant research (to be reviewed shortly) has focused on models that do quite well by improving on specific aspects of the overall prediction/trading process.

One approach has been to aim for a prediction scheme which is as good as possible at guessing the "next" price of an asset, and then use that guess to inform trading decisions [6, 7, 8]. This is frequently done by means of a so-called *directional* "up-down" trading strategy which buys the asset when the predicted price is higher than the current price and sells if it is lower. While this approach can be effective, it is based on an implicit assumption that better short-term price predictions lead to higher profitability. This is not true in general: prediction accuracy is typically measured with a *symmetric* loss function (e.g., mean absolute error) which is indifferent to the sign of the prediction error, and yet that sign may be quite important when trading. In fact, it is possible for one prediction scheme to lead to trades which are *always* profitable while another, having *better* predictive accuracy, to record only losses[1]. An alternative is to forgo predictions of the asset price itself and instead consider a *directional* model which predicts whether the price will rise or fall compared to its current level, essentially acting as a binary classifier [9, 10]. Then, the trading strategy is again to simply buy or sell the asset based on the classifier's recommendation. This approach is self-consistent from the point of

[1]Consider for example the four-day price series $Y = [101, 100, 98, 101]$, and two possible predictions of its last three samples, $P_1 = [95, 92, 105]$ and $P_2 = [102, 101, 97]$, both generated on the first day, when the price was 101. P_1 has a mean absolute error of 5 when measured against the last three samples of Y, while P_2's error is 3. However, P_1 is always correct on the direction of the price movement of Y, i.e., its predictions are always higher (resp.lower) than the previous price of Y when the price will indeed rise (resp. fall) in the next sample, while P_2 is always wrong.

view that predictions are judged in the same setting as the trading strategy, i.e., being correct on the direction of price movement.

While one can of course consider many possible variations on trading strategies and accompanying prediction schemes, there are two key points to be recognized. First, predictions should be viewed and evaluated *in the context* of the trading strategies they are used with [11]. Thus, prediction accuracy should not monopolize one's focus: large(r) prediction errors do not necessarily preclude a technique from being profitable just like small errors do not by themselves guarantee profitability. Second, because the joint model/strategy optimum appears to be elusive, one may try to improve overall profitability by *adapting* the trading strategy to make fuller use of the information contained within the set of predictions; that information may generally be much more than whether the price will rise or fall in the next time step.

This paper's contribution is a novel, systematic, model-agnostic approach to exploiting the potential profitability of predictions. The approach consists of i) a trading strategy that is effective in generating profits with reasonable trading risk, and ii) a price prediction model which is tuned for profitability instead of accuracy as typically done in the literature. The model's predictions are based on a rolling window of past price data and are fed to a novel event-based[2] trading strategy which takes advantage of the information available not in any single price prediction, but in the entire *distribution* of predicted returns. It does so by not trading simply on whether the price is predicted to rise or fall, but rather based on the expected profitability of a possible trade, conditioned on a prediction's relative position within the aforementioned distribution. As we will see, such a strategy can be optimized in some ways, leading to significant gains, with results that compare favorably to standard benchmarks as well as recent works.

Because our approach is not tied to any specific class of prediction models, we will demonstrate it on some of the popular and commonly used architectures from literature, including tree-based models and recurrent neural networks (specifically deep long short-term memory networks - LSTMs). One of the LSTM architectures tested will include an innovative feature which allows the output layer to access to the entire evolution of the network's hidden states as the input layer is "exposed" to a

[2]We will use the term "event" in the sense of probability theory (e.g., a sample of a random variable falling within a specified interval on the real line), as opposed to finance where it commonly refers to "external" events (corporate filings, legal actions, etc.).

sequence of past prices, leading to performance benefits, as we will see. We will test our proposed approach on major U.S. stock indices, namely the S&P 500, the Dow Jones Industrial Average (DJIA), the NASDAQ and the Russel 2000 (R2000), over an almost decade-long period (Jan. 2010 - Dec. 2019). These indices attract significant attention from market practitioners, both in terms of strategy benchmarking and for monitoring market performance. They are appealing because they can be traded indirectly - via their corresponding tracking exchange-traded funds (ETFs) - at zero or negligible cost as we will discuss in the sequel, and will also serve as a common ground for comparison with other studies. Moreover, our relatively long testing period will allow us to draw safer conclusions as to our scheme's profitability over time.

The remainder of the paper is organized as follows. Section 2.1 discusses relevant literature, including details on profitability and overall performance. In Section 3 we describe our proposed model and its components, as well as a class of capital allocation policies which are optimal for the model. Section 4 discusses model training and allocation policy selection. The profitability and overall performance of our approach, as well as comparisons with recent works and standard benchmarks, are discussed in Section 5.

2. Literature Review

We can identify two significant clusters in the literature which are relevant to this work because of their prediction architecture and corresponding trading strategy which aim to either achieve high profitability or beat well-established indices.

2.1. Asset Value Prediction - Directional Trading Strategy

The first cluster contains studies in which prediction and profitability evaluation are "decoupled" in the sense discussed in the previous Section: a model is trained to predict the value of an asset, but that prediction is then used with a "directional" trading strategy. Studies in this category include [6] whose stacked auto-encoder and LSTM-based model achieved daily returns with a yearly average sum of approximately 46% and 65% for the S&P 500 and DJIA, respectively, over the period 2010-2016. In [7] the authors introduced a modular architecture consisting of two LSTM networks; using a 20 trading day window for the S&P500, they reported low prediction errors and a 96.55% cumulative return over the period 2/1/2008–7/26/2017 (we will conventionally use the U.S. format, *mm/dd/yyyy* for reporting dates throughout), beating the benchmark buy-and-hold (BnH)[3] strategy (78.74%).

Several studies have applied feed-forward or convolutional neural networks (CNN) to financial time-series. A recent example is [8], which reported a 1.05% mean percentage error for the S&P500, over a one-year test period (the year 2011). Combining their predictions with a long-short[4] trading strategy, the authors achieved an average annualized return (per-year geometric average of returns) of 25% and a Sharpe ratio (a measure of reward-to-risk performance) of 2.6. An interesting CNN-based approach is the image-based model of [12]. Trading on well-known exchange-traded funds (ETFs) as well as the constituents of the DJIA, that approach could yield an annualized return of 13.01% for the ETF portfolio and 12.59% for the DJIA constituents portfolio, over 2007-2017. Finally, [3] used a portfolio of stocks to outperform a stock index, as opposed to trading the index itself. That work tested several machine learning models, including deep neural networks, gradient-boosted trees and random forests, in order to perform statistical arbitrage on the S&P500. Before transaction costs, the authors reported mean daily excess returns of 0.45% for their ensemble model during 12/1992-10/2015, however returns were negative during that period's last five years (2010-2015).

2.2. Asset Direction Prediction - Directional Trading Strategy

A second important cluster of works contains studies in which the loss function used to train the prediction model is "consistent" with the trading strategy, e.g., the model predicts the *direction* of the price movement - but not the numerical value of the price - and is used with a directional trading strategy. The work in [13] presents a hybrid trading system which combines a variation of support vector machines and random forests to generate three trading signals (buy/hold/sell). That method, applied over the period 1/2007-12/2015 in which the last 500 days were used as a test set, yielded cumulative returns of 24.46%, 27.67%, 18.30%, 43.81%, and 43.30%, for the Dow Jones, NASDAQ, S&P500, NIFTY 50, and NIFTY Bank indices respectively, with relatively small draw-downs. Another interesting study is [14], where the authors develop a learning architecture under which the trading data are first fed into a logistic regression model whose output is then fed into

[3]The so-called *buy-and-hold* portfolio is formed simply by buying the asset under consideration on the first day of the period under consideration, and selling it on the last day.

[4]A long-short trading strategy is one in which in addition to buying, one is also allowed to sell short (borrowed) assets.

gradient boosted decision trees in order to ultimately predict the direction of daily stock index changes. The authors report very strong Sharpe Ratio results of 2.6, 5.01 and 8.87, after transaction costs, for the S&P 500, NASDAQ and Shanghai Stock Exchange Composite Index, respectively. Their test set was 6/29/2016-12/26/2016 for the first two indices and 08/07/2014-12/31/2014 for the latter.

The work in [2] described a neural network which used only two features to predict the direction of next-day price movement for the largest 100 stocks in the S&P 500. A portfolio of those stocks was then formed to obtain a 16.8% annualized return using data from 2006-2013 and assuming a cost of 0.02% per transaction. Recently, [9] also studied the daily direction of the S&P 500 constituents between 1/1/1990 – 9/30/2015, comparing various machine learning techniques in order to estimate the probability for each stock to out-/under-perform the cross-sectional median. The authors reported mean daily returns of 0.46% and a Sharpe Ratio of 5.8, which indicates low risk / high reward prior to transaction costs. However, the returns were not consistent throughout the testing period, and were close to zero during its last 5 years. The work in [10] also studied the daily direction of the S&P 500 index using 60 input features and various dimensionality reduction techniques in order to reach a 58.1% prediction accuracy over 6/1/2003-5/31/2013. Using a trading strategy which either bought the S&P 500 or invested in a one-month T-bill resulted in a mean daily return of 0.1% with a standard deviation of 0.7% over a test period of 377 days ending on 5/31/2013.

2.3. Other approaches

There is also a broader cluster of studies which leverage additional information, including events, text data or crowd recommendations, for the purpose of stock prediction and trading. The model proposed in [15] generates "buy-sell-hold" decisions, if the the forecasting trading signal, extracted using support vector regression, intersects with dynamic thresholds, generated by a fuzzy rule-based model. This approach generated an average cumulative return of 18.08% using a training period from 1/2/2008-12/31/2008 and a testing period from 1/2/2009-6/30/2009. In [16] reinforcement learning was used to exploit the effects of seasonal events, such as exchange holidays. That approach was tested on the S&P 500 and the DAX from 2000 to 2012 and (neglecting transaction costs) achieved annualized and cumulative returns of 4.66% and 80.78%, respectively, for the S&P 500, (8.04% and 173.14%, respectively, for the DAX), in the average case. Other notable works include [17] and [18]. In the former principal components analysis is combined with neuroevolution augmenting topologies and applied to the S&P500 over the out of sample period of 1/30/2015 - 4/13/2017. The authors achieved a cumulative return of 18.89%. In [18], gradient boosting trees were used together with dimensionality reduction techniques and discrete wavelet transformation, to filter and reduce noise in the price time series. Using the S&P500 and a 555 trading day out of sample period (1/2015 - 4/2017) that approach achieved a 35.21% cumulative return.

Table 1 lists the works discussed above, together with the central method / AI architecture used therein, the type of objective considered, and the asset traded.

Prediction target	Trading strategy	Study cited	Method	Asset
Price	Up/Down	[6,7]	LSTM	Index
		[8]	FFN	portfolio
		[12]	CNN	ETF
		[3]	PCA	portfolio
Direction	Up/Down	[13]	SVM/RF	Index
		[14]	LR/GBT	Index
		[2]	FFN	portfolio
		[9]	LSTM/XGB/RF	portfolio
		[10]	PCA	Index
		[15]	Fuzzy model	Index
		[17,18]	Neuroevolution, Wavelet	Index
Event	RL	[16]	FFN	Index

Table 1: Summary of the studies cited, grouped according to the type of prediction undertaken (price/direction/other), type of trading strategy, main methodology/machine learning technique used, and asset traded. *Price* refers to models that are trained to predict asset price, *Direction* refers to models that predict the direction of the price movement (up/down), *Event* refers to more general events related to asset price. *LSTM*: Long Short-term memory, *FFN*: feed-forward networks, *CNN*: convolutional neural networks, *PCA*: Principal Components Analysis, *XGB*: gradient boosting, *RF*: random forests, *RL*: Reinforcement Learning. *Asset* specifies the type of asset used in each study: *Index* - the underlying traded asset is a stock index, *ETF* - an Exchange Traded Fund tracking the index, *portfolio*- a collection of stocks.

It is important to note that several of the works cited here report *average daily* returns, in some cases [6, 10] giving no annualized or cumulative returns. Unfortunately, the arithmetic average of returns is not informative as to actual capital growth/profitability, because it cannot be used to infer annual or cumulative return[5]. Also, as one might expect, it is difficult to declare a single "winner" among the different approaches, each of them being interesting in its own right: not all authors report the same

[5]For example, 1 euro invested in an asset which achieves consecutive returns of -99%, 100%, and 100%, would be worth just 4 cents, i.e., a cumulative return of -96%, while the average return would be approximately 33%.

profitability metrics, and the evaluation periods vary; even when comparing against the same asset and testing period, one cannot safely draw conclusions if the testing period is rather short (e.g., a single year, as in [8, 10]). Finally, there is also a significant number of works on the use of machine learning methods and neural networks in particular, where the goal is exclusively accurate price prediction, with no discussion of further use of that prediction, be it for trading or any other purpose. Representative papers include [19, 20]. The volume of research in this category is substantial but we will not delve deeper here because prediction accuracy by itself is not the focus of this work.

3. Prediction models and trading strategy

Our trading strategy will not make any explicit assumption on which prediction model it is used with. Here, we experimented with and compared the effectiveness of four different models, namely random forests, gradient boosted trees and two variations of LSTM networks. In principle, however, one could also use any other prediction model; we have limited ourselves to the aforementioned four because of space considerations. For our purposes, the feature vectors, x_t, will include an asset's daily adjusted close price, y_t, opening price, $y_t^{(O)}$, intra-day low, $y_t^{(L)}$, intra-day high, $y_t^{(H)}$, closing price, $y_t^{(C)}$, and the previous day's adjusted close, y_{t-1}, all obtained online [21], i.e.,

$$x_t = [y_t, y_t^{(O)}, y_t^{(L)}, y_t^{(H)}, y_t^{(C)}, y_{t-1}].$$

Operationally, the prediction model receives a sequence of numerical vectors (or batches thereof) $x_{t-T+1}, ..., x_t$, where T is a fixed time window size (for our purposes T will be measured in trading days), and produces a scalar output sequence of $\hat{y}_{t-T+2}, ..., \hat{y}_{t+1}$ which will represent the predicted price of a stock or other asset. We note that for each time t, the quantity of interest in the output sequence is its last element, $\hat{y}_{t+1}$ (i.e., the predicted price for the "next" time period), while the previous elements, $\hat{y}_{t-T+2}, ..., \hat{y}_t$, correspond to predictions for the prices $y_{t-T+2}, ..., y_t$, which are already known at time t. Although the inclusion of these output elements may seem superfluous, opting for sequence-to-sequence training will make for better predictive accuracy as well as profitability, as we will see further on.

3.1. *Random Forests and Gradient Boosting*

Random Forests (RF) have proved to be one of the earliest more powerful techniques in machine learning [22, 23, 24]. They are ensembles of decision trees and make use of two basic techniques: i) *bootstrap aggregation*, where trees are trained on different random subsets of the training set, usually with replacement, and ii) *feature bagging*, where only a random subset of input variables, known as features, is selected and evaluated, instead of looking for every possible best feature split, as the depth of the tree increases [25, 26]. These two techniques allow RFs to create trees with low correlation. Another popular and powerful technique is *boosting*, which refers to any ensemble method that combines weak learners into a stronger one. The principle that defines boosting is the training of a sequence of prediction models, such that each subsequent model learns from the errors of the previous one. There are various algorithms for achieving this, gradient boosting being one of the most popular ones: each model in the sequence is trained on the residual errors of the previous one [25]. Once the model has been fitted, its predictions, scaled by a constant known as the learning rate, are added to those of the previous model (the process resembles gradient descent and hence the name). This process gradually reduces the total error, so that adding up all of the "weak" prediction models in the sequence yields a more accurate prediction than any individual model. The learning rate controls the contribution of each model. Smaller values require more models to adequately fit the data. However, this process, known as shrinkage, usually leads to better generalization [26].

3.2. *Long short-term Memory Networks*

In addition to trees and gradient boosting we will consider two variations of LSTM networks as a means for predicting asset prices. At its core, the basic LSTM cell (whose mathematical description can be found, for example, in [27]) consists of various nonlinear transformations involving its time-varying hidden state, h_t, an internal memory state, c_t, and a batch of length-I input vectors, x_t, which the cell receives at each time $t = 0, 1, 2, \dots,$. The input, x_t, together with the previous hidden state h_{t-1} and other internal cell data, produce the next instances of the hidden and memory states via both feed-forward and recurrent connections.

The unrolled architecture of one of our LSTM networks used to predict asset prices is depicted in Fig. 1-(a); it is a deep network following the popular architecture [28] commonly used in recurrent neural network applications [29, 30]. We will opt for a Rectified Linear Unit (ReLU) activation

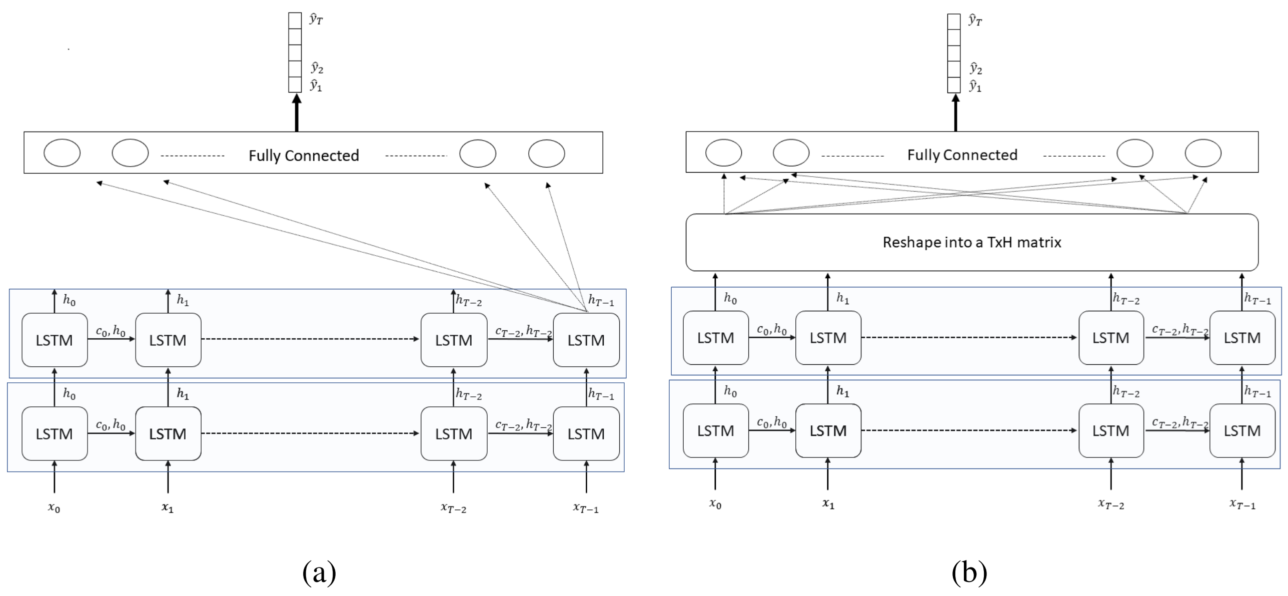

Figure 1: (a): LSTM-LH architecture, using only the last hidden state to produce the output sequence; (b): LSTM-AH variant, using all hidden states to produce the output. x_i denotes the input vector, c_i are....h_i are the instances of the hidden state, T is the length of the input sequence, H is the number of hidden units.

function, which departs from the use of tanh as a standard choice in LSTM cells, (e.g., [6, 9]); ReLU activation has been shown to be preferable to tanh in some cases, for example in CNNs [31], mainly in terms of training times. In our case, we found experimentally that it also led to better prediction and trading performance. As the LSTM cell's input is "driven" by an input sequence, $(x_{t-T+1}, ..., x_t)$, the output sequence $(\hat{y}_{t-T+2}, ..., \hat{y}_{t+1})$ is produced by "feeding" the *last hidden* state, h_t, through a fully connected layer. We will refer to this architecture as LSTM-LH.

We will also explore a variation of the above model, depicted in Fig. 1-(b) in which the *entire sequence* of hidden states are stacked together into one matrix which is then fed to the fully connected layer. This variant will be referred to as LSTM-AH. Utilizing the entire sequence will allow us to look for information in the time-evolution of the hidden state, which will be prove to be beneficial, as we will see shortly. On the other hand, the use of the last state only, essentially implies an expectation that all "useful" information present in the sequence will be encoded into the last hidden state.

3.3. Trading decisions: strategy and capital allocation

We go on to describe how asset price predictions are to be used to generate trading decisions. As we have mentioned in Section 2.1, it is common to buy or sell based on a prediction's directional accuracy, even when the underlying model was trained to attain low mean-squared or absolute errors. Here, we depart from prior approaches by taking advantage of the fact that it is possible to be nearly agnostic about an event occurring, e,g., the direction (up/down) of an asset's price movement, and yet to be able to glean significant information about the profitability of a trade when conditioning on particular events. In our case the events will have to do with the relative location of predictions within their own distribution. To make matters precise, we begin by defining the notions of *allocation policy* and *trading strategy*:

Definition 1 (Allocation policy). *Let $\mathcal{M}$ be a prediction model that produces at each time t an estimate, $\hat{y}_{t+1}$, of an asset's future value y_{t+1}, the latter resulting from some underlying stochastic process. Let $\hat{r}_t = \hat{y}_{t+1}/y_t - 1$ be the model's one-step* predicted *returns at time t, and $D_{\hat{r}}$ their distribution. An* n**-bin allocation policy** *is a pair (Q, A), where $Q \in \mathbb{R}^{n-1}$ is a vector of percentiles of $D_{\hat{r}}$ in increasing order, and $A \in \{\mathbb{R} \cup \emptyset\}^n$ a vector of capital* allocations *in units of the asset (e.g., number of shares to purchase), where the element $\emptyset$ denotes sale of all units of the asset held.*

Definition 2 (Trading strategy). *Given a prediction model, $\mathcal{M}$, the current asset price, y_t, and an n-bin allocation policy (Q, A), our proposed trading strategy consists of the following steps, executed at each time t:*

- *Query $\mathcal{M}$ for the asset's predicted return $\hat{r}_t$ in the next time step, $t + 1$.*
- *Let*

$$i = \begin{cases} 1 & \text{if} \quad \hat{r}_t < Q_1 \\ j+1 & \text{if} \quad \hat{r}_t \in [Q_j, Q_{j+1}), \\ n & \text{if} \quad \hat{r}_t \geq Q_{n-1}. \end{cases}$$

- *If $A_i > 0$ and we are currently not holding any of the asset, buy A_i units of the asset. If $A_i > 0$ and we are already holding some of the asset, or if $A_i = 0$, do nothing.*
- *If $A_i = \emptyset$, sell any and all units of the asset held.*

In our case, $\mathcal{M}$ will be any of the models described in the previous Section. Intuitively, the percentiles in Q partition the real line into n bins, and the trading strategy is to buy A_i units of the asset whenever the predicted return lies in bin i and we do not already own the asset. For simplicity, we will consider allocation policies in which $Q_1 = 0$ and $A_1 = \emptyset, A_{i>1} \geq 0$, i.e, we always sell all of our asset if the predicted return for the next trading day is negative, that being the only event on which we sell. Of course, one can envision variations of the above strategy, including negative allocation values corresponding to short sales (i.e., selling an amount of the asset under the obligation to buy it back at a later time, with the expectation that its value will have declined), assigning the sell "signal", $\emptyset$, to more than one bins as determined by Q, using time-varying values of A_i depending on the available cash in our portfolio, or measuring the allocations A_i in monetary amounts as opposed to units of the asset. Some of these options are worth exploring in their own right, but we will limit ourselves to the setting previously described because of space considerations.

Remark 1 (Relationship with up-down strategy). We note that the proposed trading policy can be viewed as a generalization of the classical directional "up-down" trading strategy. If we set $Q = 0$, i.e., use only one separating point, and $A = [\emptyset, 1]^T$, then there will be only two bins, one for positive predicted returns (buy 1 unit of the asset), and one for negative predicted returns (sell), which is exactly how the directional strategy operates.

3.3.1. Optimizing the allocation policy

Armed with the above definitions, we can now consider the question of what allocation policy is optimal for a given prediction scheme while following the trading strategy of Def. 2. We will be buying A_i units of the asset when the predicted return lies in bin $i > 1$, and will then hold those units until the predicted return falls within bin 1. Over time, this process generates a set of *holding intervals*, $I_{ij} = [\tau_{ij} + 1, \omega_{ij}]$, where τ_{ij} denotes the time of the j-th buy transaction while the predicted return was in bin $i > 1$, and ω_{ij} the time of the following sell transaction (i.e., the first time following τ_{ij} that the predicted return falls within bin 1). It should be clear from Def. 2 that the intervals I_{ij} will be non-overlapping (because we never buy when already holding the asset) and that their union will be the entire set of times during which we hold a nonzero amount of the asset.

Our allocation policy risks a monetary amount of $A_i \cdot y_{\tau_{ij}}$ to purchase A_i units of the asset at each time τ_{ij}. The net profit generated from the at-risk amount at the end of the holding interval I_{ij} (i.e., after a single buy-sell cycle) is therefore

$$A_i(y_{\omega_{ij}} - y_{\tau_{ij}}). \tag{1}$$

Over time, there will be a number of instances, N_i, that the asset was purchased while at bin i and the total net profit after $M = \sum N_i$ buy-sell transactions will then be

$$G = \sum_{i=2}^{n} A_i \sum_{j=1}^{N_i} (y_{\omega_{ij}} - y_{\tau_{ij}}), \tag{2}$$

resulting in an expected net profit of

$$\mathcal{E}\{G\} = \sum_{i=2}^{n} A_i \mathcal{E}\left\{ \sum_{j=1}^{N_i} (y_{\omega_{ij}} - y_{\tau_{ij}}) \right\}. \tag{3}$$

The expectation terms in Eq. 3 represent the expected sum of differences in asset price over each bin's holding intervals, that is, the expected total rise (or fall) in asset price conditioned on buying when the expected return falls within each particular bin and following the proposed trading strategy. We note that Eqs. (1)-(3) ignore transaction costs. This is done in order to facilitate comparison with other works which operate on the same basis and because - as we explain in the next Section - transaction costs will have a negligible effect in our case, or can be eliminated altogether.

It is clear from Eq. 3 that the expected total profit will be maximized with respect to the allocations A_i if each A_i is set to zero whenever the associated expectation term is negative, and to as large a value as possible when the expected price difference is positive. Assuming a practical upper limit, A_{max}, to the number of asset units we are able or willing to buy each time, the optimal allocation values are thus

$$A_{i>1} = \begin{cases} A_{max} & \text{if} \quad \mathcal{E}\left\{ \sum_{j=1}^{N_i} (y_{\omega_{ij}} - y_{\tau_{ij}}) \right\} > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

We will take A_{max} to be constant (across time and across the bins i) for the sake of simplicity, although other choices are possible as we have previously mentioned.

4. Model training and allocation policy selection

As we have seen, the specification of the optimal allocation and trading strategy described in the previous Section requires the (empirical) distribution of the predicted returns which are generated by our prediction model. That is, we need a trained model in order to obtain concrete values for the percentiles in Q and the bins they induce, as well as the estimated sums of price differences per bin that will determine the allocations, A, in Eq. 4. A step-by-step summary of our approach is as follows:

- Select a prediction model architecture. Train and test the selected model over a rolling window of size T, where for each trading day, t, in 1/1/2005-1/1/2008, the model parameters (e.g., network weights) are first adjusted using price data from the immediate past, (i.e., input $[x_{t-T}, \ldots, x_{t-1}]$, and output target $[y_{t-T+1}, \ldots, y_t]$); then, the model is given $[x_{t-T+1}, \ldots, x_t]$ as input and asked to predict the price sequence once step ahead, $[\hat{y}_{t-T+2}, \ldots, \hat{y}_{t+1}]$, from which we keep the next day's adjusted closing price, $\hat{y}_{t+1}$, and calculate the predicted return, $\hat{r}_t$.

- Use the distribution $D_{\hat{r}}$ of the predicted returns generated by the rolling-window process to determine the vector of cutoff points, Q. These points could be chosen to have constant values, or to correspond to specific percentiles of $D_{\hat{r}}$, or in any other consistent manner.

- For each bin determined by Q, use the history of input data and corresponding price predictions over 1/1/2005-1/1/2008 to determine the buy and sell events that would have resulted by having followed the trading strategy of Def. 2. Compute the asset price differences for each buy-sell cycle and sum over each bin, i, in which the corresponding buy transactions took place (Eq. 3). Set the allocations $A_{i>1}$ to their optimal values as per Eq. 4.

- Having set the allocation policy, (Q, A), test the profitability of the trading strategy for a range of model hyper-parameters, over a subsequent time period, 1/2/2008-12/31/2009, again training the model daily based on input data from the previous T trading days before predicting the next day's price. Select the hyper-parameter values which yield the most profitable – as opposed to most accurate – model variant.

- Apply the selected model variant to a new data sample, 1/4/2010-12/20/2019, and evaluate its profitability, G/C, where G is the net profit, in (2) and C is the initial capital invested. We will use the term *out-of-sample* (instead of *testing*) when referring to this period, in order to avoid confusion with the rolling training-testing procedure used to make predictions. Similarly, the period 1/1/2005-12/31/2009 used to construct the initial allocation policy and perform hyper-parameter selection will be referred to as *in-sample*.

We go on to discuss important details and practical considerations for each of these steps.

In contrast to other studies, e.g., [6], which make use of a more traditional training - validation - testing framework, we found that a rolling training-testing window approach was better suited to our setting, where at each time, t, the model is trained using data from the immediate past and then asked to predict the price at time $t + 1$. By updating the model's parameters or weights at each step using the most recent information to make a prediction, we avoid "lag" or "time disconnection" between the training and testing data (as would be the case, for example, if our training data was far into the past compared to the time in which we are asked to make a prediction) and allow the model to better "react" to changing regimes in the price sequence as time progresses, leading to it being more profitable. As we noted in Section 3, we used a sequence-to-sequence training schema for all of our models because doing so led to significantly higher cumulative returns compared to sequence-to-value training. We speculate it presents more information for trees to act upon and hence perform the splits and that asking the LSTM network to predict *sequences* of the asset price over a time interval, combined with the use of all hidden vectors produced in that interval, allows the network to learn possible latent micro-structures within the price sequence (e.g., trend or other short-term characteristics), thus facilitating the training process.

4.1. LSTM training

Our two LSTM models were trained using back-propagation through time (BPTT), with the ADAM optimizer, implemented in Python 3 and Tensorflow v1.8. Network weights were initialized from a uniform distribution (using the Glorot-Xavier uniform method) and LSTM cell states were initialized to zero values. We applied exponential decay to our learning rate, and trained for 1600 iterations. The number of iterations was chosen to avoid over-fitting by observing both the training and testing errors, using a mean-squared loss function. It is important to note that the rolling window training process, as described above, used a batch size of 1 (i.e., the network was trained each time on a *single* T-sized window of the data, x_t). Our numerical experiments showed that for our proposed architecture, larger batch sizes did not increase the profitability of the overall model, and sometimes even led to lower performance[6]. Depending on the choice hyperparameters (listed in Section 4.4), the time required to

[6]For example, training with a batch size of 50 on the S&P500 index resulted in twice the mean average percent error and a 15% lower cumulative return compared to training with a batch size of 1, with full details on trading

compute one rolling window step was between 7 and 30 seconds on an modestly equipped computer (Intel i7 CPU and 16 GB of RAM) which added up to 14 hrs, on average, for a complete back-test over our decade-long testing period.

4.2. Random Forests and Gradient Boosting

The RF and gradient boosted tree models were implemented using the Python-based scikit-learn and XGBoost (XGB) packages, respectively. XGB offers a parallel tree boosting algorithm leading to very fast and scalable execution [32]. As a result, the execution time for a complete back-test was on average between 5-10 minutes, on the aforementioned hardware specification. RFs, on the other hand, required longer to execute, averaging 30 minutes for a complete back-test.

4.3. Allocation policy selection

The overall number and values of the allocation policy's cutoff points (Q) were determined after numerical experimentation to ensure that all of the bins defined by Q included an adequate population of samples and were sufficient in number to allow us to discern differences in the profits realized by the trading strategy when executing a buy in different bins. We found that an effective approach, in terms of ultimate profitability, was to select the cutoff points in Q using percentiles of the distribution of the absolute expected returns, $D_{|\hat{r}|}$. Thus, Q_1 was set to zero (so that as per Def. 2 we sell any holdings if our model predicted return becomes negative), and another six cutoff points, $Q_2, \ldots, Q_7$, were chosen to correspond to the first six deciles (10%, 20%, 30%, 40%, 50%, and 60% points) of the distribution of absolute predicted returns, resulting in eight bins. For each bin, i, we could then use historical predicted and actual prices to calculate the sum of price differences (as per in Eq. 4) and determine the optimal allocation A_i. We emphasize that the above choices with respect to Q were the result of experimentation, and that the elements in Q could in principle be optimized; however, doing so is not trivial, and we will not pursue it here.

In practice, it is advantageous to allow the allocation policy (Q, A) to adapt to new data rather than be fixed for a significant amount of time, because the distributions of the predicted and realized returns are unlikely to be stationary. As the model makes a new prediction each day, we have a

performance to be given in Section 5.

chance to add this new data to the distribution of predicted returns and adjust the cutoff points in Q. In the same vein, as the trading strategy generates new buy-sell decisions, we may re-calculate the allocations A. There are several options here, such as to accumulate the new information on returns and trading decisions using in a growing time window that stretches to the earliest available data, or to again use a rolling window so that as each day new data is added to the distribution, "old" data is removed. After experimenting with different alternatives, we chose to update the percentiles in Q at each time step, using a growing window of data that began 120 time steps prior to our first out-of-sample prediction. For example, in order to making a prediction for 1/4/2010 - the first day of our out-of-sample period - the model's predicted returns between 7/15/2009 - 1/1/2010 were included in the empirical distribution used to calculate Q. This was done in order to avoid the problem of initially having no data (e.g., on the first day of our out-of-sample period) from which to calculate the Q_i.

With respect to the allocations, A_i, when computing the per-bin sums of price differences in Eq. 4, we found that it was best in practice (i.e., increased profitability) to include all of the available history of buy-sell events and corresponding price differences, up to the beginning of our data sample (1/1/2005). Moving forward, as buy and sell decisions were made at each time step, the resulting price differences were included in the sums of Eq. 4 and the allocations A were updated, if necessary. An instance of the set of bins, corresponding cutoff points and sums of price differences for each bin (computed for use on the first trading day of the out-of-sample period, 1/04/2010) are listed in Table 2. For example, if the predicted return for 1/4/2010 for the S&P500 fell within the 30-40% bin (equivalently, the interval $[0.65\%, 0.97\%)$), then we would not buy any of the asset (corresponding allocation set to zero) because if we did, the expected price difference in the asset would be negative at the time we sell the asset.

As previously discussed, it is optimal to buy $A_i = A_{max}$ units only when the predicted return falls within a bin whose realized sum of price differences is positive. In practice, the choice of A_{max} is usually subject to budget constraints or investment mandates, and could also be tied to the investor's risk profile. In our study, A_{max} was set to the maximum number of units one could buy with their available capital at the start of the trading period. For example, at the beginning of our out-of-sample period, 1/04/2010, the price of the S&P 500 index was $\$1,132.989$, and therefore if our initial capital were $C = \$28,365$ we would set $A_{max} = 25$. On a practical note, because stock indices are not directly tradeable, one would instead trade one of the available ETFs which track the index and are considered

	S&P500		DJIA		NASDAQ		R2000	
Bin i (deciles)	Q_i	δ_i	Q_i	δ_i	Q_i	δ_i	Q_i	δ_i
1: SELL	0	-	0	-	0	-	0	-
2: 0%-10%	0.12%	126.97	0.01%	-1785.45	0.05%	62.37	0.01%	139.19
3: 10%-20%	0.38%	99.92	0.02%	-1529.37	0.10%	520.54	0.03%	90.76
4: 20%-30%	0.65%	131.35	0.03%	904.60	0.12%	52.27	0.06%	23.19
5: 30%-40%	0.97%	-66.71	0.04%	-88.28	0.16%	-59.84	0.07%	87.32
6: 40%-50%	1.18%	128.67	0.06%	1062.01	0.18%	206.65	0.10%	1.64
7: 50%-60%	1.44%	-191.68	0.08%	381.46	0.24%	254.78	0.12%	6.17
8: 60%-100%	-	222.85	-	3850.18	-	719.68	-	248.96

Table 2: Snapshot of the cut-off points, Q_i (reported as percentages), for each bin-interval of the predicted return distribution. The Q_i mark the upper values of each bin, and are computed using our model's predicted returns from 7/15/2009 -1/04/2010. The δ_i denote empirical estimates of the corresponding sums of price differences from Eq. 4 when using the proposed trading strategy, over the period 1/1/2005-1/04/2010.

one of the most accessible and cost-effective options for investing in indices. Therefore, in order to apply our approach to the S&P500, we could choose to trade SPY, a popular ETF which tracks that index, and – based on the aforementioned initial capital and SPY price of \$94.55 on 1/04/2010 – set $A_{max} = 300$. In the experiments detailed in the next Section, we will choose to trade four well-known ETFs: SPY for the S&P500, DIA for the DJIA, IWM for the R2000 and ONEQ for the NASDAQ.

4.4. Hyper Parameter Tuning

Because our ultimate goal is profitability, we evaluated the cumulative returns of our model(s) under several combinations of their structural parameters, in order to identify the most profitable variant (which may generally depend on the asset under consideration). For each model and each choice of hyper-parameters, we carried out the rolling training-testing procedure described above, over the period 1/1/2005-1/1/2008, and calculated an initial allocation strategy (Q, A). Following that, we used data from 1/2/2008-12/31/2009, to calculate the profit realized by applying our trading strategy while adjusting the allocation policy after each trading day. This subsequent 505 trading day period, contained a sufficient number of rolling windows and corresponding predictions from which to compare performance for each choice of hyper-parameters. In addition, it encompasses the global financial crisis of 2009 and was long enough to include both the decline caused by the global financial crisis and the subsequent bull market. This allows us to evaluate our strategy both on down-trending and up-trending market conditions.

4.4.1. LSTM networks

A grid-search approach was used to select the parameters of the neural networks, namely the number of LSTM layers, number of units per LSTM layer, H, the dropout parameter (dropout was applied only on the input of a given neuron), and the length T of the input sequence.

Parameters	Range
Nr. of LSTM layers	$2,3$
Nr. of units (H)	$32,64,128$
Input sequence length (T)	$11,22,44$
Dropout	0,50,70%

Table 3: LSTM: Hyper-parameter grid-search values.

The network parameters leading to the highest cumulative return for each of the four stock indices of interest are listed in Table 4. On a modest 6-machine, 96-CPU, 96Gb RAM computing cluster, and

Model	Parameters	S&P 500	DJIA	NASDAQ	R2000
LSTM-AH	Nr. of LSTM layers	3	3	3	3
	Nr. of units (H)	64	64	32	32
	Input sequence length (T)	22	22	22	11
	Dropout	50%	70%	50%	50%
LSTM-LH	Nr. of LSTM layers	3	3	2	2
	Nr. of units (H)	128	32	32	64
	Input sequence length (T)	11	11	11	11
	Dropout	50%	70%	0%	70%

Table 4: LSTM Grid-search: optimal hyper-parameter values for the two variations of LSTM. *Top*: LSTM using all hidden states at the output layer, *bottom*: LSTM using only the last hidden state.

for one stock index/asset time series, the entire grid search process took less than 5 hours to complete for smaller network configurations (length-11 input sequences, 2 layers with 32 neurons per layer), while the largest configurations required up to 3 days.

4.4.2. Random Forests and Gradient Boosting

For the tree-based models we again looked for the hyperparameter values which maximized profitability over the in-sample period 1/2/2008-12/31/2009. In this case, however, the parameter set was larger while at the same time evaluating the profitability of the model under a particular hyperparameter combination was significantly faster compared to the LSTM case. For these reasons, we opted for a random search over the parameter space instead of using a fixed grid. The model hyperparameters and the ranges in which we were interested are listed in Table 5. We evaluated a total of 500 parameter combinations for RFs and 500 for gradient boosted trees (to which we will refer using the acronym XGB). Tables 6 and 7 list the final hyperparameter values which maximized profitability during the validation period for the RF and XGB models, respectively. On the aforementioned 6-machine, 96-CPU cluster, the hyperparameter search process took an average of 4 and 3 hours to complete for the RF and XGB models, respectively, for each stock index.

We note that our hyperparameter search was not exhaustive and was guided by numerical experimentation and computational time considerations. Our goal was not to identify the best model but rather to demonstrate the merits of selecting parameters for profitability while trading based on the distribution of predictions, leading to superior returns. Of course, in a live/production setting one may perform a finer-grained parameter search, and the results presented in the next Section could potentially improve even further.

RF	Range	XGB	Range
Nr. of Estimators	10 – 1000	Nr. of Estimators	10 – 1000
Min Samples split	2 – 10	Sub-sample	0.1 – 1
Min Samples Leaf	2 – 10	Learning rate	0.01 – 1
Max Leaf Nodes	2 – 5	Gamma	0.1 – 0.3
Max Depth	1 – 50	Max Depth	1 – 12
		Col Sample Bytree	0.5 – 1
		Reg Lambda	0.1 – 0.3

Table 5: Hyper-parameters and their ranges. Left: Random Forests (RF); Right: Gradient boosted trees (XGB)

Parameters	S&P 500	DJIA	NASDAQ	R2000
Nr. of Estimators	130	330	353	374
Min Samples split	3	4	3	7
Min Samples Leaf	2	4	2	2
Max Leaf Nodes	4	2	4	2
Max Depth	32	1	26	24

Table 6: Random Forest: profit-maximizing hyperparameters resulting from a random search.

Parameters	S&P 500	DJIA	NASDAQ	R2000
Nr. of Estimators	817	41	73	62
Learning rate	0.47	0.35	0.95	0.58
Sub-sample	0.4	0.3	0.5	0.2
Gamma	0.2	0.2	0.2	0.1
Max Depth	7	5	11	6
Col Sample Bytree	0.5	0.9	0.8	1
Reg Lambda	0.1	0.1	0.2	0.1

Table 7: XGB: profit-maximizing hyperparameters resulting from a random search.

5. Results

Having settled on a choice of hyper-parameters for each model, we tested their performance on our out-of-sample period, 1/4/2010 - 12/20/2019. For each trading day, t, within this period, we updated the model's parameters by training it on the time window of input data from $t-T, ..., t-1$ with a target sequence output of $y_{t-T+1} \dots y_t$, and then computed the prediction, $\hat{y}_{t-T+2} \dots \hat{y}_{t+1}$, of the next day's closing price, using input data over the window $t-T+1 \dots t$. Next, we determined the predicted return, $\hat{r}_t = \frac{\hat{y}_{t+1}}{y_t} - 1$, and the bin, i, (based on the percentiles in Q) within which it fell. We then bought or sold the amount of asset units (corresponding tracking ETF for each index) prescribed by the allocation policy. If a buy was executed, we recorded the index of the bin, i, and the asset price at that time. When that asset was later sold, the profit obtained from that transaction was added to the sum of price differences term in Eq. 4, and we recalculated the corresponding allocation A_i if the update caused a change in the sign of the sum. Finally, we included $\hat{r}_t$ in the growing set of past predicted returns and adjusted Q to account for the new distribution, thus completing our update of the allocation policy, (Q, A).

We proceed with the main results concerning i) the prediction accuracy, and ii) the profitability of our models over the out-of-sample period, for each of the stock indices studied.

5.1. Error metrics and prediction performance

Although our central goal is to achieve good trading performance, we will begin by evaluating our models using some of the metrics often cited in studies involving prediction accuracy, including mean directional accuracy[7] (MDA) which measures the accuracy of predicting the correct direction (up/down) of price movement, mean squared error[8] (MSE), mean absolute error[9] (MAE), mean absolute percentage error[10] (MAPE). Table 8 shows the predictive performance of each model for each of the four stock indices, over the same out-of-sample period (1/4/2010-12/20/2019).

Model	Metric	S&P 500	DJIA	R2000	NASDAQ
RF	MDA	49.78%	49.62%	50.26%	49.06%
	MAPE	0.85%	1.01%	1.33%	1.07%
	MAE	15.86	172.95	14.25	48.69
	MSE	491.19	56425.60	346.44	4578.09
XGB	MDA	50.22%	49.82%	50.46%	50.02%
	MAPE	0.87%	0.83%	1.18%	1.00%
	MAE	16.14	140.48	12.57	45.73
	MSE	548.61	42264.60	281.31	4726.43
LSTM-LH	MDA	49.78%	48.74%	51.50%	51.38%
	MAPE	0.65%	0.62%	0.91%	0.87%
	MAE	12.04	106.07	9.72	42.82
	MSE	303.08	24117.30	166.324	6749.22
LSTM-AH	MDA	52.08%	50.90%	49.86%	51.81%
	MAPE	0.65%	0.63%	0.90%	0.79%
	MAE	12.11	107.23	9.58	35.86
	MSE	313.98	24781.90	163.92	2784.77

Table 8: Error metrics for the prediction models trained, over the testing period 1/4/2010-12/20/2019. RF: Random forest, XGB: Gradient boosted trees, LSTM-LH: LSTM network using only the last hidden state at the output layer, LSTM-AH: LSTM network using all hidden states.

Overall, we observe that the LSTM variants performed better than the tree-based models with respect to MAPE, MAE and MSE, while models within each family have similar scores. The LSTM-AH network showed a slight advantage over LSTM-LH in terms of MDA. Moreover, the neural networks achieved a MAPE below 1% and a MDA close to 50%, for all indices.

[7]MDA $= \frac{1}{N}\sum_{t=1}^{N} \mathbb{1}(sign(\hat{y}_t - y_{t-1}) \cdot sign(y_t - y_{t-1}))$, where the indicator function $\mathbb{1}(x)$ returns 1 if x is positive, zero otherwise, y_t and $\hat{y}_t$ are the realized and predicted prices for time t, respectively, N is the length of the out-the-sample sequence.

[8]MSE $= \frac{1}{N}\sum_{t=1}^{N}(y_t - \hat{y}_t)^2$.

[9]MAE $= \frac{1}{N}\sum_{t=1}^{N}|y_t - \hat{y}_t|$.

[10]MAPE $= \frac{1}{N}\sum_{t=1}^{N}\frac{|y_t - \hat{y}_t|}{y_t}$.

A series of pairwise Diebold-Mariano (DM) tests showed that, at a 5% confidence level, both LSTM networks were better at forecasting all four stock indices when compared to either the XGB or the RF models, with the corresponding DM-statistics varying between 9.61 and 14.80 and p-values of less than 10^{-4} in all 16 test performed (2 LSTM networks, 2 tree-based models, 4 stock indices). At the same 5% confidence level, there was no significant difference in forecasting accuracy between LSTM-AH and LSTM-LH, or between XGB and RF, for any stock index/ETF. With respect to directional accuracy (MDA), while some models appeared to perform slightly better than others, applying the Pesaran-Timmermann (PT) test to the predicted vs. actual direction of price movements on all four models and all four stock indices did not confirm that any model could predict price movement direction in a statistically significant manner at a 5% confidence level. However, the fact that the models studied do not provide a significant advantage in predicting next-day price movement direction only implies that they may be ineffective when paired with the directional up-down strategy often used in the literature. It does *not*, preclude any of them from being able to produce significant profits under *other* trading strategies, such as the one proposed in Section 3.3 which does not specifically require day-to-day directional accuracy. Finally, we have not included R^2 values in Table 8 because index price series tend to exhibit high serial correlation; thus it is common for various prediction schemes to achieve R^2 values which are close to 1 and not very informative on their own (our own network's predictions for the four indices studied here, gave R^2 values between 0.990 and 0.995)

When compared against other recent studies, e.g., [6], our LSTM-AH model has significantly lower MAPE for the S&P500 and DJIA (see Table 9 - using the same test period as in [6]). We opted to compare our LSTM-AH variant when it comes to error metrics, because, as we will see in the next Section, it had a better overall profitability compared to the other models we tested. With respect to

	S&P500		DJIA	
Metrics	[6]	LSTM	[6]	LSTM
MAPE	1.1%	0.7%	1.1%	0.6%

Table 9: Comparison of our LSTM-AH network with [6] (S&P500 and DJIA) for the out-of-sample period used therein (10/1/2010 - 9/30/2016).

[8] and the metrics reported therein (see Table 10), and for the same out-of-sample period used in that work, our LSTM-AH network performed slightly better for the S&P500 in terms of MAPE and MAE,

	S&P500		NASDAQ	
Metrics	[8]	LSTM	[8]	LSTM
MAPE	1.05%	1.04%	1.08%	1.17%
MAE	13.03	12.85	52.4773	30.8463
RMSE	17.6591	17.73	70.4576	41.1703

Table 10: Comparison of our LSTM-AH network with [8] (S&P500 and NASDAQ), for the out-of-sample period used therein (the year 2011).

and better for the NASDAQ with respect to MAE and RMSE. Finally, Table 11 compares with [7], where our approach outperformed in all three statistics (MAPE, MAE, MSE).

	S&P500	
Metrics	[7]	LSTM
MAPE	1.0759%	0.8%
MAE	12.058	11.66
MSE	342.48	278.90

Table 11: Comparison of our LSTM-AH network with [7] (for the S&P500), for the out-of-sample period used therein 01.02.2008 - 7.26.2017.

We note that the LSTM-AH network as well as its LH variant, rely on a relatively simple architecture with no pre-processing layers, making it straightforward to implement and train. This is to be contrasted with more elaborate designs, such as the three-layered approach of [6] passing data through a wavelet transformation and autoencoder neural network before reaching the LSTM network, or the dual LSTM architecture of [7]. Despite its structural simplicity, our scheme's predictive performance is similar to that of more complex approaches, and compares favorably to studies with long out-of-sample periods ([6], [7]).

Remark 2 (A note on prediction accuracy). We emphasize the fact that the comparisons and discussion of predictive performance - although encouraging - are given here mainly for the sake of providing a fuller picture, and that good predictive performance means little unless a model also performs favorably in terms of profitability. In fact, there is an important point of caution to keep in mind regarding predictive accuracy, and it has to do with the proper context when citing MAPE, MAE and MSE values. Specifically, daily asset price changes are usually small on a relative basis, and thus a MAPE of less than 1%, for example, is neither unusual nor surprising. In fact, a naive approach which predicts that tomorrow's price will simply be equal to today's (i.e., $\hat{y}_{t+1} = y_t$), applied to the S&P500, would typically yield very low error metrics, e.g., MAPE = 0.64%, MAE=20.0203, MSE=844.186 over our out-of-sample period, or a MAPE of 0.83% for the period examined in [7]. Of course, this naive approach is completely useless from the point of view of trading because it does not allow us to make any decision on whether to buy or sell, highlighting the fact that favorable MAPE, MAE, and MSE scores do not by themselves guarantee success in trading. When the ultimate goal is profitability, predictions should be evaluated in the context of a specific trading strategy, and one may achieve high(er) returns by looking beyond average errors, as the trading and allocation strategies of Section 3.3 do.

5.2. Profitability - Trading simulation

In Table 12 we present the return and risk statistics for each model and each stock index along with the results from the BnH strategy for comparison. Trading performance was measured by computing the cumulative returns obtained over the entire out-of-sample period ($CR = \prod_2^N s_i/s_{i-1} - 1$, for a portfolio or asset whose value is s_i over trading days $i = 1, 2, \ldots, N$) and the corresponding annualized returns ($AR = (1 + CR)^{252/N} - 1$, where 252 is the nominal number of trading days per calendar year). We also report metrics related to trading risk which are of interest when assessing any investment, including annualized volatility (AV, the standard deviation of returns over a the out-of-sample period, multiplied by the square root of the number of trading days in a year), Sharpe ratio (SR, the difference between AR and the federal funds rate, divided by AV), and draw-down (DD, the largest peak-to-trough percent decline during an investment's lifetime).

As is evident from Table 12, the best models exhibit very strong results, beating BnH as well as recent studies (detailed comparisons will follow shortly). When considering the metrics CR, AR, AV, and DD, the overall best model across all indices was LSTH-AH in the sense that it always ranked first or second. Considering each index separately and judging based on CR as well as return-to-risk ratio (SR) the best models were XGB, LSTM-AH, LSTM-AH, and LSTM-LH for the S&P 500, DJIA, NASDAQ, and R2000, respectively. The use of all hidden states in the output layer of the LSTM network (LSTM-AH variant) proved to be beneficial, as that model achieved the lowest AV for the S&P500, DJIA and NASDAQ, amongst all four models, and the smallest DD for the S&P500, DJIA, and R2000. A comparison of LSTM-AH vs. LSTM-LH shows that the former yields lower AV and DD against the latter, in 3 out of 4 indices. Hence, apart from the strong performance of the model, the use of all hidden states adds "defensive" characteristics, for a more balanced model.

Index	Model	CR	AR	AV	SR	DD
SP500	XGB	428.83	18.23	26.39	0.69	−21.20
	RF	315.95	15.41	23.59	0.65	−22.90
	LSTM-LH	347.16	16.25	24.01	0.68	−28.53
	LSTM-AH	350.32	16.34	18.67	0.88	−16.94
	BnH	188.87	11.25	14.79	0.76	−11.18
DJIA	XGB	284.52	14.48	25.94	0.56	−17.97
	RF	385.71	17.20	25.99	0.66	−18.06
	LSTM-LH	216.51	12.27	31.78	0.39	−21.70
	LSTM-AH	403.21	17.64	18.82	0.94	−11.19
	BnH	172.87	10.61	14.08	0.75	−10.71
R2000	XGB	203.35	11.79	41.93	0.28	−30.21
	RF	307.69	15.16	33.05	0.46	−37.88
	LSTM-LH	374.16	17.01	23.87	0.71	−25.56
	LSTM-AH	333.76	15.88	32.08	0.50	−24.83
	BnH	167.34	10.38	19.60	0.53	−15.76
NASDAQ	XGB	353.12	16.40	39.28	0.42	−30.86
	RF	331.34	15.83	27.93	0.57	−28.20
	LSTM-LH	461.56	19.04	25.34	0.75	−21.31
	LSTM-AH	497.17	19.68	23.06	0.85	−21.47
	BnH	293.32	14.75	17.08	0.86	−12.45

Table 12: Returns and risk statistics for the out-of-sample period (1/04/2010-12/20/2019). *BnH* represents the buy-and-hold strategy (the asset was bought once at the beginning of the trading period and sold at the end), RF: Random forest, XGB: Gradient boosted trees, LSTM-LH: LSTM network using only the last hidden state at the output layer, LSTM-AH: LSTM network using all hidden states. *CR* denotes cumulative return, *AR* is annualized return, *AV* is annualized volatility, *SR* is the Sharpe ratio, *DD* is draw-down (largest peak-to-trough percent decline during an investment's lifetime).

The dominance of the LSTM models over the tree-based ones is evident when we compute the Information Ratio[11], a measure used to compare two trading strategies. While the SR considers the return vs. risk of each strategy in isolation, the IR captures the relative return and relative risk of one strategy against the other. Hence, we are able to compare strategies by controlling for differences in returns and risk. In Table 13 we observe that LSTM-AH and LSTM-LH are both better than RF and XGB, with the exception of the S&P 500 index, where XGB is better than all others. LSTM-AH compares favorably against LSTM-LH on the S&P 500, DJIA and NASDAQ while LSTM-LH is better with the R2000.

[11]Information Ratio (IR) is the ratio of the annualized return difference over the annualized volatility difference between two trading strategies A, B: $IR(A, B) = \dfrac{AR_A - AR_B}{AV_A - AV_B}$

		XGB	RF	AH	LH
S&P 500	**XGB**	-	**0.16**	**0.09**	**0.10**
	RF	-0.16	-	-0.05	-0.06
	AH	-0.09	0.05	-	0.00
	LH	-0.10	0.06	0.00	-
DJIA	XGB	-	-0.14	-0.13	0.10
	RF	0.14	-	-0.02	0.31
	AH	**0.13**	**0.02**	-	**0.19**
	LH	-0.10	-0.31	-0.19 -	
NASDAQ	XGB	-	0.02	-0.11	-0.08
	RF	-0.02	-	-0.17	-0.13
	AH	**0.11**	**0.17**	-	**0.03**
	LH	0.08	0.13	-0.03	-
R2000	XGB	-	-0.12	-0.12	-0.16
	RF	0.12	-	-0.03	-0.10
	AH	0.12	0.03	-	-0.04
	LH	**0.16**	**0.10**	**0.04**	-

Table 13: Information Ratios (IRs) taking into account the active return (difference of annualized returns) and active risk (difference of annualized standard deviation of returns) between each pair of models. High IR indicates that the model corresponding to the row is better in delivering higher returns with lower risk than the model associated with the column, i.e., for the SP500 the value between (XGB, RF) =0.16 implies that XGB is better than RF. For each stock index the best model (shown in bold) is the one that has only positive IR values against all others.

Remark 3 (A note on transaction costs). The previous discussion and the returns cited above have assumed no transaction costs. This is not done solely in order to facilitate comparisons with other works which also ignore transaction costs; it is a realistic assumption because our strategy trades only one asset, an ETF, does not take short positions where costs have a larger impact, and trades sparingly as noted above. Specifically, retail investors can trade ETFs (such as the index-tracking ETFs used in our study) with *zero* transaction costs through several online brokers (e.g., RobinHood[12], Vanguard[13], and others). For institutional investors, transaction costs are minimal, ranging from 2 to 5 basis points per trade regardless of transaction amount, which would make a negligible difference in the returns achieved. For example, even with a 50 basis points per-trade transaction cost (approximately 10 times higher than the typical cost for institutional investors), the cumulative return achieved for the S&P500 (ETF) using our LSTM-AH model would be 312.9% instead of 350.32%, a reduction of only 0.418%

[12]https://robinhood.com

[13]https://investor.vanguard.com/investing/transaction-fees-commissions/etfs

on an annualized basis; for the DJIA, R2000 and NASDAQ, the differences would be even smaller.

5.3. Comparisons with recent works

The LSTM variants, including the modified AH version, generated much stronger returns than those of BnH, at similar or lower risk as measured by the SR. RFs and XGB trees had consistently lower SRs than the BnH strategy. Comparisons with recent works that also attempt to beat some of the major stock indices used here, show that our trading strategy is competitive in terms of profitability, albeit at a relatively higher risk sometimes. Although the approaches cited here differ in methodology, underlying asset(s) traded and testing periods, they all focus on beating a benchmark, most notably the S&P500, hence performance comparisons can be made on that basis. We emphasize the fact that each of the following comparisons was made on the same data set used in the corresponding paper we compared against. That is, for each work, we re-trained/tested our own model(s) on the data used by the authors and then compared performances. These pair-wise comparisons (our approach vs. each of the others) are thus fair in the sense that the two alternative methodologies assessed on the same data set each time.

We begin with those works having the longest (approximately 9-year) test periods and progress to those with relatively short ones. In each case, the numbers cited are those of our best-performing model for the stock index on which the comparison is made, based on Table 12. A direct comparison to [7] (also trading the S&P500 over a long out-of-sample period, 1/2/2008-7/26/2017), shows that our approach outperforms by a significant margin in terms of cumulative return (206.12% vs 96.55%) over the same time period. Next to [12], which employed deep convolutional neural networks instead of LSTMs, our model achieved an annualized return of 14.63% compared to the 10.77% reported in that work when trading the same tracking EFT as we have (SPY), or 13.01% when using a portfolio of ETFs, over the period of 1/1/2007-12/31/2016.

Works with somewhat shorter (2-3 year) out-of-sample periods are more numerous, and include the reinforcement learning-based approach of [16] on the S&P 500. For comparison, we re-ran our trading simulation with the same in-sample data from 1/1997-12/1999 and out-of-sample period 1/2000-12/2012 from [16]. Our model achieved cumulative and annualized returns of 143.3% and 7.1%, respectively, outperforming the average case in [16] (80.78% and 4.66%, resp.). Although no SRs were reported from which to compare risk-vs-reward, our draw-down was higher, i.e., 29.76% vs 16% for the average case in [16], respectively. Compared to [13], our approach outperformed on the S&P500 and NASDAQ (42.63% vs 18.30% and 32.87% vs 27.67%, respectively) and lagged on the DJIA (5% vs 24.46%), while compared to [3] and [9], our approach yielded a significantly greater cumulative return on the S&P 500 (136% vs -17.96% and -15%, respectively, over their out-of-sample period, 2010-2015) and, as a result, a higher positive SR (0.9 vs negative for the other two works), noting however that those works traded a portfolio of stocks derived from the S&P500 and not the index (or ETF) itself. Looking to additional approaches such as [17], and matching that work's out-of-sample period (1/30/2015 - 04/13/2017) our model strongly outperformed in terms of cumulative returns for the SP500 (67.48% vs 18.89%) at the cost of significantly higher draw-down (-27.25% vs -9%). Against [18], and again testing on the out-of-sample period used therein, (2/2015 - 04/2017), our model achieved a cumulative return of 67.48% vs 35.21% reported in [18].

We go on to works with relatively short out-of-sample periods. Next to the model from [8] for the S&P500, using the same out-of-sample period (the year 2011), our approach attained a higher cumulative return of 44.19% vs 25% but a lower SR of 1.5 vs 2.6 and a higher draw-down of 11.77% vs 4%, although this comparison is made cautiously because unlike the previous works, which reported performance over multi-year periods, [8] had a 1-year evaluation period. Similarly, the work in [14] reports very good results in terms of SR (which were also better than ours) during their out of sample period 1/7/2016–12/23/2016 (1.09 vs 1.98[14]). Against the fuzzy network approach of [15] our model approach outperformed in terms of cumulative return (18.42% vs 11.90%) over their out-of-sample period 1/2/2009 - 6/30/2009, and slightly lagged on SR (0.7539 vs 0.8883) which is a direct result of our strategy exhibiting higher variance. However, these comparisons must be made cautiously in light of the short out-of-sample period.

We note that some of the works cited did not report SRs or draw-downs; in those cases we have compared in terms of cumulative and annualized returns only. Finally, our results are not directly comparable with those in works who report arithmetic *average* or *summed* returns only (e.g., [6] and others), from which unfortunately one cannot deduce annualized returns or other standardized measures of trading performance, as we have explained in Section 2.3.

[14]the SR in [14] was reported on an 6-month basis, the numbers have been adjusted to an annualized basis using the "standard" formula $SR = AR/AV$

5.4. Replicating other approaches and statistical forecasting techniques on our data set

Although our main focus is on comparing our own approach pair-wise against a number of others from the literature (as opposed to ranking existing works against each other), we performed some additional numerical experimentation by replicating some of the models mentioned in the previous section, [3, 7, 9, 14], as well as three benchmark statistical forecasting techniques, in the interest of providing a fuller picture on how multiple approaches would perform on a common, long out-of-sample period. We chose the aforementioned works because they were methodologically closest to our approach and domain of expertise. In each case, we carefully implemented the model described in the corresponding paper, trained it on the S&P500 index, and tested it on our out-of-sample data, 1/04/2010-12/20/2019, for comparison with our own models.

The comparisons are summarized in Table 14 where our approach outperforms, although it is important to note that the results were obtained via our own implementations of the other authors' works, and that numerical experimentation was required to infer some of the relevant parameters which were needed for training but were not listed in the corresponding works[15]. Overall, during our out-of-sample period, the four approaches did not perform better than the BnH benchmark. The positive-yet-

Model	CR	AR	AV	SR	DD
XGB	428.83	18.23	26.39	0.69	−21.20
RF	315.95	15.41	23.59	0.65	−22.90
LSTM-LH	347.16	16.25	24.01	0.68	−28.53
LSTM-AH	350.32	16.34	18.67	0.88	−16.94
[3]	41.23	3.56	53.48	0.07	−35.07
[7]	49.31	4.11	9.45	0.43	−5.89
[9]	85.64	6.47	27.47	0.24	−24.92
[14]	4.62	2.21	30.86	0.07	−13.45
BnH	188.87	11.25	14.79	0.76	−11.18

Table 14: Returns and risk statistics for our proposed architectures as well as those from [3,7,9,14], obtained for the same out-of-sample period (1/04/2010-12/20/2019), with the S&P500 index. *BnH* represents the buy-and-hold strategy in which the asset was bought once at the beginning of the trading period and sold at the end, RF: Random forest, XGB: Gradient boosted trees, LSTM-LH: LSTM network using only the last hidden state at the output layer, LSTM-AH: LSTM network using all hidden states. *CR* denotes cumulative return, *AR* is annualized return, *AV* is annualized volatility, *SR* is the Sharpe ratio, *DD* is draw-down, defined as the largest peak-to-trough percent decline during an investment's lifetime.

[15]To this end, we used for [3, 7, 9] batch size equal to 512, learning rate equal to 0.001, the list of 10 stocks used: MSFT,AAPL,AMZN,GOOG,UNH,JPM,JNJ,V,WMT,PG

low returns of [3, 9] seem to be in line with their 2010-2015 performance as presented in the original papers, which was negative (and thus seems to have "hurt" their overall performance over the first half of our data set). For [7], performance lagged compared to that during the time period cited by the authors (possibly stemming from the 2008-2009 market dip which is not part of our sample), but the corresponding draw-down was impressively low. With the model from [14], the strong performance over the six month period cited in that paper did not appear to be sustainable over the much longer out-of-sample period.

Finally, for the purposes of additional benchmarking, we compared our proposed approach against three well-known statistical forecasting techniques [33], namely ARIMA, Simple Exponential Smoothing (SES) and Holt's method. Their performance over our out-of-sample period is summarized in Table 15. Of the three, only the ARIMA model performed better than the BnH benchmark in the case of the

Model	Index	CR	AR	AV	SR	DD
ARIMA	SP500	247.27	16.15	16.08	1.00	−13.27
	DJIA	21.65	2.36	26.16	0.09	−17.71
	R2000	124.10	10.09	30.96	0.33	−15.62
	NASDAQ	210.01	14.44	30.23	0.48	−21.49
SEM	SP500	139.98	11.10	20.07	0.55	−16.14
	DJIA	105.24	8.94	16.96	0.53	−13.77
	R2000	74.87	6.89	29.99	0.23	−21.40
	NASDAQ	17.18	1.91	45.66	0.04	−42.60
Holt	SP500	174.11	2.63	18.67	0.14	−30.15
	DJIA	71.46	1.39	10.23	0.14	−18.05
	R2000	108.78	1.90	31.53	0.06	−42.85
	NASDAQ	132.84	2.18	24.51	0.09	−35.61
BnH	SP500	188.87	11.25	14.79	0.76	−11.18
	DJIA	172.87	10.61	14.08	0.75	−10.71
	R2000	167.34	10.38	19.60	0.53	−15.76
	NASDAQ	293.32	14.75	17.08	0.86	−12.45

Table 15: Returns and risk statistics for the out-of-sample period (1/04/2010-12/20/2019) for the ARIMA, Simple Exponential Smoothing (SEM) and Holt method. *CR* denotes cumulative return, *AR* is annualized return, *AV* is annualized volatility, *SR* is the Sharpe ratio, *DD* is draw-down, defined as the largest peak-to-trough percent decline during an investment's lifetime.

S&P500; all three techniques performed far worse than our proposed approach (see Table 12).

5.5. Model robustness with longer forecasting horizons

Although our results thus far focused on daily prediction and trading, we also tested the performance of our models using weekly as well as monthly forecasting horizons. The training and testing process was identical to what was discussed for the daily operation scenario (i.e grid search and hyper-parameter tuning for all architectures and all traded assets), this time using weekly and then monthly data. The performance results are summarized in Table 16, and are to be compared with those from the daily setting of Table 12. In Table 16, we have omitted AR and AV, (they can be inferred from the CR and SR values), to avoid clutter.

		Weekly			Monthly		
Index	Model	CR	SR	DD	CR	SR	DD
SP500	XGB	339.32	0.62	−31.63	324.84	1.08	-17.67
	RF	52.91	0.84	−12.45	84.25	0.24	-37.89
	LSTM-LH	286.80	1.79	−6.26	260.93	1.21	-13.27
	LSTM-AH	389.00	2.41	−7.73	422.99	2.12	-7.28
	BnH	188.87	0.76	−11.18			
DJIA	XGB	206.77	1.08	−26.52	296.24	1.01	-22.64
	RF	64.53	1.00	−12.11	29.95	0.13	-39.73
	LSTM-LH	260.59	1.09	−13.68	327.42	1.42	-14.84
	LSTM-AH	232.47	1.20	−12.76	340.23	1.93	-1.32
	BnH	172.87	0.75	−10.71			
R2000	XGB	145.93	0.42	−30.68	261.98	0.74	-21.37
	RF	77.51	0.76	−13.34	63.65	1.59	-19.18
	LSTM-LH	289.44	0.44	−43.33	358.68	1.16	-23.68
	LSTM-AH	273.12	0.57	−33.15	290.15	0.92	-26.73
	BnH	167.34	0.53	−15.76			
NASDAQ	XGB	151.60	0.35	−37.38	367.75	1.69	-5.49
	RF	73.45	0.85	−9.13	52.71	0.15	-37.16
	LSTM-LH	355.35	0.89	−18.26	340.56	1.30	-16.62
	LSTM-AH	466.61	1.20	−23.02	337.41	1.46	-15.77
	BnH	293.32	0.86	−12.45			

Table 16: Returns and risk statistics for the out-of-sample period (1/04/2010-12/20/2019) using weekly and monthly data. RF: Random forest, XGB: Gradient boosted trees, LSTM-LH: LSTM network using only the last hidden state at the output layer, LSTM-AH: LSTM network using all hidden states. *CR* denotes cumulative return, *SR* is the Sharpe ratio, *DD* is draw-down, defined as the largest peak-to-trough percent decline during an investment's lifetime.

In terms of profitability, the main observation is that the neural network models still performed very well, and appeared to show robustness under the different forecasting horizons. In most cases, the daily prediction horizon gave higher returns with the LSTM models, however both LSTM variants consistently beat he BnH benchmark by a wide margin, as well as the three statistical forecasting benchmarks, when switching to weekly and monthly forecasting. The LSTM-AH model always ranked first or second in profitability, with the exception of the NASDAQ using the monthly horizon. On the other hand, tree-based methods were generally lacking when the forecasting horizon was increased, with the exception of XGB using monthly forecasting. The RF model in particular performed much worse than BnH and gave best returns with a daily horizon.

A noteworthy difference over the daily prediction/trading scenario was observed in terms of SR, which was generally better over the longer prediction horizons, in some cases spectacularly so (e.g., for the S&P500, trading with the LSTM-AH model and weekly predictions gave a SR of 2.41 vs 0.88 with daily predictions). This appears to be due to the decreased volatility of weekly and monthly predictions compared to their daily counterparts. Thus, the proposed model(s) may provide interesting opportunities for high profitability and a favorable reward-to-risk ratio by adopting longer forecasting horizons. Finally, we note that for all indices, the maximum profitability across all models was achieved under a *daily* horizon; this might be expected because a shorter time horizon provides more opportunities for profitable trades, if the model and trading strategy can take advantage.

6. Conclusions and future work

Motivated by the complexity involved in designing effective stock price prediction models and accompanying trading strategies, as well as the prevalence of directional approaches, we presented a novel, systematic, model-agnostic approach to exploiting the profitability of predictions. The trading decision-making policy attempts to glean more information from the distribution of those predictions, compared to the oft-used directional "up/down" strategy. Our approach did not focus directly on constructing a more precise or more directionally accurate prediction scheme; instead, we selected the model's hyper-parameters to identify the most profitable model variant, instead of the one with lowest prediction error, in an effort to regard the prediction and decision process as a whole, and recognizing the fact that predictive accuracy alone does not guarantee profitability. At the same time, our proposed trading strategy exploited the fact that the location of a prediction within its distribution carries information about the expected profitability of a trade to be executed based on that prediction.

The performance of our proposed models was tested on four major US stock indices, namely the S&P500, the DJIA, the NASDAQ and the Russell 2000 over the period 1/4/2010 - 12/20/2019. Besides beating the indices themselves, our approach also outperformed those proposed in several recent works, in terms of the cumulative or annualized returns attained. With respect to long-term risk profile, our results were better than those of the stock indices (buy-and-hold) but more volatile than some in the recent literature. Of course, the attractiveness of high returns as well as risk-reward ratio are largely investor-dependent.

Of the models tested, the best overall in terms of cumulative and annualized return, annualized volatility and draw-down, was the deep LSTM-AH neural network proposed in this work. It ranked first in most test cases, outperforming the other models (with with the exception of XGB on the S&P500, and the LSTM-LH on the R2000), while having the smallest annualized volatility and lowest draw-down in all but one cases. The LSTM-AH model also outperformed the buy-and-hold benchmark every time, by a wide margin. An important factor contributing to the network's effectiveness was the use of the entire history of the LSTM cell's hidden states as inputs to the output layer as the network was exposed to an input sequence.

Opportunities for future work include further experimentation with more sophisticated allocation and trading strategies, the possible inclusion of short sales, as well as the use of a variable portion of the time history of the LSTM hidden states when making predictions, to see where the optimum lies. Also, it would be of interest to optimize the manner in which the percentiles of the predicted return distribution are chosen when forming our allocation strategy, in order to maximize profitability and reduce risk. Finally, one could in principle use the proposed distribution-based trading strategy with any other price prediction scheme, and it would be interesting to see what performance gains, if any, could be achieved that way.

References

[1] Eugene F. F. and Kenneth R. F. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1):3–56, 1993.

[2] M. Sethi, P. Treleaven, and S. Del Bano Rollin. Beating the S&P 500 index — a successful neural network approach. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 3074–3077, July 2014.

[3] C. Krauss, X. Do, and N. Huck. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research*, 259(2):689–702, June 2017.

[4] Eugene F. F. and Kenneth R. F. The capital asset pricing model: Theory and evidence. *Journal of Economic Perspectives*, 18(3):25–46, September 2004.

[5] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Trans. on Neural Networks and Learning Systems*, 28(3):653–664, March 2017.

[6] W. Bao, J. Yue, and Y. Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLoS ONE*, 12(7): e0180944, 2017.

[7] Y. Baek and H. Y. Kim. ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module. *Expert Systems with Applications*, 113:457–480, December 2018.

[8] F. Zhou, H. Zhou, Z. Yang, and L. Yang. EMD2FNN: A strategy combining empirical mode decomposition and factorization machine based neural network for stock market trend prediction. *Expert Systems with Applications*, 115:136–151, 2019.

[9] T. Fischer and C. Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, October 2018.

[10] X. Zhong and D. Enke. Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications*, 67:126–139, 2017.

[11] G. Leitch and J. E. Tanner. Economic forecast evaluation: Profits versus the conventional error measures. *American Economic Review*, 81(3):580–90, 1991.

[12] O. Sezer and M. Ozbayoglu. Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. *Applied Soft Computing*, 70:525–538, 04 2018.

[13] M. Thakur and D. Kumar. A hybrid financial trading support system using multi-category classifiers and random forest. *Applied Soft Computing*, 67:337 – 349, 2018.

[14] F. Zhou, Q. Zhang, D. Sornette, and L. Jiang. Cascading logistic regression onto gradient boosted decision trees for forecasting and trading stock indices. *Applied Soft Computing*, 84:105747, 2019.

[15] P. Chang, J.-L. Wu, and J.-J. Lin. A Takagi-Sugeno fuzzy model combined with a support vector regression for stock trading forecasting. *Applied Soft Computing*, 38:831 – 842, 2016.

[16] D. Eilers, C. L. Dunis, H.-J. von Mettenheim, and M. H. Breitner. Intelligent trading of seasonal effects: A decision support algorithm based on reinforcement learning. *Decision Support Systems*, 64:100–108, 2014.

[17] J. Nadkarni and R. Neves. Combining neuroevolution and principal component analysis to trade in the financial markets. *Expert Systems with Applications*, 103, 03 2018.

[18] J. Nobre and R. Neves. Combining principal component analysis, discrete wavelet transform and XGBoost to trade in the financial markets. *Expert Systems with Applications*, 125(1):181–194, July 2019.

[19] E. Chong, C. Han, and F. C. Park. Deep learning networks for stock market analysis and prediction: methodology, data representations, and case studies. *Expert Systems with Applications*, 83(C):187–205, October 2017.

[20] S. Gu, B. T. Kelly, and D. Xiu. Empirical asset pricing via machine learning. Swiss Finance Institute Research Paper No. 18-71, `http://dx.doi.org/10.2139/ssrn.3281018`, June 2018.

[21] Yahoo. Yahoo Finance, Symbol Lookup (2018), 2018. `https://finance.yahoo.com/quote/` [Accessed: May 1, 2018].

[22] M. Ballings, D. Van den Poel, N. Hespeels, and R. Gryp. Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, 42(20):7046–7056, November 2015.

[23] Y. Xie, X. Li, and W. Ying E. W. T. Ngai. Customer churn prediction using improved balanced random forests. *Expert Systems with Applications*, 36(3):5445–5449, April 2009.

[24] C. Lindner, P. A. Bromiley, M. C. Ionita, and T. F. Cootes. Robust and accurate shape model matching using random forest regression-voting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1862–1874, Sep. 2015.

[25] J. Friedman T. Hastie, R. Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, second edition, New York, 2000.

[26] A. Geron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, first edition, 2017.

[27] B. Krause, L. Lu, I. Murray, and S. Renals. Multiplicative LSTM for sequence modelling, 2016. arxiv:1609.07959.

[28] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[29] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, Th. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550:354–359, Ooctober 2017.

[30] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[32] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[33] Spyros Makridakis, Evangelos Spiliotis, and Vassilis Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PLoS ONE*, 13, 03 2018.