Draft version April 28, 2025
Typeset using LaTeX **default** style in AASTeX7

Comparative Analysis of Deep Learning Models for Price Prediction and Portfolio Construction of US ETFs: LSTM vs. Random Forest

Fusheng LUO,[1] Mia EDMONDS,[1] Yuxin CHEN[1] And Wenzhe CHEN[1]
Equity Markets and Quantitative Trading

[1] *Johns Hopkins University*

1. INTRODUCTION

Over the past few decades, exchange-traded funds (ETFs) have steadily grown from a financial innovation to a major investment tool with global relevance. Their appeal lies in a combination of traits: they're flexible, relatively transparent, and generally low-cost. Since their introduction into North America during the early 1990s, ETFs have diversified extensively, now covering equities, bonds, commodities, currencies, and even derivatives. What once managed a few billion dollars has grown into a multi-trillion-dollar industry, reshaping how investors, both large and small, approach the markets (Hill et al. 2015).

In addition to the general advantages above, ETFs have several design features that give them additional appeal to investors. First, ETFs are traded on stock exchanges so investors can purchase or sell them at market prices during the day. In contrast, mutual funds settle only once a day. More importantly, this intraday liquidity on an exchange is supplemented by an "inkind" creation and redemption process involving Authorized Participants (AP) that helps keep ETF prices close to their underlying value.1 In particular, this creation and redemption process is arbitrage, driving a tight spread between the market price and the net asset value (NAV) (Gastineau 2001). Another important design feature of ETFs is that they are generally passively managed to track specific indexes. As a result, they tend to have lower management costs and greater transparency. In practice, most funds will disclose their holdings on a daily basis (Madhavan 2016).

When it comes to how ETFs are constructed, the methodology is typically based on rules. That is, the rules determine what goes into the portfolio, which often follows a specific index strategy. For example, an equity ETF might buy stocks by market cap or by sector. A bond ETF might buy by credit quality or duration. When assessing how an ETF is constructed, one important metric is the tracking error, or the difference between the ETF's performance and its underlying benchmark. Interestingly, how an ETF reproduces its index (complete replication, sampling, or synthetic instruments) can lead to differences in tracking precision (Johnson 2009).

There is another interesting feature of ETFs that has to do with how new shares are issued and existing shares are redeemed. Instead of using cash transactions, APs deliver or receive a basket of securities. This is important because, if an ETF begins to trade at a discount to the value of its holdings, arbitrage will drive it back to that value. Furthermore, this means that long-term shareholders do not pay the costs of others entering and exiting the fund (Madhavan & Sobczyk 2016), which is not always the case with mutual funds.

In recent years, there has been increasing interest in using deep learning on finance data, especially when the data at hand are time-series, such as stock prices. Of the tools considered for use with deep learning of finance data, Long Short-Term Memory (LSTM) networks have attracted attention. LSTM is a type of recurrent neural network (RNN). It is designed to handle long-term dependencies better than standard RNNs. This is important because traditional neural networks have struggled with vanishing gradients. Because of this, LSTMs are being used more often to identify patterns in financial data. Specifically, they are used to predict financial data in markets, which are known to be particularly difficult to model, e.g. ETFs (Nelson et al. 2017b).

There are a few ETFs that we tend to see used in financial modeling and machine learning due to popularity and trading volume. SPY is widely followed and tracks the S&P 500 Index. QQQ focuses on the Nasdaq-100 and gives exposure to large tech. Investors seeking higher returns might look into HYG, which invests in high yield corporate bonds (and thus has significant credit risk). XLF targets the financial sector. And investors looking to hedge their

interest rate exposure might look into TLT, which gives exposure to long-term U.S. Treasury bonds. These funds tend to be used heavily in LSTM research due to their deep trading history.

Moreover, ETFs are cheap to trade. Their expense ratios are typically lower than 0.1% compared to more than 1% for active fund offerings (Poterba & Shoven 2002). In addition, the in-kind redemption mechanism makes ETFs tax efficient as it reduces capital gains distributions. This can make a significant difference in the long run and its compounded effect.

The ETF market has evolved to include far more than just index-tracking products. Today, investors can choose from thematic ETFs, smart beta strategies, leveraged or inverse products, and even actively managed funds. However, this explosion of variety has raised some concerns. For example, research by Israeli et al. (2017) points out that ETF trading may affect the behavior of underlying securities—especially when high-frequency trading is involved. Ben-David et al. (2018) even argue that large ETF flows can add to short-term market instability under certain conditions.

Academic interest in ETFs continues to grow, especially around questions of market behavior and price discovery. Hamm (2014), for instance, shows that ETFs can help reveal fair prices during times when underlying markets are closed—such as with international or commodity-linked ETFs. At the same time, leveraged and inverse ETFs have drawn scrutiny due to their complexity and how volatility can compound over time (Pan & Zeng 2019).

Ultimately, this study tries to bring together the key insights—both theoretical and empirical—around how ETFs operate and influence the broader market. From their structural design to their impact on liquidity and pricing, ETFs are clearly more than just a trading tool. As Schoenfeld (2004) noted years ago, the growth of ETFs marks a major shift in how modern portfolios are built. That shift only seems to be accelerating.

A promising direction for ETF price forecasting is the application of deep-learning methods, in particular long short-term memory (LSTM) networks, which can capture complex, non-linear temporal dependencies that linear models often miss (Singh & Yadav 2024). Empirical work on size-performance relationships has shown that ETFs do not adhere to constant returns-to-scale, exhibiting instead non-linear scale effects that challenge the assumptions of traditional regression frameworks (Paudel & Naka 2023). Likewise, factor-based analyses demonstrate that linear factor models leave significant residual structure in ETF returns, suggesting richer interactions among market exposures than can be captured by standard multifactor regressions (Blitz & Vidojevic 2021). LSTM models, by contrast, are specifically designed to learn from sequences of past observations—automatically weighting recent versus more distant information and adjusting to regime shifts—making them well-suited to predict intraday and daily price movements of ETFs with potentially higher accuracy than classic econometric or factor-based techniques.

2. DATA SOURCES

Our ETF data is sourced from Yahoo Finance. The raw dataset initially comprises five key features: "Low," "High," "Close," "Open," and "Volume." The "Low" feature represents the lowest trading price of the day, while "Close" indicates the final trading price. "Open" reflects the initial price at which trading commenced, and "High" denotes the highest price reached during the trading day. Meanwhile, "Volume" captures the total trading volume accumulated throughout the day.

These five features provide a limited basis for developing effective deep learning models, which underscores our decision to create additional features derived from the existing price and volume data in the subsequent section.

The dataset we are working with is substantial, covering the period from February 14, 2014, to February 21, 2025. We have divided our dataset accordingly: 75% is allocated for training (spanning from the start until May 22, 2022), 10% is designated for validation (from May 2, 2022, to April 1, 2023), and the remaining portion is set aside for testing (from April 1, 2023, to February 20, 2025). Additionally, the dataset includes 95 tickers, as detailed in Table 1.

3. FEATURES BUILDING AND DATA PREPROCESSING

We start our analysis with daily time series data. The data includes open, close, high, low prices, and volume. First, we perform a log-return transformation on prices to make them stationary. Next, we use min-max scaling to normalize volume data. Then, exponential smoothing with an $\alpha = 0.6$ is applied. This smoothing gives more weight to recent data points, helping the model respond better to recent trends and improving the prediction capability.

When using exponential smoothing with (`adjust=False`), the calculation becomes recursive and follows the formula:

$$s_t = \alpha \cdot x_t + (1 - \alpha) \cdot s_{t-1}$$

where:

Table 1. List of 95 ETF Tickers

AGG	BND	DBC	DIA	DVY
EEM	EFA	EMB	EWA	EWC
EWG	EWH	EWJ	EWU	GLD
HYG	IAK	IAT	IAU	IBB
ICF	IDU	IEF	IGV	IHE
IHF	IHI	IJJ	IJK	IJS
IJT	ITB	ITOT	IUSG	IUSV
IVV	IWB	IWD	IWF	IWM
IWN	IWO	IWP	IWS	IYC
IYE	IYF	IYG	IYH	IYK
IYM	IYR	IYW	LQD	PFF
QQQ	REM	SCHA	SCHB	SCHE
SCHF	SCHX	SCHZ	SDY	SHY
SLV	SOXX	SPY	TLT	UNG
USO	VB	VBK	VBR	VEU
VIG	VNQ	VO	VOE	VOO
VOT	VTI	VTV	VUG	VWO
VYM	XLB	XLE	XLF	XLI
XLK	XLP	XLU	XLV	XLY

- s_t is the smoothed value at time t,
- x_t is the actual value at time t.

Here, alpha is the smoothing factor, which controls how much weight is assigned to recent observations:

- A higher alpha (e.g., 0.5) gives more weight to recent data, allowing the smoothing to respond more quickly to changes.
- A lower alpha (e.g., 0.1) results in heavier smoothing, meaning the response to recent changes is slower.

Additionally, we add several technical indicators to our dataset. These indicators include Relative Strength Index (RSI), Moving Average Convergence/Divergence (MACD), Bollinger Bands, and On-Balance Volume (OBV). This results in a total of 14 features. Finally, we use one-hot encoding to represent different ticker symbols clearly. After these preprocessing steps, the dataset has a shape of (2764, 1177).

Below is a clear description of each technical indicator:

3.1. *Relative Strength Index (RSI)*

RSI is a momentum oscillator that compares the magnitude of recent gains to recent losses to measure the speed and change of price movements. It ranges from 0 to 100. Values above 70 typically indicate an "overbought" (potentially overvalued) market, while values below 30 indicate an "oversold" (potentially undervalued) market.

$$RSI = 100 - \frac{100}{1 + RS},$$

where RS is the relative strength.

The calculation steps are as follows:

(i) **Compute the daily price change:**

$$\Delta P_i = P_i - P_{i-1}.$$

(ii) **Define the gains and losses:**

$$G_i = \max(\Delta P_i, 0) \quad \text{and} \quad L_i = \max(-\Delta P_i, 0).$$

(iii) **Compute the initial average gains and losses:**

$$\text{AvgGain}_0 = \frac{1}{N}\sum_{i=1}^{N} G_i, \quad \text{AvgLoss}_0 = \frac{1}{N}\sum_{i=1}^{N} L_i.$$

(iv) **Smooth the averages using Wilder's method for $t > N$:**

$$\text{AvgGain}_t = \frac{(\text{AvgGain}_{t-1} \times (N-1)) + G_t}{N}, \quad \text{AvgLoss}_t = \frac{(\text{AvgLoss}_{t-1} \times (N-1)) + L_t}{N}.$$

(v) **Calculate the relative strength (RS) and RSI:**

$$RS = \frac{\text{AvgGain}}{\text{AvgLoss}}.$$

3.2. *Moving Average Convergence Divergence (MACD)*

MACD is a trend-following momentum indicator that shows the relationship between two moving averages of a security's price. These are the short-term 12 days EMA and the long-term 26 days EMA. MACD has three components:

(i) MACD Line: Shows raw momentum change by calculating the difference between short-term and long-term EMAs.

(ii) Signal Line: An EMA (commonly 9 periods), which acts as a smoothed version to help filter noise.

(iii) MACD Histogram: Highlights the difference between MACD and Signal lines to visualize crossovers and divergences. The Histogram measures the distance between these two lines, giving insight into the strength and direction of the momentum.

Calculate the Exponential Moving Averages (EMAs): For a time period N, the EMA is defined as:

$$EMA_t = \alpha\, P_t + (1-\alpha)\, EMA_{t-1},$$

where the smoothing factor is:

$$\alpha = \frac{2}{N+1}.$$

For the 12-day EMA, $\alpha_{12} = \frac{2}{12+1} \approx 0.1538$, and for the 26-day EMA, $\alpha_{26} = \frac{2}{26+1} \approx 0.0741$.

(i) **Compute the MACD Line:**

$$\text{MACD Line} = EMA_{12} - EMA_{26}.$$

(ii) **Compute the Signal Line (a 9-day EMA of the MACD Line):**

$$\text{Signal Line} = EMA_9(\text{MACD Line}).$$

(iii) **Calculate the MACD Histogram:**

$$\text{MACD Histogram} = \text{MACD Line} - \text{Signal Line}.$$

3.3. *Bollinger Bands*

Bollinger Bands are used to gauge market volatility (this is the reason why we don't calculate the realized volatility here) and define dynamic support and resistance levels. The bands consist of:

- **Bollinger_MAVG**: the moving average (often a simple moving average) of the closing prices over a defined window (20 periods here).

$$\text{Bollinger\_MAVG} = \frac{1}{N}\sum_{i=1}^{N} P_i.$$

- **Bollinger_High (Upper Band)**: Calculated by adding a multiple (typically 2) of the price's standard deviation to the Bollinger moving average. It indicates a level where the price is considered relatively high or potentially overbought. Calculated as:

$$\text{Bollinger\_High} = \text{Bollinger\_MAVG} + k\sigma,$$

where k (typically 2) is the multiplier and σ is the standard deviation of the prices over N periods.

- **Bollinger_Low (Lower Band)**: Calculated by subtracting the same multiple (typically 2) of the price's standard deviation from the Bollinger_MAVG. It indicates a level where the price is considered relatively low or potentially oversold. Calculated as:

$$\text{Bollinger\_Low} = \text{Bollinger\_MAVG} - k\sigma.$$

MACD captures momentum and trend shifts by analyzing the difference between EMAs and smoothing it with a signal line, while Bollinger Bands measure volatility and show how prices move relative to a moving average. Together, they provide complementary insights into market behavior.

3.4. *On-Balance Volume (OBV)*

The On-Balance Volume (OBV) indicator is a momentum metric based on volume. It accumulates volume by adding it on days when the closing price rises and subtracting it when the closing price falls. This running total reflects potential buying or selling pressure. OBV is particularly useful for capturing the direction and relative shifts in volume, making it well-suited for use with our normalized volume data.It is calculated as:

$$OBV_t = OBV_{t-1} + \text{Volume}_t \times \text{sgn}(P_t - P_{t-1}),$$

where the sign function is defined as:

$$\text{sgn}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases}$$

4. TRADING STRATEGIES

We employ two regression models, random forest (RF) and long short-term memory (LSTM), to forecast the percentage return for the following day. Our long-short trading strategies are developed based on a market-neutral assumption. Each day within the test period, we begin by ranking our predictions from highest to lowest, resulting in 95 ranked tickers. We then classify the top 30% as "+1" (specifically, 28 tickers receive this label) and the bottom 30% as "−1," while the remaining tickers are assigned a label of "0." This relabeling process is repeated daily to better reflect the ever-changing dynamics of the market.

With these daily labels, we take a long position of \$1 on each "+1" ticker and a short position of \$1 on each "−1" ticker. Consequently, our cash account will have a net position of zero at the end of the trading day. Throughout the test period, we adjust our portfolios daily to evaluate our potential profitability by the conclusion of the trading period. As Figure 1 shows, it is the glimpse of our labeling method.

5. LONG SHORT-TERM MEMORY MODEL (LSTM)

5.1. *LSTM introduction*

The Long Short-Term Memory (LSTM) network, introduced by Sepp Hochreiter and Jürgen Schmidhuber in 1997, is a specialized type of Recurrent Neural Network (RNN) designed to address the **vanishing/exploding gradient problem** in traditional RNNs. This issue hinders standard RNNs from learning long-term dependencies in sequential data. LSTMs excel in tasks requiring memory over extended sequences, such as time-series forecasting, natural language processing (NLP), and speech recognition. Specifically in our work, the daily OHLC prices and volume data for the ETFs are classic time-series data, applying the LSTM to our predictions should outperform many of the traditional methods due to the superiority of the LSTM in handling sequential datasets.

	Date	Ticker	Close	Close_scaled	prediction	rank	n_assets	signal
2169	2022-12-19	AGG	-0.006140	-1.905875	-0.008231	89.0	95	-1
2170	2022-12-20	AGG	-0.006786	-2.104053	0.488872	53.0	95	0
2171	2022-12-21	AGG	0.002753	0.823000	0.927987	53.0	95	0
2172	2022-12-22	AGG	-0.000305	-0.115555	0.662376	47.0	95	0
2173	2022-12-23	AGG	-0.003458	-1.083116	0.615147	45.0	95	0
...	...	...	...	...	...	...	...	...
13555	2025-02-12	DVY	-0.005399	-0.532048	-0.022436	40.0	95	0
13556	2025-02-13	DVY	0.009369	0.816135	-0.037915	53.0	95	0
13557	2025-02-14	DVY	-0.000221	-0.059357	-0.051284	53.0	95	0
13558	2025-02-18	DVY	0.010169	0.889093	-0.085456	68.0	95	0
13559	2025-02-19	DVY	0.004523	0.373680	-0.100534	75.0	95	0

Figure 1. A glimpse of how our labeling works in general

The LSTM comes out of the ordinary because of its unique inner structure, which consists of forget gates and input/output gates.

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{1}$$

where h_t is the hidden state at time t, c_t is the cell state at time t, x_t is the input at time t, h_{t-1} is the hidden state at time $t-1$, i_t, f_t, g_t, o_t are the input, forget, cell, and output gates, σ is the sigmoid function, $\odot$ is the Hadamard product.

The workflow in Figure 2 exemplifies the LSTM's core computational steps:

- **Forget Gate Dynamics**: The upper path values (+1.62, ×2.70) correspond to:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \approx 0.9 \quad \text{(sigmoid-scaled)} \tag{2}$$

  where the ×2.70 operation represents $f_t \odot c_{t-1}$.

- **Input Gate Processing**: The middle section's ×1.65 and ×2.00 operations reflect:

$$i_t \odot g_t = \sigma(W_i \cdot [h_{t-1}, x_t]) \odot \tanh(W_c \cdot [h_{t-1}, x_t]) \tag{3}$$

- **Output Generation**: The ×4.38 scaling demonstrates:

$$h_t = o_t \odot \tanh(c_t), \quad o_t = \sigma(W_o \cdot [h_{t-1}, x_t]) \tag{4}$$

This visualization concretely shows how LSTMs maintain separate but interacting memory pathways through learned arithmetic operations.

5.2. *Model Architecture*

We employ a deep recurrent neural network with attention for regression tasks, specifically designed for the ETFs' time-series prediction. The model takes as input a sliding window of 20 consecutive trading days of historical features and outputs a prediction for the return (percentage change of daily close price) of the next trading day (compared to

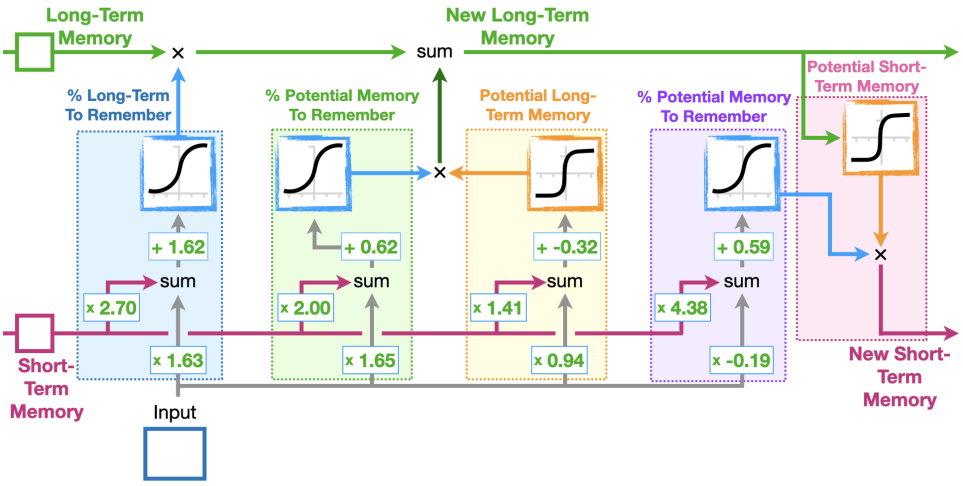

Figure 2. Visualization of LSTM memory cell operations during one timestep. The diagram illustrates three key processes: (1) **Long-Term Memory** modulation through forget gate operations (top path, showing scaling by $\times 2.70$), (2) **Short-Term Memory** integration via input gates (middle section, with $\times 1.65$ input transformation), and (3) **Memory State Update** through output gating (bottom paths, featuring $\times 4.38$ output scaling). The arithmetic operations demonstrate how additive ($+$) and multiplicative ($\times$) transformations implement LSTM's gating mechanisms.

the last trading day). It consists of a three-layer stacked Long Short-Term Memory (LSTM) network with a hidden size of 64. Each input sequence has shape $(20, 15)$, where 15 denotes the number of features per day. To enhance temporal interpretability, an additive attention mechanism is applied: it computes a context vector as a weighted combination of hidden states of LSTM, where the weights reflect the relevance of each timestep for the prediction (the weights are adjusted through "attention scores"). This context vector is then passed through a fully connected layer to produce a scalar regression output. The figure 3 schematically showcases the architecture of our LSTM model.

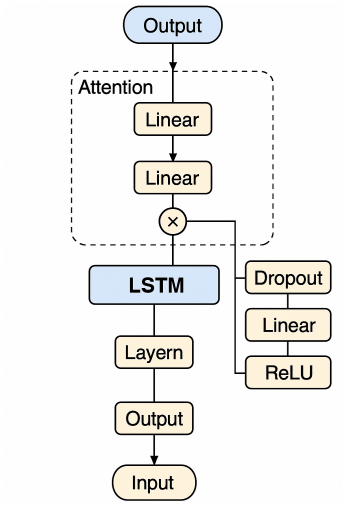

Figure 3. Detailed architecture of the proposed LSTM-Attention model. The model includes input layer, LSTM layer, normalization layer (LayerNorm), attention mechanism, dropout layer and linear layer with some hidden layers as well.

5.3. *Hyperparameter Tuning*

Due to the substantial computational expense associated with training our LSTM model, we confined our investigation to a limited set of hyperparameter configurations. A random search was subsequently performed across the entire

training dataset to identify the optimal combination of these parameters. Model performance was rigorously evaluated using several metrics, including validation loss, mean absolute error (MAE), root mean squared error (RMSE), and the coefficient of determination (R^2). The experimental outcomes corresponding to the various hyperparameter settings are summarized in Table 2.

	Search Space	Optimal Value
Hidden size	$\{32, 64, 128\}$	64
Epoch	1 to 40	16
Emb dimension	$\{8, 16, 32\}$	16
Numbers of LSTM layers	$\{2, 3, 4\}$	3
Dropout rate	$\{0.0, 0.1, 0.2, 0.3\}$	0.1
Learning rate	$\{1e-5, 1e-4, 1e-3\}$	$1e-5$
Weight decay	$\{1e-6, 1e-5, 1e-4\}$	$1e-6$

Table 2. Hyperparameter tuning in the search space. Note that "Emb dimension" is a technique named "learnable embeddings", the embedding dimension which determines the size of the embedding vectors for our categorical input indices. Among the input features, the tickers' name is also one feature fed into the model, and by one-hot encoding we transfer the tickers' name into a vector containing only 0 and 1. In a nutshell, this value is the dimension of the vector where we transfer 95 Boolen values into (there are 95 ETFs tickers).

5.4. *Auxiliary Loss Supervision*

The LSTM model in our dataset is struggling to deal with the daily percentage returns; consequently, it generates a negative R^2 score value in table 3 when trying to use the "close price" as the prediction targets. Here we illustrate several factors for this negative phenomenon:

1. In the testing periods, we can see an economic recession due to the COVID-19 pandemic. This may impair the predictability of our model as the training phase does not contain such recession.

2. The loss function in this context is not the mean squared error loss; rather, it is the Smooth L1 loss, also known as Huber Loss. This loss function is designed to perform more effectively in the presence of outliers, as it behaves like mean squared error (MSE) for small errors and like mean absolute error (MAE) for larger errors. The transition between these behaviors is controlled by a parameter, denoted as β, which typically ranges between 0 and 1.

3. Percentage returns are inherently noisy. Daily returns, in particular, exhibit high variance, a low signal-to-noise ratio, and weak autocorrelation, making their properties challenging for LSTM models to grasp. Consequently, our model tends to adopt a conservative learning approach.

Alternatively, Nelson et al. (2017a) employs the "close price" after applying exponential smoothing as the target, achieving an accuracy of over 55.9%. In this section, we aim to implement a similar strategy using the "close_smoothed" features developed in section 3.

We incorporate *auxiliary loss supervision* into our LSTM model by modifying the output to be two-dimensional; specifically, we are predicting both "close_smoothed" and "close price" simultaneously. We utilize mean squared error (MSE) loss for both predictions with equal weighting, integrating them into the model's overall loss function. This approach has allowed us to avoid negative R^2 values, leading to improved performance of the model compared to its previous results.

6. RANDOM FOREST

6.1. *Random Forest Introduction*

The Random Forest algorithm, introduced by Leo Breiman in 2001, is an ensemble learning method that constructs a multitude of decision trees during training and outputs the average prediction of the individual trees. It was designed to address the high variance and overfitting often encountered with single decision trees, especially in datasets with noisy

or complex relationships. Random Forests are particularly well-suited to structured tabular data and have become a standard baseline in many machine learning applications due to their robustness, interpretability, and minimal need for hyperparameter tuning. In our work, where we use daily ETF prices, volume, and derived technical indicators as features, Random Forests offer a powerful non-parametric approach for modeling complex interactions in financial time series data, particularly when sequential dependencies are less dominant than feature interactions.

The strength of Random Forest lies in its use of bootstrap aggregation and feature randomness, which together reduce correlation between trees and lead to better generalization on unseen data.

6.2. *Model Architecture*

To model ETF performance and provide a comparison for our LSTM network, a Random Forest regressor was implemented. To evaluate the predictive performance of a Random Forest Regressor on financial time series data, we trained the model using historical data up to the end of 2022 and tested it on data from 2023 onward. The model was tasked with predicting the "close_scaled" variable, a normalized version of the ETFs' closing price. Feature selection included a curated set of technical indicators and engineered variables that capture market dynamics, trends, and volatility patterns. By using only data prior to 2023 for training, the setup simulates a realistic forecasting environment that avoids look-ahead bias.

The performance of the model was assessed using standard regression metrics such as mean absolute error (MAE), root mean squared error (RMSE), and the coefficient of determination (R^2). This provided information on its precision and robustness in unseen data.

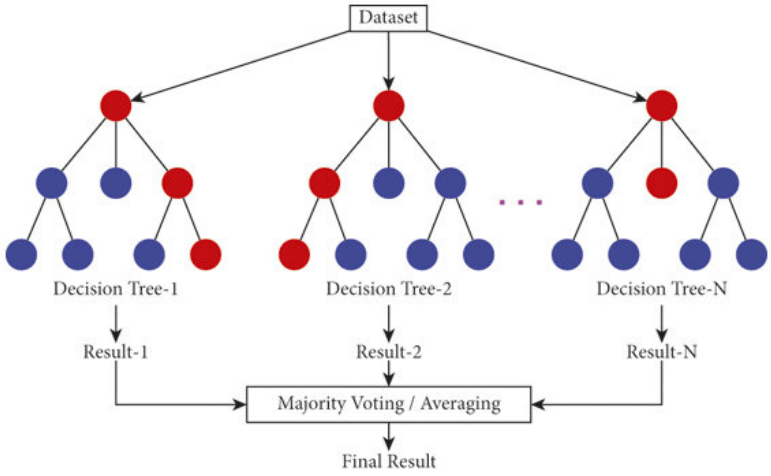

Figure 4. Visualization of a Random Forest Model with three trees

6.3. *Hyperparameter Tuning*

The Random Forest model was tuned using a GridSearchCV procedure, in which we systematically tested a range of values for two key hyperparameters: the number of estimators (i.e., trees in the forest) and the maximum depth of each tree. This allowed us to identify the configuration that best balanced predictive accuracy with model simplicity. The final model employed 250 estimators and a maximum tree depth of 5, which provided strong generalization while avoiding overfitting to the noise inherent in financial time series data. A relatively shallow tree depth was especially important in this context to reduce variance and prevent the model from memorizing idiosyncratic patterns in the training set. For the remaining parameters, such as min_samples_split, min_samples_leaf, and max_features, we retained the default values provided by the scikit-learn implementation. These defaults are optimized for general purpose performance and helped keep the model interpretable and computationally tractable. Given the high dimensionality of our dataset and the large number of daily observations across 95 ETFs, a full grid search across all hyperparameters would have significantly increased computational cost with only marginal potential performance improvements. By focusing our tuning on the most impactful parameters, we achieved a practical balance between model performance and runtime efficiency.

6.4. *Feature Importance*

To check the importance of our features we used sklearns feature_importances_. The feature importance results reveal that Low_scaled is by far the most influential predictor in the model, contributing approximately 61.9% of the total importance. This is followed by High_scaled, which accounts for about 32.3%, indicating that the high and low price levels play a dominant role in the model's predictions. Open_scaled also provides some value, though much less significantly at around 4.4%. The remaining features, including MACD_Hist, MACD_Line, RSI_14, MACD_Signal, Volume, OBV and the various Bollinger Band measures, collectively contribute very little to the model, each with an importance well under 1%. Encoding the ETF tickers as features contributed little to the prediction model, with their feature importance values all near zero. These results suggest that while technical indicators such as MACD, RSI, and Bollinger Bands are included, they have minimal impact compared to basic scaled price information, particularly the low and high prices.

7. RESULTS

7.1. *LSTM*

As we are building regression models, we use related regression metrics to evaluate the predictability of our models. Table 3 gives the values of different metrics from three models. The "Naive Baseline" model is just a simple model which uses the mean value of the training targets as prediction consistently. In particular, after our normalization, the R^2 score of this Naive Baseline model should be around 0, with small difference because of the fact that the training dataset is only proportion of the whole dataset. This is just a vanilla model that we want to set a comparison with. The metrics contain "RMSE", "MAE" and "R^2", they are the root mean square error, mean absolute error and R^2 score.

One abnormal observation is that for two of the models, we have negative R^2 scores. A zero value of R^2 score generally implies predicting the average. Hence, our negative values tell us that our model may even be worse than "simply predicting the mean". After applying the *auxiliary loss supervision*, our model performs better in the prediction of the "close_smoothed" values, as illustrated in Figure 6, the distribution of our predictions compared to the actual returns indicates a general tendency to underfit the true distribution in "close price" but a good fit in "close_smoothed".

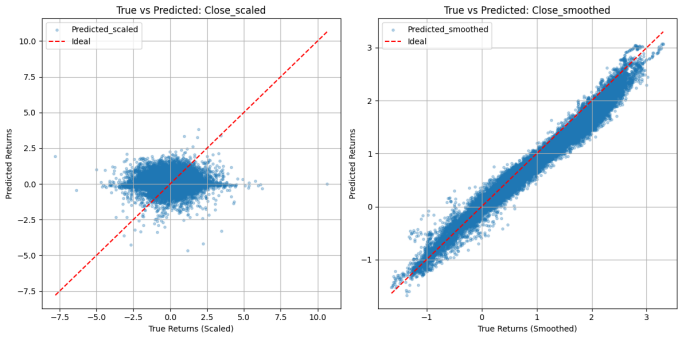

Figure 5. True Returns vs Predicted Returns of the LSTM Model

	LSTM on close	LSTM on smooth	Naive Baseline	Random Forest
RMSE	0.905	0.203	0.858	0.428
MAE	0.657	0.174	0.651	0.321
R^2	-0.013	0.928	-0.001	0.749

Table 3. Comparison of regression metrics on the test set among different models, Naive Baseline model is just using the mean value of the training dataset as the only prediction

Despite these limitations, the LSTM model's output still supported a reasonably profitable trading strategy. The long-short strategy based on LSTM predictions achieved an annualized return of 7.60% and a Sharpe ratio of 1.190 (Table 4), indicating that even weak point forecasts can yield valuable ranking signals for trading. The strategy

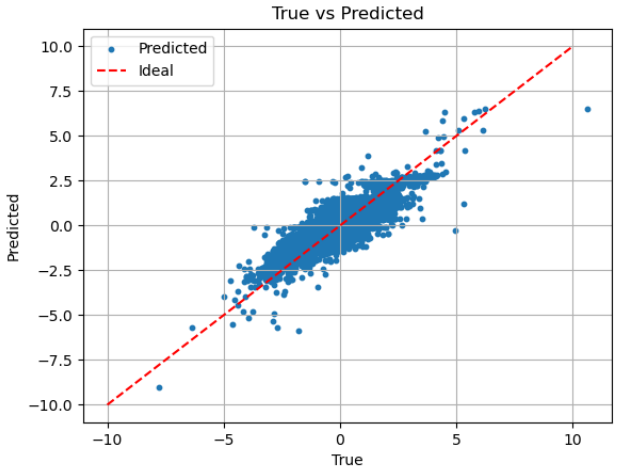

Figure 6. True Returns vs Predicted Returns of the Random Forest Model

experienced relatively low volatility and drawdown, suggesting robustness across diverse market conditions. Figure 8 confirms steady growth over the test period.

7.2. *Model Evaluation and Trading Performance*

We evaluate the performance of our predictive models on the test set using standard regression metrics (Table 3) and assess the trading strategies derived from these predictions using financial performance metrics (Table 4).

Table 3 presents a comparison of root mean squared error (RMSE), mean absolute error (MAE), and R^2 across four approaches: LSTM on raw close prices, LSTM on smoothed prices, a naive baseline model, and a Random Forest model. The LSTM trained on smoothed prices achieves the best predictive performance, with the lowest RMSE (0.203) and MAE (0.174), and a strong R^2 value of 0.928. In contrast, the LSTM trained on raw close prices and the naive baseline perform similarly, with negative R^2 values close to zero, indicating poor explanatory power. The Random Forest model offers intermediate performance, outperforming both the naive and the LSTM-on-close models, achieving an R^2 of 0.749.

We further assess the economic value of the model predictions through backtesting a long-short (L-S) trading strategy based on the LSTM and Random Forest outputs. As shown in Table 4, the L-S strategy using Random Forest predictions achieves the highest Sharpe ratio of 1.397, compared to 1.190 for the L-S strategy based on LSTM predictions. The Random Forest L-S strategy also achieves a smaller maximum drawdown of -1.57% versus -2.65% for the LSTM L-S strategy, suggesting better downside protection. However, the LSTM L-S strategy produces a higher annualized return of 7.60%, compared to 4.08% for the Random Forest-based strategy. Both model-driven L-S strategies outperform simple mean reversion (MR) and momentum (Mom) strategies, which exhibit lower Sharpe ratios and much higher annualized volatilities. Max drawdown and annualized return values for MR and Mom strategies are omitted due to their poor and volatile performance.

Overall, these results highlight that predictive accuracy (as measured by regression metrics) does not always align perfectly with trading performance. While the LSTM on smoothed data is the most accurate predictor, the Random Forest model produces more stable trading outcomes, reflected in a higher Sharpe ratio and lower drawdown.

7.3. *Random Forest and Comparative Performance*

The Random Forest (RF) model demonstrated strong predictive performance compared to other approaches. It achieved an R^2 value of 0.749, along with an RMSE of 0.428 and an MAE of 0.321 (Table 3). While the LSTM model trained on smoothed data achieved even higher predictive accuracy ($R^2 = 0.928$), the RF model outperformed the LSTM trained on raw close prices and the naive baseline, both of which exhibited near-zero or negative R^2 values. These results suggest that, for daily ETF return forecasting, capturing structured feature relationships through nonlinear modeling may be more effective than leveraging sequential temporal patterns at the daily frequency analyzed.

In terms of trading performance, the RF-based long-short strategy achieved the highest Sharpe ratio of 1.397 (Table 4), indicating superior risk-adjusted returns. Although its annualized return (4.08%) was lower than that of

the LSTM-based strategy (7.60%), the RF strategy exhibited a smaller maximum drawdown of −1.57% compared to −2.65% for the LSTM-based strategy, as well as a lower annualized volatility of 0.018 compared to 0.069. This suggests that the RF model provided more stable returns, balancing profitability with lower downside risk. Overall, the Random Forest model highlights the advantage of combining strong predictive precision with robust, consistent trading outcomes.

7.4. *Baselines and Strategic Implications*

We further compare our learning-based strategies against traditional momentum (Mom), mean-reversion (MR), and a hybrid mean-reversion plus momentum (MR+Mom) strategy. As shown in Table 4, these baseline strategies performed poorly. The momentum strategy, in particular, produced a negative Sharpe ratio of −0.48, indicating unfavorable risk-adjusted returns. The MR+Mom hybrid offered only marginal improvement, suggesting that simple, unsupervised heuristics are insufficient for capturing the dynamics of daily ETF returns. These results reinforce the value of supervised learning approaches that can adapt to more complex, nonlinear patterns in the data.

	L-S (LSTM)	L-S (RF)	MR	Mom	MR+Mom
Sharpe Ratio	1.190	1.397	0.24	-0.44	0.14
Annualized Volatility	0.069	0.018	5.01	4.78	4.11
Max Drawdown	-2.65%	-1.57%	-60.15%	–70.55%	-90.15%
Annualized Return	7.60%	4.08%	-2.59%	-4.34%	-3.15%

Table 4. Comparison of performances of trading strategies based on different models. Note: L-S (model name) is the long-short strategy based on the predictability of the model; MR is the mean-reversion strategy (rolling in 5 days), and MOM is the momentum strategies (rolling in 232 days); MR + MOM is the combined portfolio (with weight of 0.8 on Mean-reversion and weight of 0.2 on Momentum strategy) with some weights in both strategies

Figure 9 illustrates that these baseline strategies exhibit higher volatility and more abrupt drawdowns than model-based approaches. These findings reinforce the value of machine learning models, even when they face predictive challenges at the point-estimate level.

7.5. *Visual Comparison of Strategy Returns*

Figures 7, 8, and 9 visualize cumulative returns across strategies. RF yields the smoothest growth curve, while LSTM captures a moderate uptrend with relatively contained risk. In contrast, the baseline strategies show larger swings and inconsistent performance. Taken together, these results suggest that learning-based methods offer promising tools for ETF strategy development in noisy market environments.

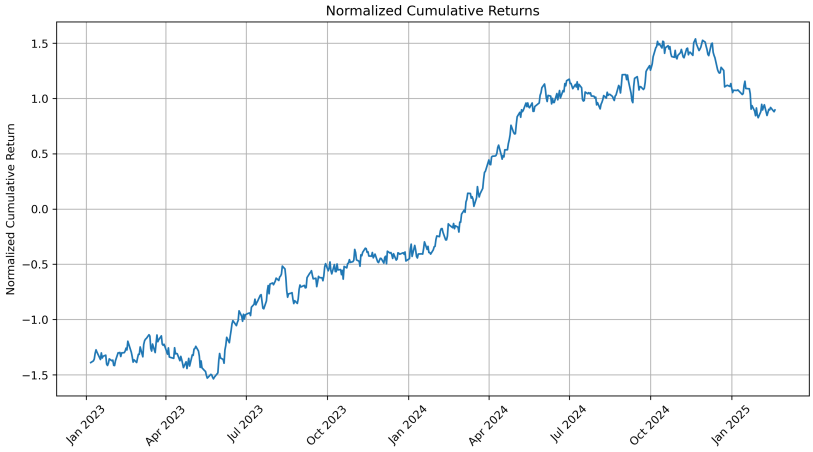

Figure 7. Normalized Cumulative Returns of the Random Forest Model from 2023/01/01 to 2025/02/19

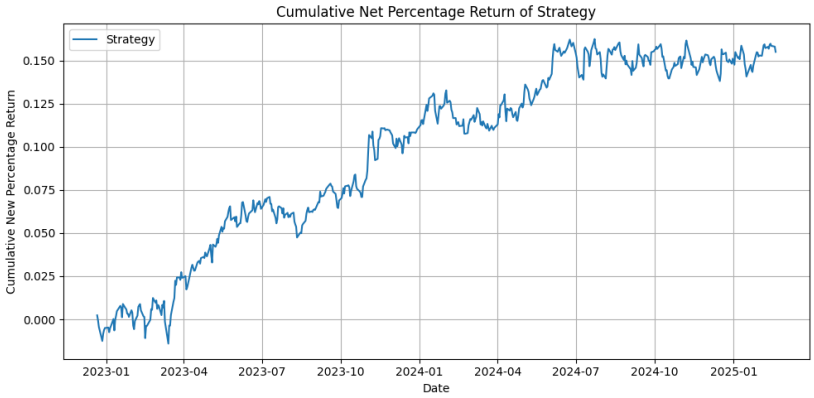

Figure 8. Cumulative return of Long-short strategies based on predictions of LSTM model from 2023/01/01 to 2025/02/19

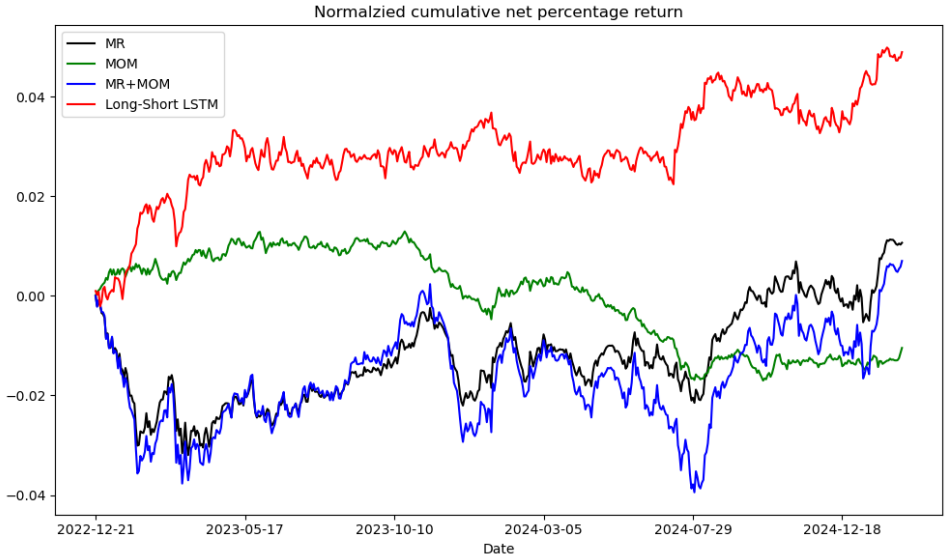

Figure 9. Normalized Cumulative return under 15% annualized volatility value, of the market-neutral trading strategies: momentum strategy (MOM), mean-reversion strategy (MR), the combination of both strategies (with 0.8 on mean-reversion, 0.2 on momentum) and the long-short strategy under the predictions of the LSTM model over the back-test period 2022/12/21 to 2025/02/19

Figure 7, 8 and 9 are the cumulative returns of all the trading strategies. From the figures, we can see that our long-short trading strategies capture the directions of the market perfectly without losing even a little money throughout the COVID periods until 2024. In 9, we choose the annualized volatility value to be 15% because we just want to compare our strategies in the real market, the number 15 itself does not contain any specific meaning.

8. CONCLUSION AND FURTHER RESULTS

This study explored the application of machine learning models, specifically LSTM networks and Random Forest regressors, to predict ETF returns and construct market-neutral trading strategies. While both models delivered insights, the Random Forest approach demonstrated superior predictive accuracy and a higher Sharpe ratio, highlighting its strength in capturing non-sequential interactions among technical features. In contrast, the LSTM model, though less precise in point forecasting, still generated viable ranking signals that supported a profitable trading strategy, with a Sharpe ratio of 1.19.

These results underscore an important insight: in noisy financial environments, models do not need to be precise in their predictions to be useful in practice. Even weak forecast signals, when correctly ranked, can yield excess returns.

Moreover, both learning-based strategies outperformed traditional baselines (mean-reversion, momentum), suggesting that supervised models better adapt to evolving market dynamics.

Several directions remain open for future work. First, expanding the model suite to include gradient boosting machines (e.g., XGBoost) or temporal convolutional networks could further clarify the tradeoff between sequential learning and nonparametric regression. Second, incorporating macroeconomic variables or sentiment analysis from financial news may improve signal quality. Third, refining position sizing through reinforcement learning or Bayesian optimization could enhance trading performance beyond uniform allocation. Finally, real-world backtesting with transaction costs, slippage, and liquidity constraints will be critical for assessing the deployability of these strategies.

In sum, this paper shows that structured machine learning frameworks can inform robust and scalable ETF trading strategies. With further refinement and integration of alternative data sources, these tools hold promise for enhancing systematic portfolio construction.

9. AI STATEMENT

Artificial Intelligence (AI) tools were utilized solely in 'Canvas mode' for drafting and editing purposes in our Introduction, LSTM, Random Forest, and Conclusion sections. No AI-generated content was used outside of this collaborative document environment.

10. CODE

Due to the volume of the coding part, we cannot present our work in the paper; please refer to the link: https://github.com/RomeoisFushengLuo/ETF_prediction_Portfolio_construction to view our codes.

REFERENCES

Ben-David, I., Franzoni, F., & Moussawi, R. 2018, The Journal of Finance, 73, 2471

Blitz, D., & Vidojevic, M. 2021, The Journal of Alternative Investments, 23, 82

Gastineau, G. L. 2001, The Journal of Portfolio Management, 27, 88

Hamm, S. J. W. 2014, The effect of ETFs on stock liquidity, Tech. Rep. 2014-108, Board of Governors of the Federal Reserve System

Hill, J. M., Nadig, D., & Hougan, M. 2015, A comprehensive guide to exchange-traded funds (ETFs) (CFA Institute Research Foundation)

Israeli, D., Lee, C. M. C., & Sridharan, S. A. 2017, Review of Accounting Studies, 22, 1048

Johnson, W. 2009, Journal of Financial Planning, 22, 42

Madhavan, A. 2016, Exchange-Traded Funds and the New Dynamics of Investing (Oxford University Press)

Madhavan, A., & Sobczyk, A. 2016, Journal of Investment Management, 14, 1

Nelson, D. M., Pereira, A. C., & De Oliveira, R. A. 2017a, in 2017 International joint conference on neural networks (IJCNN), Ieee, 1419–1426

Nelson, D. M. Q., Pereira, A. C. M., & Oliveira, R. A. d. 2017b, in Proceedings of the International Joint Conference on Neural Networks (IJCNN), 1419–1426

Pan, K., & Zeng, Y. 2019, Journal of Financial Economics, 133, 72

Paudel, K., & Naka, A. 2023, Journal of Asset Management, 24, 474, doi: 10.1057/s41260-023-00321-4

Poterba, J. M., & Shoven, J. B. 2002, American Economic Review, 92, 422

Schoenfeld, S. A. 2004, Active Index Investing: Maximizing Portfolio Performance and Minimizing Risk through Global Index Strategies (John Wiley & Sons)

Singh, E., & Yadav, A. P. 2024, Srusti Management Review, XVII, 242