Unveiling the Nexus Between ML Model Complexity and U.S. Securities Strategy Efficacy

Darien Nouri DAN9232@NYU.EDU
Charles Wang MW4899@NYU.EDU
Yihao Zhong YZ7654@NYU.EDU

1. Motivation

The topic under exploration is the comparative analysis of different predictive modeling techniques, including Convolutional Neural Networks (CNN), Long Short-Term Memory networks (LSTM), Autoregressive Integrated Moving Average (ARIMA), and a naive Ordinary Least Squares (OLS) model, in generating trading signals for U.S. equities. The primary aim is to investigate the relationship between model complexity and the performance of stock/portfolio risk-return strategies. This involves examining how each model's predictive accuracy translates into actionable trading strategies and whether more complex models necessarily offer investment performance when considering risk-adjusted returns.

In the quest for superior investment returns, the finance industry continually explores advanced modeling techniques. This study contributes to understanding how different deep-learning and time-series-based machine-learning techniques in predictive modeling can be leveraged for financial gains. By examining the risk-return profiles associated with each model, this research aids in better risk management and portfolio optimization, a crucial topic for both individual buyers and institutional investors (Jansen, 2020).

This work is unique in its comprehensive comparison of models ranging from simple to complex within a consistent framework. Our research provides a more nuanced understanding of the trade-offs involved in employing complex predictive models for stock market investments. It not only assesses predictive accuracy but also directly links this accuracy to practical investment strategy outcomes. This holistic approach, focusing on the end-to-end process of model prediction to investment performance, will also provide a near real-time strategy performance in an equity market trading day for up-to-date forward-testing research outcomes and validation.

2. Related Works

In the *Deep learning for stock market prediction from financial news articles*, it explores the use of deep learning models, including CNNs, to predict stock market movements based on financial news articles . It demonstrates the potential of CNNs to extract relevant features from textual data for market prediction (Sezer and Ozbayoglu, 2020). Similarly in Hu et al. (2018), by processing vast amounts of textual data, they uncovers underlying patterns and sentiments that may not be immediately apparent, providing valuable insights into market movements. This approach leverages complex neural network architectures, such as Convolutional Neural Networks (CNNs) for text classification and Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks for capturing temporal

dependencies in sequential data. The goal and end result is to predict stock market trends or price movements by quantifying the impact of news sentiment on market behavior. Their finding showed a significant improvement in predicting stock market trends, outperforming baseline models that rely solely on quantitative market data.

In the *A Comparative Study of SVM and LSTM Deep Learning Algorithms for Stock Market Prediction*, the author employed Support Vector Machine and Long Short-Term Memory (LSTM) networks to predict the stock market (Sai Krishna Lakshminarayanan, 2019). The model is trained by Dow Jones Index (DJI) stock price data combining the oil and gold prices. In addition, they also applied a moving average to assess the effect of models. They measure the performance using RMSE, MSE, MAE, MAPE and R-squared. The study found out that LSTM advanced model with moving averages has the best performance for stock price prediction outperforming all other SVM models and LSTM model without moving averages. This research addressed the problem of volatility and noise in the data by using a moving average. However, the study did not state how they treat the problem of stationarity in time series.

3. Methodology

Pre-processing and Feature Selection/Engineering:

We will undertake comprehensive time series analysis, including observation of the Autocorrelation Function (ACF) and performing the Augmented Dickey-Fuller (ADF) testing, to prepare our data. Additionally, technical indicators like moving averages, z-scores, and lagged values will be incorporated to enrich our models' input.

In our research, we delve into the utilization of Convolutional Neural Networks (CNNs), a class of deep learning models that have revolutionized the field of machine learning by setting new standards in tasks involving image classification, video analysis, and speech recognition. Unlike traditional feed-forward networks that rely on general matrix multiplication, CNNs employ a specialized operation known as convolution. This operation is particularly adept at processing data with inherent spatial or temporal structures. Leveraging the grid-like structure inherent in time-series data, CNN architectures can be adeptly applied to both univariate and multivariate time series analysis in our research.

Long Short-Term Memory (LSTM) networks, a specialized form of Recurrent Neural Networks (RNNs), incorporate complex units designed to capture long-term dependencies in sequence data. Unlike standard RNNs, LSTMs are equipped with an internal state and multiple gates—namely, input, output, and forget gates—that regulate information flow, thereby effectively addressing the challenges of vanishing and exploding gradients. This architecture allows LSTMs to maintain and modify their state over long sequences, making them adept at tasks requiring the understanding of long-term temporal relationships (Jansen, 2020).

In our research, we employ LSTMs due to their ability to model time-series data, where understanding the long-term context and dependencies is crucial. Their sophisticated gating mechanism enables the preservation and selective retrieval of relevant information across long data sequences, enhancing our models' predictive accuracy and robustness.

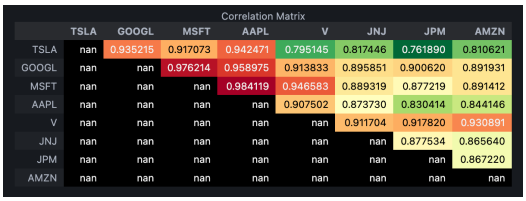
The Autoregressive Integrated Moving Average (ARIMA) model, which combines autoregressive (AR) and moving average (MA) components, offers a streamlined approach to

time series modeling. This integration simplifies the model structure, reduces the number of required parameters, and helps mitigate the risk of overfitting, enhancing model efficiency, as indicated by its Autocorrelation Function (ACF). Worth mentioning, in practical applications, such as forecasting macroeconomic fundamentals, the Seasonal ARIMA (SARIMAX) model extends ARIMA's capabilities to include seasonal effects.

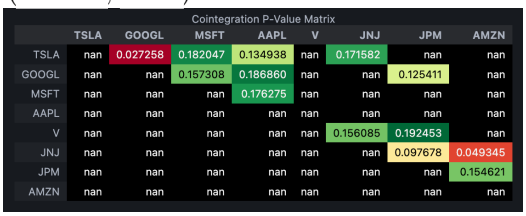
Evaluation

A custom-built back tester lib written in Python will serve as the evaluation platform, featuring classes for portfolio management, data preprocessing, and model-specific operations. This setup ensures a consistent evaluation environment. Adding how to evaluate the CNN, and LSTM. What metrics we need to look at on the models and stock price.

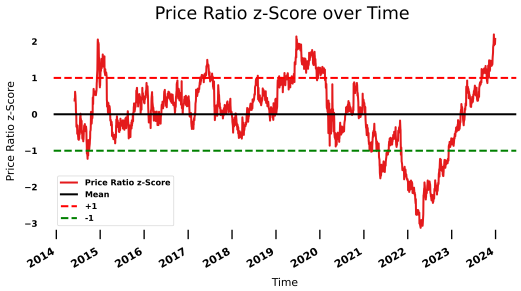
We also conduct a cointegration test. Cointegration is a statistical property observed in certain sets of time series data. It suggests that the time series will tend to revert to a common trend, offering a metric for us to generate a trading signaling strategy. When the residuals deviate significantly from the mean, it suggests that the short-term price movements have strayed from the historical relationship, potentially presenting an opportunity to go long on an undervalued asset or short on an overvalued one, with the expectation that the prices will eventually revert back towards their equilibrium. In essence, cointegration helps us identify stable and mean-reverting assets. (Jansen, 2020).



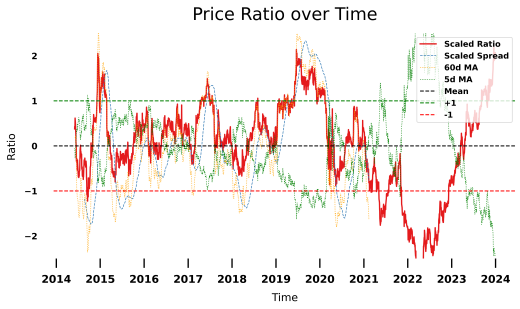
Correlation Matrix



Cointegration Matrix



Price Ratio Zscored



Price Ratio With Rolling Mean

References

Zhongsheng Hu, Weiheng Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 261–269, 2018. Accessed: 2024-03-03.

Stefan Jansen. *Machine Learning for Algorithmic Trading: Predictive models to extract signals from market and alternative data for systematic trading strategies with Python.* Packt Publishing, 2 edition, 2020. ISBN 978-1839217715.

John McCrae Sai Krishna Lakshminarayanan. A comparative study of svm and lstm deep learning algorithms for stock market prediction. *AICS*, 2019. doi: https://ceur-ws.org/Vol-2563/aics_41.pdf. Accessed: 2024-03-03.

Omer Berat Sezer and Murat Ozbayoglu. Financial time series forecasting with deep learning : A systematic literature review: 2005-2019. *Applied Soft Computing*, 90:106181, 2020. doi: 10.1016/j.asoc.2020.106181. Accessed: 2024-03-03.

Unveiling the Nexus Between ML Model Complexity and U.S. Securities Strategy Efficacy (Report 2)

Darien Nouri DAN9232@NYU.EDU

Charles Wang MW4899@NYU.EDU

Yihao Zhong YZ7654@NYU.EDU

1. Methodology

1.1 Pair Selection

In our project, we followed a systematic approach to select pairs of stocks for trading strategies. Firstly, we conducted cointegration and correlation analyses to identify suitable pairs. Cointegration analysis helps in determining if two non-stationary time series, represented by X_t and Y_t, have a stable long-term relationship by finding a stationary linear combination of them.

After calculating cointegration and correlation, we prioritized pairs based on their cointegration strength. This step involved sorting the pairs by their cointegration values to focus on those with stronger long-term relationships.

Furthermore, we applied a filter to exclude pairs that exhibited high correlation coefficients exceeding 0.8. This filtering process is crucial because highly correlated pairs offer limited profit potential since there are minimal deviations between the stock prices, reducing the opportunity to capitalize on price divergences.

1.2 Sentiment Analysis on Alternative Data Source

Analyzing financial news articles and headlines through natural language processing techniques can offer a more comprehensive understanding of market movements, addressing the limitations of traditional data sources like historical price data and financial statements. We explore using sentiment analysis of real-time news headlines as a potentially faster and more direct indicator of market sentiment shifts and nascent volatility, compared to the delayed impact on stock prices themselves. Our hypothesis is that analyzing sentiment differences between mean-reverting stock pairs could provide quicker insights into their relationship and serve as an early indicator of market volatility. We introduce 'Sentiment Divergence,' which calculates the variance in sentiment between stocks within a pair. By focusing on the relative differences in both sentiment and stock prices, we maintain consistency and compare the variables in a normalized space, enabling a more meaningful examination of their relationship.

Our approach involves leveraging web scraping techniques to gather daily financial news headlines from reputable sources such as Nexis, Business Research & News Media database. Over the research period spanning from January 1, 2021, to March 31, 2024, we have amassed a substantial dataset containing around 10 headlines per day for each ticker.

We utilized FinBERT (Huang and Yang, 2022), a financial pre-trained BERT model from Huggingface, to perform sentiment analysis on financial news headlines, yielding more accurate sentiment detection compared to standard BERT. Sentiment labels (positive, neutral, negative) are assigned numeric values (3, 2, 1) and the most probable label is chosen as the headline's sentiment. A news title's sentiment score is the product of its probability and numeric sentiment value. Daily sentiment scores can be noisy, so we apply moving averages to better understand the underlying trend. Additionally, we focus on market sentiment rather than individual opinions.

Indexes are computed as the mean of all sentiment scores for that day, and 7, 14, and 31-day moving averages are applied to identify longer-term sentiment trends.

$$SI = \frac{1}{|H|} \sum_{h \in H} P(h) \times V(h) \tag{1}$$

$$MA_{t,n} = \frac{1}{n} \sum_{i=t-n+1}^{t} SI_i \tag{2}$$

Where:

- $P(h)$ is the probability of the sentiment for headline h.
- $V(h)$ is the numeric value of the sentiment (3 for positive, 2 for neutral, 1 for negative).
- SI_i is the sentiment index for day i.
- $MA_{t,n}$ is the moving average at time t over n days (7, 14, or 31).

1.3 Feature Engineering

Firstly, Our team uses the ta package from python to implement feature engineering. Firstly our project uses daily stock data from Yahoo Finance. Date', 'Open', 'Low', 'Close', 'Adj Close' and 'Volume' are our baseline features of interest. Now, for the purposes of a pairs strategy, raw price data alone is insufficient. We need to derive insights into the relationship between the selected pairs. Therefore, we calculate several technical indicators to enhance our analysis.

We generate several indicators that could reflect the performance of a stock and add these indicators as new features of the input data.

A moving average acts as a dynamic threshold, above which a stock might be considered overvalued, and below which it could be deemed undervalued. By analyzing moving averages of stock pairs, we can identify potential entry and exit points for trades based on the relative positioning of the stocks' prices to their moving averages.

Secondly, we calculate the difference between the pairs of stock prices and use the spread as a new feature of our data set. We focus on calculating the spread for each of close, open, high, low specifically. Due to the fact that stock prices are unpredictable, our team decided to turn to predict the spread between the two stocks, based on the pair trading strategy (Victor Chang1 and Hsu, 2020). The change of spread over time will be used to determine a buy or sell.

Thirdly, our team used the moving average when using the spread as a feature. Moving average could make it easier to spot the direction of the trend of a stock in the long term and smooth market fluctuations by partial out the short-term variances and noises providing a general overview of the stock price movement.

1.4 OLS, ARIMA, LSTM

Our team uses the OLS, ARIMA, Long Short-Term memory networks to predict the spread between the stock prices in order to capturing long-term dependencies and patterns in historical prices movements.

1.4.1 Spread

The 'spread' in pair trading normally refers to the difference in price between the two stocks, and trading decisions are made based on the spread's deviation from its historical average. Compared to the vanilla price difference used in milestone 1, we update a new spread calculation. We calculate the spread in a stock pair trading context by first conducting a regression analysis between the prices of two stocks. Essentially, it computes the linear relationship between the

prices of stock 1 (s1) and stock 2 (s2) over a specified rolling window. By adjusting for β, the spread calculation isolates the idiosyncratic component of stock 2's price movements that cannot be explained by stock 1's price movements.

The regression is $s_2 = \beta_0 + \beta_1 \times s_1 + \epsilon$ where β_0 is the intercept of the regression line, β_1 is the slope of the regression line, indicating how much s_2 changes for a one-unit change in s_1, ϵ represents the error term.

$$\text{Spread} = s2 - (\beta_1 \times s1)$$

For a rolling window approach, this calculation is applied within each window, updating β_1 for each segment of the data defined by the window size.

The z-score of the spread is then calculated to standardize the spread across the entire dataset:Z-Score $= \frac{\text{Spread} - \mu_{\text{Spread}}}{\sigma_{\text{Spread}}}$

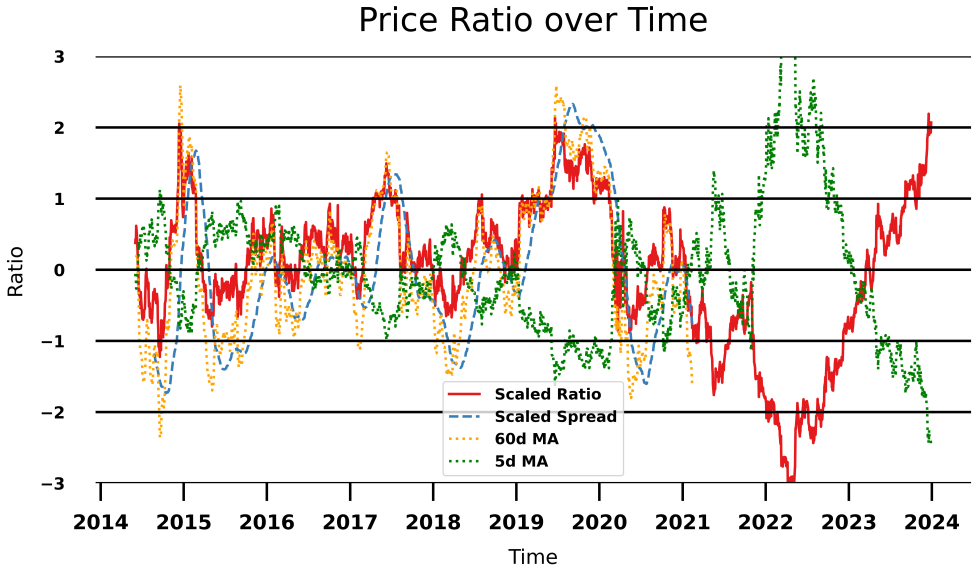


Figure 1: Scaled Z-Score Spread

1.4.2 LSTM

LSTM is a type of RNN model that focuses from sequence and time of the events. For milestone 1, our focus is on dataset processing and features refinement, the LSTM model is a naive model without any fine-tuning on hyperparameters. Our model for this milestone is considered "vanilla" because it uses a straightforward implementation of LSTM layers without advanced or complex modifications. It includes two LSTM layers with a high dropout rate to prevent overfitting, followed by a Flatten layer to reshape the data, Dense layers for prediction, and additional Dropout layers to further combat overfitting. The use of tanh activation functions is typical for LSTMs, as it helps manage the gradients during the learning process. The structure of the LSTM model is shows in Figure 1.

2. Results

2.1 Feature Engineering

Our team generate several features displayed in Table 1 2 .

2.2 Alternative Data Source

Our team focused on getting text data which is news headlines. We collect the financial news headlines from Nexis Newsdesk (Nexis newsdesk, 2024) from 2021-01-01 to 2024-03-31 for these companies (listed by tickers): APPL, AXP, BAC, BLK, C, CRM, FDX, GE, GOOGL, GS, HON, INTC, JPM, MS, UNP, WFC. For each companies our team was able to collect around 5k - 6k news headlines.

We first use Linear Regression on our data. The model yields a mean absolute percentage error of 7.99 %. And the model suggests that Daily Log Return, Exponential Moving Average and close price are the three top features which are important to the model.

Then we use the ARIMA model to train and test on our data. The MAPE (Mean Absolute Percentage Error) for 1 step ahead forecasting is 0.1354, 5 step ahead forecasting is 0.1037, 15 step ahead forecasting is 0.2363. And the test MAPE for the model is 0.3753.

Finaly, we implement LSTM model, The MAPE for 1 step ahead forecasting is 0.37, 5 step ahead forecasting is 0.39, 15 step ahead forecasting is 0.46. And the test MAPE for the model is 0.3753.

3. Analysis

3.1 Sentiment Analysis

The sentiment index for Goldman Sachs shows daily spikes tied to specific news events, which didn't significantly alter the long-term sentiment trend, as evidenced by the 7, 14, and 31-day moving averages. This pattern aligns with Goldman Sachs's prominence in the financial sector, where shifts in public sentiment are common but show some long-term stability. For Honeywell International Inc., despite occasional spikes in daily sentiment, the 14 and 31-day moving averages reveal a stable public sentiment over time, reflecting the company's lower profile in public discourse. The FinBERT model, which assesses sentiment based on the presence of keywords indicating positive or negative outlooks. Key words like "up", "higher" as positive, and assign negative label to titles with key words like "down", "lower".

3.2 Model Performance

According to the results, the performance of the naive linear regression model performs the best, followed by ARIMA model and LSTM model. But intuitively, LSTM model should have the best performances. It may because of the problem of train data setup and too many noisy features included. Lower MAPE values indicate better predictive accuracy. For ARIMA model, the MAPE values increase with the forecasting horizon—from 0.1354 for 1 step ahead to 0.2363 for 15 steps ahead—which is expected because predicting further into the future typically becomes less accurate as uncertainty increases. However, the MAPE jumps significantly in test set to 0.3753, suggesting that while the ARIMA model might perform reasonably well in the short term, its long-term (or out-of-sample) forecasting ability is considerably weaker.

For the LSTM model, the MAPE values are relatively high even at 1 step ahead (0.37) and increase as the forecasting horizon extends, with the test MAPE matching the ARIMA model at 0.3753. This indicates that the LSTM model does not outperform the simpler ARIMA model in this case.

LSTMs are powerful for capturing complex patterns in time series data. However, if the data is too noisy or the signal-to-noise ratio is low, the model might struggle to learn meaningful patterns, leading to poor performance. LSTMs can benefit significantly from well-prepared and

engineered features. Currently, we did not use engineered features to fit into LSTM because we want naive LSTM with no-engineered dataset as a benchmark.

4. Plans for Additional analysis

4.1 Adding more alternative data

we consider adding more alternative data like unemployement rate, Done & Jones Industry index to improve the generality and accuracy of our LSTM model. For now, our models are only trained on stock price related data. In the next phase, we will add sentiment score into the model to improve the accuracy of the model. After adding more features from other alternative data source, we consider implementing dimension reduction to mitigate the problem of overfitting and avoid the curse of dimensionlaity.

4.2 LSTM Model Improvement

In the next milestone we will consider ensemble methods: combining the predictions from multiple models (e.g., different configurations of LSTM or incorporating ARIMA, GRU and BiLSTM models) that can improve accuracy and robustness. Also, adjusting the model's hyperparameters, such as the number of LSTM units, dropout rates, learning rate, and the number of layers. Systematic experimentation or techniques like grid search and Bayesian optimization can be used to find the optimal configuration. Besides, integrating attention mechanisms can allow the model to focus on specific parts of the input sequence that are most relevant for the prediction, which can enhance performance, especially on more complex sequences. Finally, we will exploring more sophisticated LSTM variants, such as Peephole LSTM or Convolutional LSTM (ConvLSTM), which are designed for specific types of data and problems, might offer performance benefits. We will also use the 60 days moving average as our new input data as the current data does not have a good performance.

Table 1: Ranked Pairs by Correlation and Co-integration

Aggregate Rank	Ticker 0	Ticker 1	Correlation	Co-integration p-value
0	GS	BLK	0.8344	0.0160
1	JPM	CRM	0.8390	0.0265
2	INTC	C	0.8583	0.1598
3	WFC	UNP	0.7002	0.0293
4	HON	GS	0.6692	0.0188

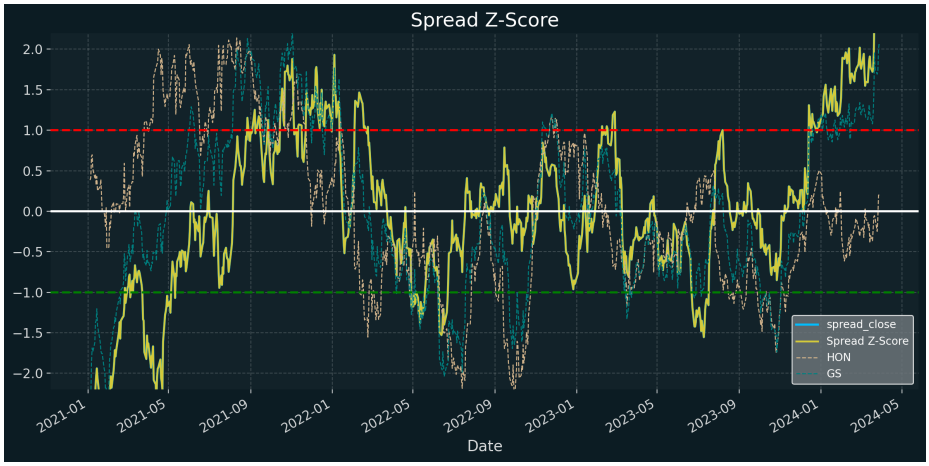


Figure 2: Spread between HON and GS

Table 2: Extracted and Engineered Features for Pairs Trading Strategy
max width=

Feature Name	**Description**
S1_rsi	Relative Strength Index for stock S1 with window=14
S2_rsi	Relative Strength Index for stock S2 with window=14
S1_adi	Accumulation/Distribution Index for stock S1
S2_adi	Accumulation/Distribution Index for stock S2
S1_vpt	Volume-price trend for stock S1
S2_vpt	Volume-price trend for stock S2
S1_atr	Average True Range for stock S1 with window=14
S2_atr	Average True Range for stock S2 with window=14
S1_bb_ma	Bollinger Bands Moving Average for stock S1 with window=20
S2_bb_ma	Bollinger Bands Moving Average for stock S2 with window=20
S1_adx	Average Directional Movement Index for stock S1 with window=14
S2_adx	Average Directional Movement Index for stock S2 with window=14
S1_ema	Exponential Moving Average for stock S1 with window=14
S2_ema	Exponential Moving Average for stock S2 with window=14
S1_macd	Moving Average Convergence Divergence for stock S1
S2_macd	Moving Average Convergence Divergence for stock S2
S1_dlr	Daily Log Return for stock S1
S2_dlr	Daily Log Return for stock S2

References

Improved pairs trading strategy using two-level reinforcement learning framework. *Engineering Applications of Artificial Intelligence*, 126:107148, 2023. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2023.107148. URL `https://www.sciencedirect.com/science/article/pii/S0952197623013325`.

Zhongsheng Hu, Weiheng Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 261–269, 2018. Accessed: 2024-03-03.

Hui Wang Huang, Allen H. and Yi Yang. Finbert: A large language model for extracting information from financial text. https://onlinelibrary.wiley.com/doi/full/10.1111/1911-3846.12832, 2022. Accessed: 2024-04-06.

Stefan Jansen. *Machine Learning for Algorithmic Trading: Predictive models to extract signals from market and alternative data for systematic trading strategies with Python.* Packt Publishing, 2 edition, 2020. ISBN 978-1839217715. URL https://www.packtpub.com/product/machine-learning-for-algorithmic-trading-second-edition/9781839217715.

Nexis newsdesk. Nexis Newsdesk Media monitoring and analytics. https://www.lexisnexis.com/en-us/professional/media-monitoring/newsdesk.page, 2024. Accessed: 2024/04/05.

A. Rostamian and J.G. O'Hara. Event prediction within directional change framework using a cnn-lstm model. *Neural Computing and Applications*, 34:17193–17205, 2022. doi: 10.1007/s00521-022-07687-3. URL https://link.springer.com/article/10.1007/s00521-022-07687-3#citeas. Accessed: 2024-03-03.

John McCrae Sai Krishna Lakshminarayanan. A comparative study of svm and lstm deep learning algorithms for stock market prediction. *AICS*, 2019. doi: https://ceur-ws.org/Vol-2563/aics_41.pdf. Accessed: 2024-03-03.

SEC EDGAR. SEC EDGAR Database. https://www.sec.gov/files/dera/data/financial-statement-and-notes-data-sets/, 2024.

Jaimin Shah, Darsh Vaidya, and Manan Shah. A comprehensive review on multiple hybrid deep learning approaches for stock prediction. *Intelligent Systems with Applications*, 16:200111, 2022. ISSN 2667-3053. doi: https://doi.org/10.1016/j.iswa.2022.200111. URL https://www.sciencedirect.com/science/article/pii/S2667305322000497.

H. G. Shin, I. Ra, and Y. H. Choi. A deep multimodal reinforcement learning system combined with cnn and lstm for stock trading. pages 7–11, 2019. doi: 10.1109/ICTC46691.2019.8939991. Accessed: 2024-03-03.

Qianwen Xu1 Victor Chang1, Xiaowen Man and Robert Hsu. Pairs trading on different portfolios based on machine learning. 9, 2020. doi: doi.org/10.1111/exsy.12649. Accessed: 2024-04-06.

Yahoo! Finance. yfinance: Yahoo! Finance market data downloader. https://pypi.org/project/yfinance/, 2024. Accessed: 2024/02/29.

Unveiling the Nexus Between ML Model Complexity and U.S. Securities Strategy Efficacy (Report 3)

Charles Wang MW4899@NYU.EDU
Darien Nouri DAN9232@NYU.EDU
Yihao Zhong YZ7654@NYU.EDU

1. Final Methodology and Results

Our research aimed to investigate the relationship between machine learning model complexity and the efficacy of U.S. securities trading strategies. We employed a comprehensive approach, incorporating advanced feature engineering, refined pair selection, and a diverse range of predictive models.

1.1 Stock Pair Selection

While optimal pair selection was not the primary objective of our research, we developed a systematic approach to identify stock pairs from the S&P 500 universe. The purpose of this method was to control for potential selection bias and ensure a consistent, statistically-driven process for pair identification across all models and experiments.

We chose to focus on the S&P 500 universe not to optimize pair selection, but rather to ensure a pool of stocks likely to have meaningful news coverage. This approach enabled potential information gain from sentiment analysis while maintaining a broad, representative sample of the U.S. equity market.

Our selection criteria were designed to identify pairs with a statistical relationship, balancing long-term equilibrium with sufficient price divergence. Specifically, we focused on two key metrics:

1. Cointegration: We sought to minimize the p-value of the cointegration test, indicating a stronger long-term equilibrium relationship between the stocks.

2. Correlation: We maximized correlation up to a threshold of 0.8, ensuring the stocks moved similarly enough to be considered a pair, but not so closely as to eliminate trading opportunities.

This approach allowed us to identify pairs in a consistent, reproducible manner across all experiments, reducing the risk of cherry-picking or inadvertent bias in our analysis. Table 1 presents examples of pairs identified using this method.

Table 1: Example Stock Pairs Identified by Statistical Method

Pair	Ticker 1	Ticker 2	Correlation	Cointegration p-value
1	GS	BLK	0.8344	0.016
2	JPM	CRM	0.8390	0.0265
3	INTC	C	0.8583	0.1598
4	WFC	UNP	0.7002	0.0293
5	HON	GS	0.6692	0.0188

It's important to note that these pairs were not selected for their potential profitability, but rather as a consistent dataset across which we could evaluate our various models. This approach allowed us to focus on our primary research question: the relationship between model complexity and trading strategy efficacy, while minimizing the impact of pair selection on our results.

1.2 Enhanced Feature Engineering and Selection

To capture the multifaceted nature of financial markets, we expanded our feature set to encompass over 200 technical indicators for each stock and their spreads. This extensive set of features allowed us to model a wide range of market dynamics and patterns, potentially improving our predictive power.

A key innovation in our approach was the incorporation of sentiment analysis. Using natural language processing techniques, we analyzed financial news headlines to derive sentiment scores. Interestingly, as shown in Figure 1, the 60-day moving average of positive news mentions emerged as the most influential feature in our models.

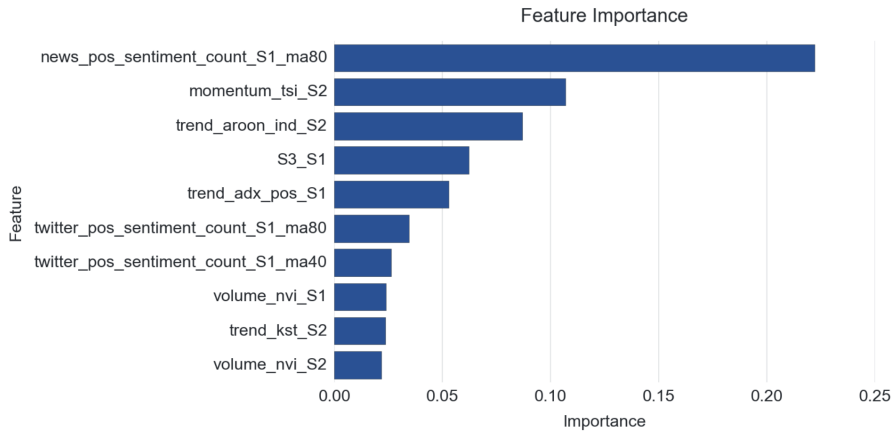


Figure 1: Top 10 Features Ranked by Importance

This finding underscores the significant role that alternative data sources, particularly those capturing market sentiment, can play in predicting stock price movements. It suggests that our models were able to extract valuable information from news sentiment that was not fully reflected in traditional price and volume data.

1.3 Model Development and Evaluation

To thoroughly investigate the impact of model complexity on trading performance, we implemented and compared a diverse range of algorithms:

1. Gradient Boosting Regressor
2. Random Forest Regressor
3. LightGBM Regressor
4. Vanilla LSTM
5. Hypertuned LSTM
6. BiLSTM with Dropout

This selection encompasses both traditional machine learning approaches and more complex deep learning architectures, allowing us to assess the trade-offs between model simplicity and predictive power.

We trained each model to forecast the spread between selected stock pairs, using historical data from 2021-2022 as our training set and data from 2023-2024 as our test set. This temporal split allowed us to evaluate how well our models generalized to future, unseen market conditions.

For our more complex LSTM models, we employed advanced hyperparameter tuning techniques. Specifically, we used HyperBand, an algorithm that efficiently allocates resources to the most promising hyperparameter configurations. We set a maximum budget of 100 epochs, with up to 10 epochs per configuration. Figure 2 visualizes this tuning process.

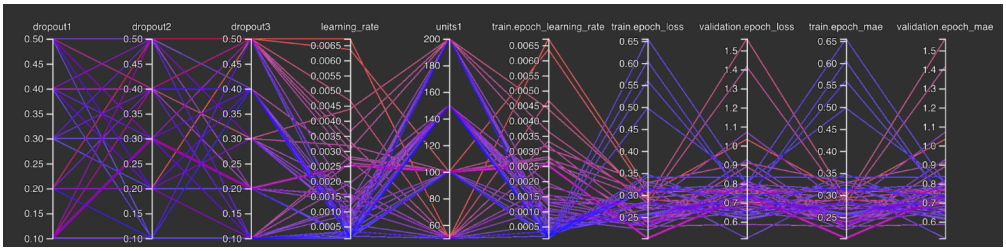


Figure 2: HyperBand Visualization of LSTM Hyperparameter Tuning

To illustrate the difference in model complexity, we present the architectures of our vanilla LSTM (Figure 3) and hypertuned LSTM (Figure 4) models. The vanilla LSTM consists of two LSTM layers with 64 units each, while the hypertuned version has a significantly more complex architecture with nearly 15 times the number of trainable parameters (2,062,205 vs. 147,755).

Model: "Vanilla LSTM"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 3, 50)	29,000
lstm_1 (LSTM)	(None, 50)	20,200
dense (Dense)	(None, 1)	51

Total params: 147,755 (577.17 KB)

Trainable params: 49,251 (192.39 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 98,504 (384.79 KB)

Figure 3: Vanilla LSTM Architecture

Throughout the training process, we closely monitored model performance to ensure proper learning and to detect any issues such as overfitting or underfitting. Figure 5 displays the training and validation curves for our LSTM and BiLSTM models, providing insight into the learning dynamics of these complex architectures.

This rigorous development and evaluation process allowed us to comprehensively assess the relationship between model complexity and predictive performance in the context of our pairs trading strategy.

Model: "Tuned LSTM"

Layer (type)	Output Shape	Param #
lstm_2 (LSTM)	(None, 3, 200)	236,000
dropout (Dropout)	(None, 3, 200)	0
lstm_3 (LSTM)	(None, 3, 200)	320,800
dropout_1 (Dropout)	(None, 3, 200)	0
lstm_4 (LSTM)	(None, 100)	120,400
dropout_2 (Dropout)	(None, 100)	0
dense_1 (Dense)	(None, 100)	10,100
dense_2 (Dense)	(None, 1)	101

Total params: 2,062,205 (7.87 MB)

Trainable params: 687,401 (2.62 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 1,374,804 (5.24 MB)

Figure 4: Hyperband Tuned LSTM Architecture

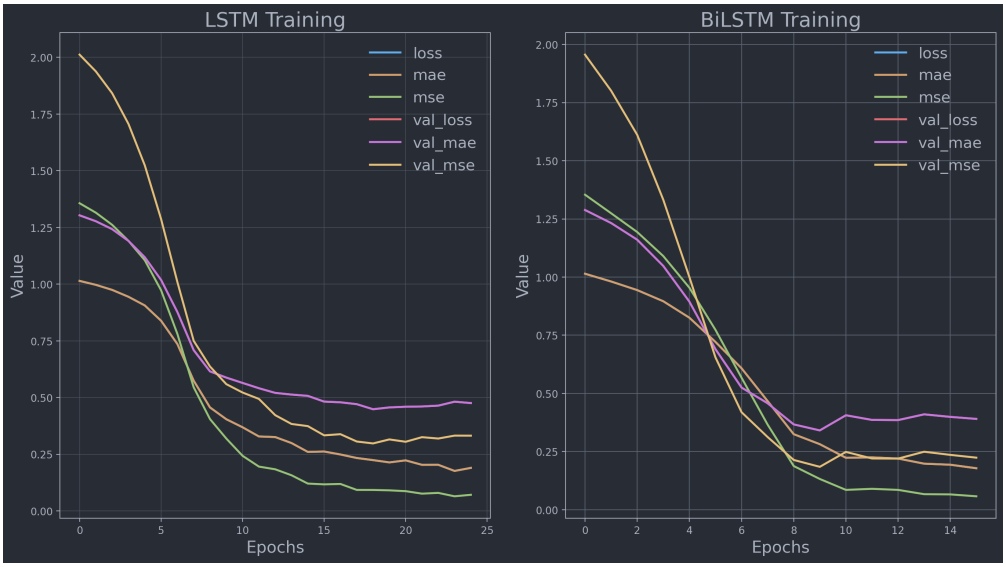


Figure 5: LSTM and BiLSTM Training and Validation Curves

1.4 Model Performance Analysis

Our results revealed a clear trend in performance as a function of model complexity. The more sophisticated models, particularly the hypertuned LSTM and BiLSTM with dropout, demonstrated superior performance in capturing the dynamics of the spread. This is likely due to their ability to process the large, complex feature set we engineered, including non-linear technical indicators and sentiment data.

Interestingly, we observed that simpler models performed better in certain aspects of our evaluation. This phenomenon can be attributed to the unique market conditions present in our test set (2023-2024), which coincided with a significant bull market featuring over 30

The non-stationarity of the true spread caused by this bull run period presented a challenge for our models. Those that were more biased in the negative direction inadvertently performed better by effectively longing the market. This observation highlights a fundamental issue with building our strategy within a purely frequentist framework, which struggled to generalize given the underlying shift in the test distribution.

To address this limitation, we adapted our approach to incorporate Bayesian principles. By adjusting the distribution of the predicted z-score spread during each trading day based on prior information, we were able to create more adaptive models. This Bayesian approach allowed our models to dynamically update their beliefs about the spread based on observed data, resulting in improved performance and enhanced robustness to non-stationary market environments.

These findings underscore the importance of considering market conditions and adapting model architectures accordingly when developing trading strategies. They also highlight the potential benefits of combining traditional machine learning techniques with more advanced approaches like deep learning and Bayesian methods to create more robust and adaptive trading systems.

1.5 Backtesting Results

We implemented a simple mean reversion strategy for backtesting. The strategy takes the model's predicted z-score spread and executes trades when the spread deviates significantly from its mean.

Table 2 summarizes the backtesting results across all models.

Table 2: Backtesting Results Across Models

Model	Final Portfolio Value	Annualized Returns	Sharpe Ratio	Sortino Ratio	Max Drawdown	CAGR	r2	RMSE
GradientBoostingRegressor	\$180,549.02	44.40%	2.6620	3.8357	-34.95%	44.33%	0.6582	0.7213
RandomForestRegressor	\$150,277.44	28.82%	1.8627	2.5793	-40.26%	28.75%	0.6885	0.6887
LGBMRegressor	\$153,462.12	30.51%	1.8692	2.6689	-40.01%	30.45%	0.6634	0.7158
Vanilla LSTM	\$118,982.87	13.85%	0.6610	0.4621	-81.16%	13.85%	-0.1247	1.3369
LSTM	\$221,007.24	80.75%	4.1318	7.4471	-20.96%	80.72%	0.4565	0.9293
BiLSTM with Dropout	\$203,613.07	70.02%	3.3429	4.3380	-24.15%	69.99%	0.4087	0.9694

2. Discussion and Interpretation

2.1 Model Performance

Our results show a clear trend in performance as a function of model complexity. The more complex models, particularly the hypertuned LSTM and BiLSTM with dropout, demonstrated superior performance in capturing the dynamics of the spread. This is likely due to their ability to process the large, complex feature set we engineered, including non-linear technical indicators and sentiment data.

The hypertuned LSTM model achieved the highest annualized returns (80.75%) and Sharpe ratio (4.1318), indicating strong risk-adjusted performance. The BiLSTM with dropout also performed well, suggesting that the bidirectional architecture and regularization techniques were beneficial.

Interestingly, the simpler Gradient Boosting Regressor outperformed some of the more complex models, achieving the third-best performance overall. This highlights the importance of feature engineering and the potential of ensemble methods in capturing complex market dynamics.

2.2 Impact of Market Conditions

It's crucial to note that our test period (2023-2024) coincided with a significant bull market, with the S&P 500 seeing over 30% gains. This market condition likely influenced our results, potentially favoring strategies that took more long positions.

To address the non-stationarity caused by the bull market, we adapted our strategy to incorporate a Bayesian approach. This allowed us to adjust the distribution of the predicted z-score spread during each trading day based on prior information, resulting in improved performance and robustness to changing market conditions.

2.3 Importance of Alternative Data

The inclusion of sentiment analysis as a feature proved valuable, with the 60-day moving average of positive news mentions contributing significantly to model performance. This underscores the potential of alternative data sources in capturing market dynamics not reflected in traditional price and volume data alone.

3. Conclusion and Future Work

Our study demonstrates a clear relationship between model complexity and trading strategy performance, with more sophisticated models generally outperforming simpler ones. However, the strong performance of the Gradient Boosting Regressor highlights that simpler models can still be highly effective when coupled with robust feature engineering.

The incorporation of sentiment analysis and the adoption of a Bayesian approach to handle non-stationarity were key innovations that improved our strategy's performance and adaptability.

Future work could explore:

1. Expanding the alternative data sources to include macroeconomic indicators and social media sentiment.
2. Investigating more sophisticated ensemble methods that combine predictions from multiple model types.
3. Extending the backtesting period to include diverse market conditions for a more comprehensive evaluation.
4. Exploring advanced LSTM variants such as Convolutional LSTM (ConvLSTM) for potentially improved performance on complex sequences.
5.

In conclusion, while our results are promising, it's important to note that they are based on a specific market period and may not generalize to all market conditions. Continuous monitoring, adaptation, and rigorous risk management remain crucial for any trading strategy implementation.

References

Amine Bakhach, Ahoora Rostamian, and John G O'Hara. Cnn-lstm based framework for trading signal generation using tick bars/candlesticks. *Neural Computing and Applications*, 34:17193–17205, 2022.

Kumar Hirdesh, Piyush Agarwal, and Monika Rani. Comparative study of lstm and dnn for daily stock forecasting. In *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pages 1080–1085. IEEE, 2020.

Sangwoo Lee and Won-Seok So. Multimodal reinforcement learning for stock trading using image and time series data. In *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1575–1577. IEEE, 2020.

Otabek Sattarov and Jaeyoung Choi. Multi-level deep q-networks for bitcoin trading strategies. *Innovative Methods in Financial Analysis*, 2024.

Jaimin Shah, Darsh Vaidya, and Manan Shah. A comprehensive review on multiple hybrid deep learning approaches for stock prediction. *Intelligent Systems with Applications*, 16:200111, 2022.

Alessio Staffini. Stock price forecasting using generative adversarial networks. *Fuzzy AI Applications in Finance*, 8:837596, 2022.

Zhizhao Xu and Chao Luo. Adaptive learning for pairs trading with double deep q-network and temporal difference learning. *Engineering Applications of Artificial Intelligence*, 126:107148, 2023.

Xiaodong Yan, Justin H Qian, Jiahui Ma, Alexander Zhang, Samuel E Lennon, Meng-Ping Bu, Kyle J Lin, Xin Liu, Han Wang, Vinod K Sangwan, et al. Reconfigurable heterojunction transistors for in-sensor support vector machine classification. *Nature Electronics*, 6(10):777–785, 2023.