Quantitative Trading Project with LSTM Model

Chuanyue Nie Kaile Chen Yize Liu

Abstract

We use LSTM networks to forecast the value of the S&P 500 index, using data from 2010 to the end of 2023, with daily frequency. Based on the forecasts from the LSTM model we generate buy and sell investment signals solely. We verified the model by testing various of LSTM model combinations and to optimize using in-sample examples and check if there is a problem of overfitting. Afterward, we employ them in algorithmic investment strategies and test if the LSTM model can facilitate investment with suitable parameters.

1. Introduction

We mainly want to study whether LSTM-type recurrent neural networks can give trading signals that allow us to earn excess returns over the market benchmark in the real stock market. We will use the LSTM-based RNN model to predict the daily closing price of the stock. If the predicted closing price is higher than the previous day's closing price, we will make a buy decision. If the closing price is lower than the previous day's predicted closing price, we will make a sell decision.

The main advantage and the novelty of our work is that LSTM as a time series model can be used to predict prices. If we convert these prices into binary signals, we are curious if such signals can still be used as an efficient way to trade the model and make excess returns. Therefore, in addition to maximizing forecasting accuracy and minimizing overfitting, we need to backtest our models to get the real market returns, Sharpe ratios, etc. We would like to explore the following questions.

1. Can the LSTM model accurately generate the correct signals and closely fit the historical stock prices?
2. Can the model, through its trading signals, outperform the benchmark return and exhibit lower volatility over the past few years?

Our project is mainly based on Micha´nków, J.; Sakowski, P.; ´Slepaczuk, R. LSTM in Algorithmic Investment Strategies on BTC and S&P500 Index. Sensors (2022).

The difference in our model is that this paper will be looking at shorter time intervals, like 15 minutes, or 30 minutes, and will be comparing different markets side by side, like the Bitcoin market. We will only look at the S&P500 market and the time interval is one day. They use LSTM directly in their paper to generate trading signals, whereas we would like to use the LSTM model to do the prediction of the stock price first and then display trading signals based on the stock price.

2. Methods

2.1. LSTM Model

We choose the LSTM model as our main model to handle the data. LSTM is chosen for its ability to learn long-term dependencies in sequence data, which is vital for predicting time series where past information is predictive of future events. LSTM units are composed of memory cells, with each cell having three types of gates (input gate, output gate and forget gate). These gates use tanh and sigmoid functions to regulate the flow of information through the cell, deciding how much and which information should be stored in a long-term state, passed on to another step, or discarded. In the following graph, it shows a single cell of LSTM network:

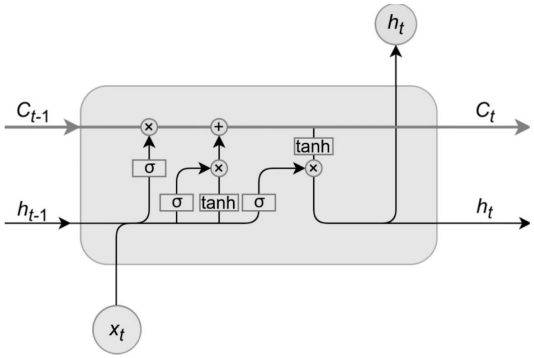

Figure 1. LSTM cell. Source: Matsumoto F., (2019)

In our research, we used the rolling window method to get the data set: the input vector for the LSTM network was a series of past observations from S&P500 data, and the output vector was a single value predicted for the next period.

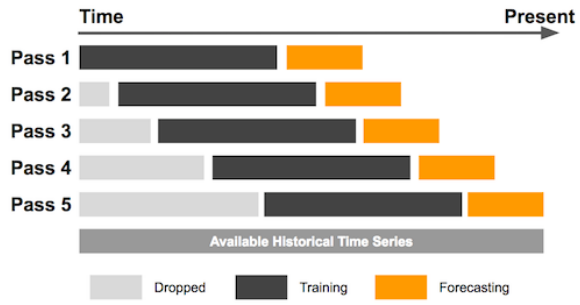

Figure 2. Rolling Window Method. Source: David Andrés, (2019)

2.2. Data Selection

Data was sourced from Yahoo Finance using the yfinance library, a popular choice due to its ease of access and reliability in fetching historical market data. The dataset included daily stock prices from January 1, 2013, to December 31, 2023, covering features like Open, High, Low, Close, and Volume. In order to improve the model's generalization ability, we have also added two more features: Return on position and Sharp ratio during holding period.

Table 1. Features and their meanings

FEATURES	MEANING
RETURN ON POSITION	THE INDEX RETURN RATE WITHIN 10 DAYS.
SHARP RATIO DURING HOLDING PERIOD	AN INDICATOR THAT MEASURES THE RELATIONSHIP BETWEEN FUND RISK AND RETURN.

The data was normalized using MinMaxScaler, ensuring all features contribute equally to the model's predictions, improving convergence during training. The data was transformed into sequences representing 20 consecutive trading days to predict the next day's price.

We used fixed parameters including batch size, number of epochs, and number of steps and we also used variable parameters including number of layers, units per layer, dropout rate, and input shape. We used cross-validation during grid search to evaluate each model configuration and the Mean Squared Error (MSE) metric to quantify the prediction accuracy on the validation set.

FIXED PARAMETERS:

- **test_ratio**: Determines the proportion of data reserved for testing (40%), to assess the model's performance on unseen data.
- **batch_size**: Specifies the number of samples processed before the model updates its weights (32 samples per batch).
- **n_epochs**: Number of complete passes through the training dataset (50 epochs) to ensure adequate training without over-fitting.
- **num_steps**: The number of time steps in each input sequence (20 steps), defining the temporal extent of data the model considers.

PARAMETERS SPACE:

- **num_layers**: The range of LSTM layers explored (2 to 4 layers) to identify optimal depth for capturing data dependencies.
- **units_per_layer**: Number of neurons per LSTM layer tested (50, 128, 256) to balance between learning capacity and computational efficiency.
- **dropout_rate**: Proportion of neurons randomly deactivated during training to prevent over-fitting, with rates from 0.1 to 0.3 explored.
- **input_shape**: Shape of the input data, set dynamically based on the training data's dimensions, crucial for matching the LSTM input requirements.

2.3. Prediction Method

Model Implementation

A sequential model is initiated, enabling us to stack layers linearly. Each LSTM layer is added sequentially with the specified number of units. The first layer is defined with an `input_shape`, and `return_sequences` is set based on the presence of subsequent LSTM layers. A dropout layer follows each LSTM layer to mitigate over-fitting by randomly deactivating a fraction of the neurons during training. Then, a Dense layer with a single unit concludes the model, designed for regression output. The model is compiled using the Adam optimizer and mean squared error loss, which are conventional choices for regression problems.

Model Training

Forecasting is based on the assumption that there are similar patterns between future data and historical data. The model, instantiated with specific configurations, is trained on historical data. It processes the data in batches and epochs, then employs the trained model to generate predictions on the test dataset. These predictions are adjusted back to their original range using the minimum and maximum values of the test data, ensuring the output is interpretable and comparable to actual values. If the predicted price is greater than the previous day's actual price, it is recorded as 1; if the predicted price is less than the previous day's actual price, it is recorded as -1. Based on this result, corresponding trading operations are performed in the backtesting stage.

2.4. Backtesting Metrics

Annual Return:

$$\text{Annual Return} = (1 + \text{Periodic Return})^{\frac{252}{d}} - 1$$

Sharpe Ratio:

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p}$$

Cumulative Return:

$$\text{Cumulative Return} = \left(\frac{\text{Final Value}}{\text{Initial Value}}\right) - 1$$

Annual Volatility:

$$\text{Annual Volatility} = \sigma_p \times \sqrt{252}$$

Max Drawdown:

$$\text{Max Drawdown} = \min\left(\frac{T - P}{P}\right)$$

where P is the peak value before the largest drop, and T is the trough value after the peak.

3. Results

3.1. Parameters

Table 2. Parameters after Tuning

PARAMETER	SELECTED VALUES
TEST_RATIO	0.4.
BATCH_SIZE	32
N_EPOCHS	50
NUM_STEPS	20
NUM_LAYERS	3
UNITS_PER_LAYER	128
DROPOUT_RATE	0.1

We used validation set and found the best parameters as above.

3.2. Traning and testing Evaluation

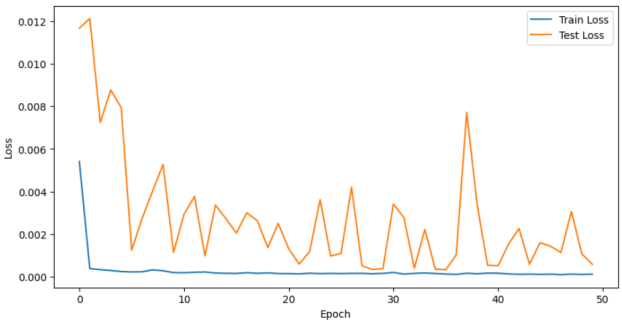

Figure 3. Trainng and Testing Loss in 50 epochs

The analysis of the model's performance based on training and test loss is as follows:

Training Loss (Blue Line): The training loss decreases rapidly from a high initial value and then follows a generally steady downward trend.

Test Loss (Orange Line): Despite significant fluctuations and noticeable spikes in loss values at specific epochs, such as between 20 and 30, and near 50, the test loss generally shows a decreasing trend. This suggests that the model possesses good generalization ability, although its performance on new data (test set) is less stable compared to the training set.

3.3. Predicted Stock VS Actual Price

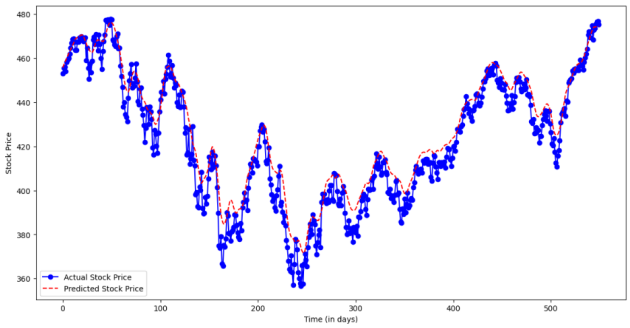

Figure 4. Predicted Stock Price and Actual Price for SP500

Overview

Actual Prices: Shown in blue, these represent the true daily closing prices of the stock and serve as our benchmark for assessing model accuracy.

Predicted Prices: Indicated by the red dashed line, these are the outputs of our LSTM model, forecasted based on the training data provided.

Key Observations

- **Trend Following:** The LSTM model demonstrates a strong capability in following the general trends observed in the actual stock prices. It captures both the rises and falls in stock value to a considerable degree.
- **Close Tracking:** There are numerous instances where the predicted prices closely align with the actual prices, suggesting that the model effectively captures the underlying patterns in stock price movements.
- **Discrepancies:** Despite the general alignment, there are evident discrepancies where the predicted prices diverge from the actual prices. These deviations are more noticeable during periods of high volatility or sharp price changes, indicating areas where the model may struggle to capture sudden market dynamics.
- **Performance Over Time:** As time progresses, the model appears to adapt and improve, which could be indicative of the accumulative learning effect of the LSTM network as it processes more data.

3.4. Back-testing Results

Table 3. BackTesting Results For LSTM Model

METRIC	VALUE
ANNUAL RETURN	62.61%
CUMULATIVE RETURN	188.95%
ANNUAL VOLATILITY	18.90%
SHARP RATIO	2.67
MAX DRAWDOWN	-9.48%
SKEW	0.16
KURTOSIS	1.25

Table 4. SP500 Performance

METRIC	VALUE
ANNUAL RETURN	4.00%
CUMULATIVE RETURN	8.91%
ANNUAL VOLATILITY	19.11%
SHARP RATIO	0.25
MAX DRAWDOWN	-24.50%
SKEW	-0.09
KURTOSIS	1.33

The LSTM model exhibited superior performance compared to the S&P 500 benchmark index across all key metrics. Notably, the LSTM achieved significantly higher annual and cumulative returns, demonstrating its ability to capture market trends effectively. The Sharpe ratio of 2.67 indicates strong risk-adjusted returns, and the low maximum drawdown highlights its resilience in market downturns. Despite slightly higher annual volatility, the model remains efficient, evidenced by positive skew and controlled kurtosis.

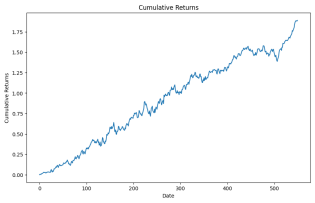

Figure 5. Cumulative return of LSTM M

The results suggest that the LSTM model is a valid tool for predicting the S&P 500 index and can provide substantial value to investors seeking to maximize returns while minimizing risk.

4. Process

4.1. Benefits of using LSTM

Since our model uses a specific number of steps moving forward, it can often achieve high accuracy in price prediction. Even if the predicted price deviates significantly from the actual price on a given day, the model can still make more accurate predictions for future days by using the actual price of the following day as the basis for its next prediction. This backward correction mechanism leverages the LSTM's ability to capture both short- and long-term dependencies in time series data. LSTM networks, with their inherent ability to remember information across long sequences, are good at capturing trends and cyclical patterns in historical data. Despite occasional errors in predictions, the continuous learning process ensures that the network quickly adapts to market changes by incorporating the latest actual prices.

4.2. Window Size Experimentation

Initially, the project aimed to optimize the window size of input sequences for the LSTM model, hypothesizing that varying the number of days in each sequence could significantly affect the predictive power and efficiency of the model. We experimented with different window sizes ranging from 10 to 50 days. While longer windows captured more extensive market dynamics, they significantly increased the model's training time and computational load. Eventually, we settled on a 20-day window, a common choice in financial modeling, balancing the model's complexity and performance without excessively long training times.

4.3. Direct signal prediction

Another significant pivot was the initial attempt to use the LSTM model to directly generate trading signals instead of predicting stock prices. The idea was to use a binary classification approach where the model would predict 'buy' or 'sell' signals directly based on the historical price sequences. However, adapting LSTM networks to effectively work with a binary loss function proved challenging. The model struggled to converge, likely due to the nuances and noise inherent in financial time series data, which are not as straightforward as typical classification.

4.4. Adjusting Hyperparameters and Architecture

In our efforts to fine-tune the LSTM's architecture to combat overfitting, we experimented with the number of layers and the number of units per layer. We introduced dropout layers and tested various regularization methods. However, despite these efforts, a notable issue was the inconsistency in model performance across different runs. Each time we tested various hyperparameters, the models exhibited different behaviors and sometimes did not converge to a stable point.

5. Contributions

5.1. Primary libraries

We utilized yfinance, a Python library, to fetch historical stock data of the SPY ETF from Yahoo Finance. The dataset spanned from January 1, 2013, to December 31, 2023, and included daily prices such as open, high, low, close, and volume.
To fine-tune our models and identify the best hyperparameters, we used GridSearchCV from sklearn.modeselection. This method was instrumental in selecting the optimal one based on cross-validated performance metrics.
We constructed LSTM-based neural networks using tensorflow and keras. Our models included multiple LSTM layers, dropout for regularization, and dense layers for output. We experimented with different architectures to find the optimal setup. And, we used the Adam optimizer as optimizer.

5.2. Other libraries

Matplotlib: We used matplotlib.pyplot for visualizing data and results, which was crucial for interpreting model performance and diagnostics.
Pandas: We used Pandas for data manipulation and cleaning, pandas were indispensable throughout the project for its powerful data handling capabilities.
NumPy: We used NumPy for numerical operations, especially in data preprocessing and transformation stages.

5.3. Attribution

Our project's methodology was informed by the paper by Michańków, J.; Sakowski, P.; Ślepaczuk, R. titled "LSTM in Algorithmic Investment Strategies on BTC and SP500 Index" published in Sensors 2022. While the inspiration for the application of LSTM models to financial time series came from this paper, all code implemented in the project was entirely original, crafted by our team to address the specific nuances of our dataset and project objectives.

The code for our project is available on GitHub at this repository.

References

6. Reference

1.Matsumoto, F. Redes Neurais—LSTM. 2019. Available online: https://medium.com/turing-talks/ (accessed on 10 January 2022).
2.Forecasting methods in Time Series Published by David Andrés on January 19, 2023 https://mlpills.dev/time-series/forecasting-in-time-series/
3.LSTM in Algorithmic Investment Strategies on BTC and S&P500 Index by Micha nk ow, J.; Sakowski, P.; Slepaczuk, R.